

```
2
+- Mvns_ComputerSystem
+- CIM_ComputerSystem
+- CIM_ManagedElement
+- Mvns_ComputerSystem [ClassName="Mvns_ComputerSystem", CreationClassName="CIM_ComputerSystem", CIM_EnabledLogicalElement, CIM_CreatedElement, CIM_CreatedInClass="CIM_ComputerSystem.CreationClassName", CIM_CreatedObjectID="3F09F9F37C16"]
29
+- CIM_ComputerSystem [ClassName="CIM_ComputerSystem", CIM_EnabledLogicalElement, CIM_CreatedElement, CIM_CreatedInClass="CIM_ComputerSystem.CreationClassName", CIM_CreatedObjectID="3F09F9F37C16"]

PROPERTY_COUNT
COLLATION
OWNER
NAME-SPACE
PATH
+- MvnsNodeList
+- CIM_ComputerSystem [ClassName="CIM_ComputerSystem", CIM_EnabledLogicalElement, CIM_CreatedElement, CIM_CreatedInClass="CIM_ComputerSystem.CreationClassName", CIM_CreatedObjectID="3F09F9F37C16"]
+- Microsoft Virtual Computer System
+- Msvn_ComputerSystem
+- Microsoft Virtual Computer System
+- TESTUMI
22
22
5
20090508025634.000000-000
3F09F9F37C16
236728
22

-----
```

Visual Basic Database

Mengupas pemrograman database menggunakan Visual Basic
untuk pembuatan aplikasi toko

Herry Raditya Wibowo
Jubilee Enterprise

Visual Basic Database

Visual **B**asic **D**atabase

Herry Raditya Wibowo
Jubilee Enterprise

PENERBIT PT ELEX MEDIA KOMPUTINDO



KOMPAS GRAMEDIA

Visual Basic Database

Herry Raditya Wibowo & Jubilee Enterprise

©2014, PT Elex Media Komputindo

Hak cipta dilindungi undang-undang

Diterbitkan pertama kali oleh

Penerbit PT Elex Media Komputindo

Kelompok Gramedia, Anggota IKAPI, Jakarta 2014

elizabet@elexmedia.co.id

121141677

ISBN: 978-602-02-4584-3

Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit.

Dicetak oleh Percetakan PT Gramedia, Jakarta

Isi di luar tanggung jawab percetakan

Kata Pengantar

Aplikasi yang utuh dan memiliki nilai jual, umumnya melibatkan fungsi database. Oleh karena itulah, untuk menjadi seorang programmer yang "serius", pengenalan dan penguasaan database cukup penting untuk dilakukan.

Buku ini secara khusus mengupas fungsi database pada Visual Basic menggunakan MS SQL Server 2014 Express sebagai alat bantunya. Dengan demikian, Anda bisa mempraktikkan ilmu pemrograman database secara mudah dan murah.

Untuk membantu Anda, buku ini secara khusus mengupas pembuatan aplikasi toko. Dengan begitu, saat pembahasan berakhir, Anda menguasai ilmu pemrograman database dan mendapatkan aplikasi toko yang bisa Anda modifikasi sendiri.

Akhir kata, semoga buku ini bermanfaat bagi Anda dan bisa membantu penguasaan database di lingkungan Visual Basic.

Yogyakarta, 4 Juni 2014

Gregorius Agung

Founder Jubilee Enterprise

"Information Technology is Our Passion and Book is Our Way"

Do you need top-notch IT Book? Just thinkjubilee.com

Script aplikasi toko yang dibahas, dapat diunduh
dari situs: www.thinkjubilee.com/download/

Daftar Isi

Kata Pengantar	v
Daftar Isi	vii

BAB 1 INSTALASI VISUAL STUDIO EXPRESS 1

Microsoft Visual Studio Express 2013 Edition	2
Microsoft SQL Server 2014 Express.....	11

BAB 2 MENGENAL VISUAL STUDIO 2013 EXPRESS FOR WINDOWS DESKTOP 25

Default Form	27
Toolbox.....	29
Menambahkan Tool pada Form	30
Menambahkan Label pada Form	32
Pengantar Property	36
Apa Itu Property?.....	37
Menambahkan Warna	41
Menyimpan Project Anda	45

BAB 3 MENGENAL DATABASE..... 47

Database dan Database Management Systems	47
SQL Server Management Studio.....	50

BAB 4 SQL.....	63
Relational Database.....	63
Mengambil Data: Query SQL SELECT	64
Clause SELECT ... FROM	65
Clause WHERE.....	66
Clause ORDER BY.....	70
Penggunaan SQL pada Aplikasi.....	72
Mengambil Data dari Dua buah Table	73
Mengambil Data dari Tiga buah Table.....	75
Menyimpan Data	75
Mengubah Data	76
Menghapus Data	77
BAB 5 PERSIAPAN PROJECT.....	79
Perancangan Aplikasi	79
Persiapan Workspace.....	80
BAB 6 FORM IMPORT DATA.....	91
Pemrograman Database	95
DataSet dan DataAdapter	97
Menampilkan Data dalam DataSet.....	98
BAB 7 FORM BARANG - LIHAT SAJA.....	111
BAB 8 FORM BARANG	121
BAB 9 FORM KATEGORI, USER, DAN CUSTOMER	139
Form Kategori	139
Form User.....	144
Form Customer.....	153

BAB 10 FORM JUAL	161
Sub-Routine setupLayout()	181
Event JualDGV_CellValidating	181
Event btnBaru_Click	181
Event TextChanged	182
BAB 11 FORM LOGIN DAN FORM UTAMA	191
Form Login.....	191
Form Utama	195
Lampiran: Tipe Data	205
Tentang Penulis	209

BAB 1

Instalasi

Visual Studio

Express

Secara pribadi, penulis percaya bahwa saat ini adalah saat yang paling baik untuk menjadi seorang pengembang perangkat lunak. Dalam dunia yang demikian berkembang, di mana setiap perusahaan menggunakan perangkat lunak, para pengembang memiliki peran dan tanggung-jawab yang sedemikian besar dalam memajukan inovasi antara berbagai macam industri. Apakah itu mengembangkan perangkat lunak untuk suatu korporasi ataupun pengguna akhir, apakah itu memanfaatkan mobilitas yang dimungkinkan dari perangkat komunikasi ataupun ukuran yang dimungkinkan dengan *cloud*, apakah itu memupuk seni pemrograman sebagai sebuah hobi ataupun profesi, apakah itu seorang pemrogram pemula atau ahlinya, para pemrogram di manapun juga memiliki potensi untuk membangun suatu solusi yang kreatif dan menyeluruh yang memungkinkan untuk mengubah dunia.

Dalam konteks tersebut, penulis akan mengajak Anda untuk mengenal dan memahami aplikasi pengembang perangkat lunak, dan menggunakan kannya untuk mengembangkan suatu solusi yang akan bisa digunakan untuk Anda sendiri.

Dan salah satu yang paling mudah didapat saat ini adalah **Microsoft Visual Studio 2013**. Versi **Professional** dari aplikasi ini dijual pada harga **US\$499**. Cukup mahal, jika Anda adalah seorang programmer pemula. Karena itu, pembahasan dalam buku ini akan menggunakan versi **Express**. Versi **Express** ini adalah versi yang lebih ringan, dengan beberapa batasan, jika dibanding dengan versi **Professional**. Namun, dengan versi **Express**, Anda sudah bisa untuk membuat sebuah aplikasi penuh. Dan versi **Express** diberikan secara cuma-cuma oleh **Microsoft**. Anda cuma perlu mengunduhnya, dan melakukan registrasi dengan login menggunakan **Microsoft ID** Anda.

Pendukung kedua yang akan dipakai dalam pembahasan ini adalah **Microsoft SQL Server 2014 Express**.

Microsoft Visual Studio Express 2013 Edition

Hal pertama yang harus dilakukan adalah mengunduhnya dari website **Microsoft**.



Gambar 1-1. Unduh dari Microsoft Download Center

Ada empat versi utama dari versi **Express** ini:

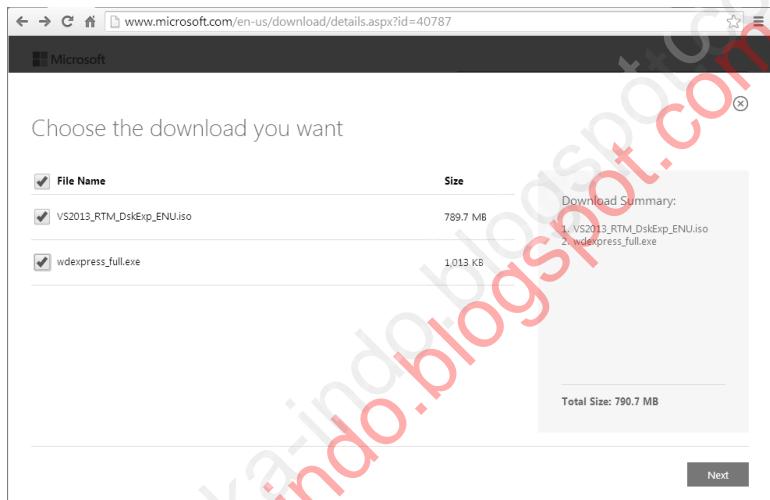
- **Visual Studio Express 2013 for Web**
Versi ini dikhususkan dalam pengembangan aplikasi berbasis web, termasuk di dalamnya fitur terintegrasi untuk platform **Windows Azure**.
- **Visual Studio Express 2013 for Windows**
Versi ini dikhususkan dalam pengembangan aplikasi untuk **Windows Store**.
- **Visual Studio Express 2013 for Windows Desktop**
Versi ini dikhususkan dalam pengembangan aplikasi **Windows** yang konvensional, khususnya dalam bahasa **C#, VB.Net** dan **C++**.

- **Visual Studio Team Foundation Server Express 2013**

Versi ini lebih ditekankan dalam penyediaan *source-control*, tracking untuk item kerja, *application lifecycle management*, dan *build automation* untuk tim yang terdiri dari maksimal lima orang pengembang.

Di sini Anda akan menggunakan **Visual Studio Express 2013 for Windows Desktop**, karena Anda akan membuat aplikasi yang berbasis pada **Windows-Form**.

Tekan tombol **Download**, dan Anda akan dibawa masuk ke tampilan berikutnya.

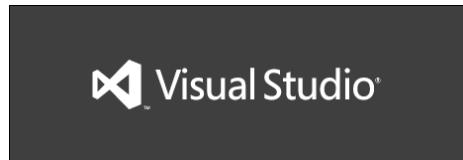


Gambar 1-2. Pilihan file untuk diunduh

Ada dua buah file yang bisa Anda unduh. File pertama adalah sebuah file **ISO**. Anda dapat mengunduhnya, dan kemudian membakarnya dalam sebuah **DVD**, untuk kemudian bisa melakukan instalasi secara *off-line* (tanpa terkoneksi ke Internet).

File kedua adalah *on-line* installer. File berukuran kecil, namun dalam instalasi, Anda harus selalu terkoneksi pada Internet.

Buku ini akan membahas instalasi menggunakan file installer **wdexpress_full.exe**. Jadi, unduhlah file tersebut, dan kemudian jalankan.



Gambar 1-3. Banner pembuka

Banner pembuka akan kemudian ditampilkan, dan setelah menunggu beberapa saat, kotak dialog setup akan ditampilkan.

Kotak dialog ini akan menunjukkan berapa besar file akan diunduh dari internet. Selain itu, ada beberapa checkbox yang harus Anda pilih untuk melanjutkan instalasi.



Gambar 1-4. Persetujuan

Pilihlah checkbox pertama. Checkbox kedua bisa Anda pilih, atau bisa juga Anda biarkan kosong.

Setelah itu, tekan tombol **Install** untuk melanjutkan proses instalasi.



Gambar 1-5. Proses instalasi

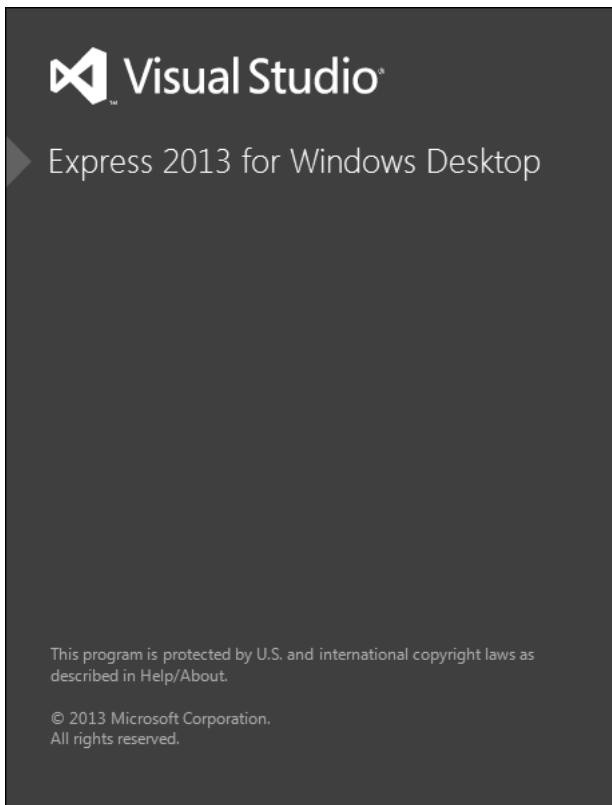
Proses pengunduhan ini akan bervariasi lamanya, tergantung dari kecepatan internet Anda. Dalam beberapa kasus, akan lebih mudah jika Anda memulai proses instalasi ini pada malam hari sebelum Anda tidur, dan membiarkan komputer dan internet bekerja semalam, dan kesokan harinya Anda akan menemukan bahwa proses instalasi sudah selesai (kecuali jika terjadi kesalahan, atau terputusnya internet Anda di malam hari).



Gambar 1-6. Proses instalasi selesai

Setelah proses instalasi selesai, Anda bisa mulai menjalankan aplikasi ini. Tekanlah tombol **Launch** untuk melanjutkan.

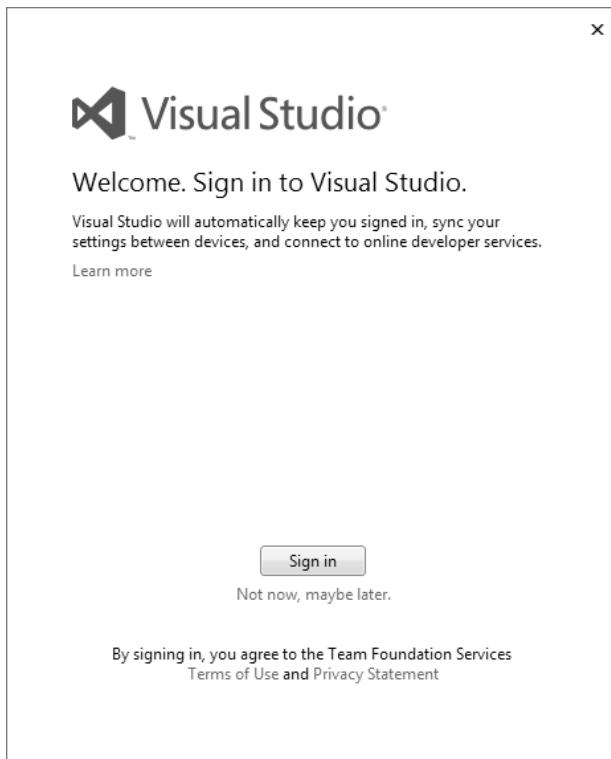
Proses **Launch** ini akan dimulai dengan membuka sebuah banner pembuka.



Gambar 1-7. Banner pembuka Visual Studio Express 2013 for Windows Desktop

Setelah beberapa saat, Anda akan dibawa pada langkah selanjutnya, di mana Anda diminta untuk sign-in pada **Visual Studio** dengan menggunakan **Microsoft ID** Anda, untuk meregistrasi aplikasi Anda.

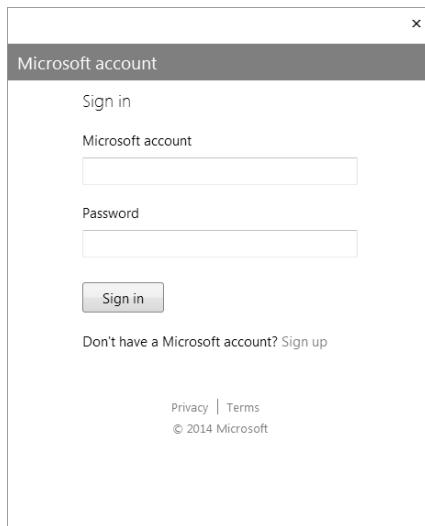
Anda bisa saja tidak melakukan proses ini, dan langsung menggunakan aplikasi Anda. Namun, tanpa melakukan registrasi, aplikasi Anda hanya akan bisa digunakan selama **30** hari saja. Dan pada akhir, Anda tetap akan diminta untuk melakukan proses sign-in lagi.



Gambar 1-8. Sign in to Visual Studio

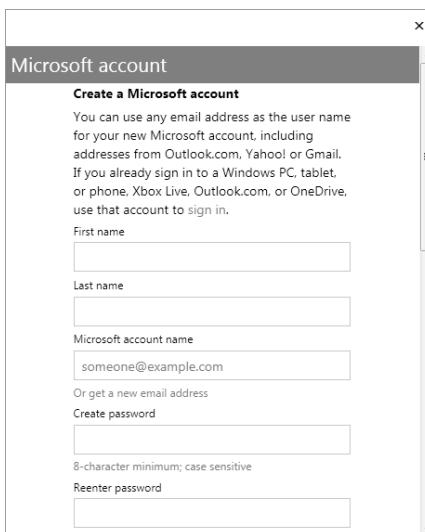
Proses sign in ini akan membantu Anda dalam proses sinkronisasi setting aplikasi Anda. Data setting Anda akan disimpan, dan akan bisa Anda gunakan pada mesin lain. Hal ini akan menjadi sangat membantu apabila misalnya Anda banyak melakukan perubahan setting pada komputer Anda, misalnya susunan menu, warna tertentu untuk workspace, dan sebagainya, dan Anda harus berpindah-pindah pada beberapa komputer dalam proses pemrograman Anda.

Tekanlah tombol **Sign in**, atau jika Anda belum mau melakukannya, Anda bisa menekan hyperlink **Not now, maybe later**.



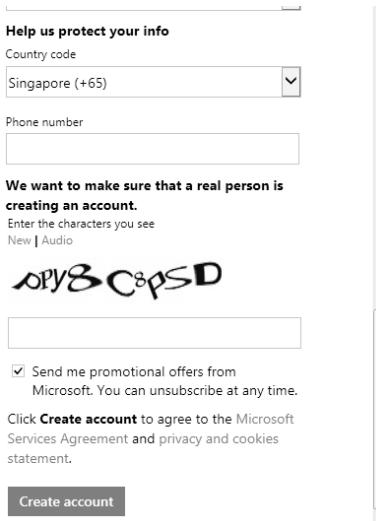
Gambar 1-9. Proses Sign in

Masukkan **Microsoft Account** dan **Password** Anda pada *textbox* yang tersedia. Jika Anda belum memilikinya, tekan hyperlink **Sign up**.



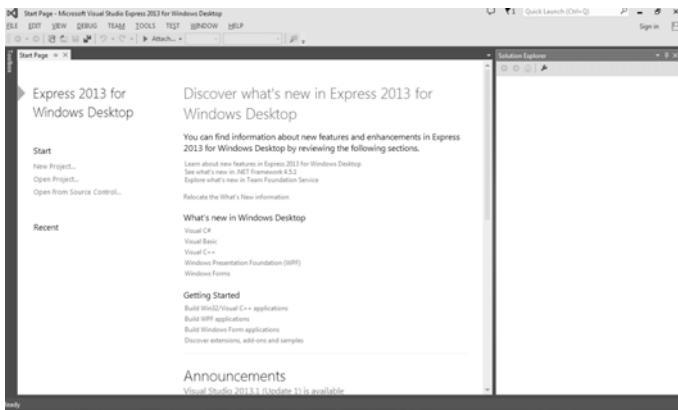
Gambar 1-10. Proses Sign up

Pada proses berikut ini, Anda akan membuat sebuah **Microsoft Account**. Isilah semua textbox yang diperlukan untuk membuat account Anda.



Gambar 1-11. Proses Sign up – lanjutan

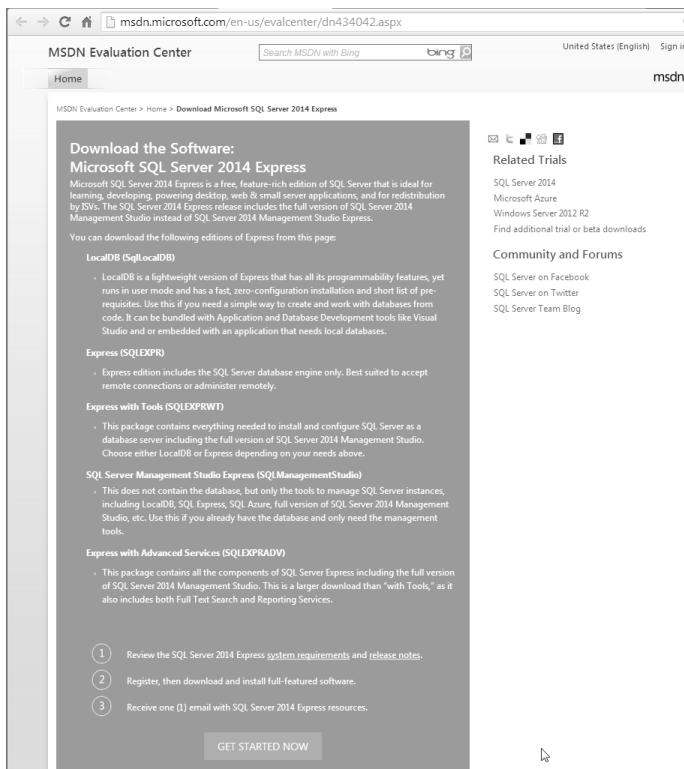
Setelah semua terisi dengan sempurna, tekanlah tombol Create account. Jika tidak ada masalah dengan data Anda, Anda akan dibawa masuk ke layar utama dari Visual Studio Express 2013 for Windows Desktop.



Gambar 1-12. Layar utama setelah proses Sign in

Microsoft SQL Server 2014 Express

Seperti sebelumnya, langkah pertama yang harus Anda lakukan adalah mengunduhnya dari website **Microsoft**.



Gambar 1-13. Unduh dari Microsoft Download Center

Pada tampilan ini, akan dijelaskan tipe-tipe file yang bisa Anda unduh kemudian. Anda bisa memilih **LocalDB** jika database yang akan diunduh hanya untuk implementasi ke *end-user*.

Namun, untuk pembelajaran Anda di sini, nantinya Anda akan memilih tipe file yang paling lengkap, yaitu **Express with Advanced Services**. Tipe ini juga yang paling besar ukurannya.

Tekan tombol **Download**, dan akan ditampilkan halaman untuk melakukan registrasi terlebih dahulu.

Sign in

Microsoft account What's this?

Keep me signed in

[Can't access your account?](#)
[Sign in with a single-use code](#)

[Don't have a Microsoft account? Sign up now](#)

Gambar 1-14. Registrasi

Jika Anda belum memiliki **Microsoft ID**, Anda bisa membuatnya melalui link **Sign up now** di bawah.

Microsoft

Profile Center

Home My profile Manage communications Help

Sign out

SQL Server 2014 Express
Thank you for taking the time to fill out the following online form. If you do not want to submit your information, click **Cancel**.

* Indicates a required field

My Name (Personal Information)

* First Name

* Last Name

* My E-Mail Address

To subscribe, select the communication(s) below and your preferred delivery format (HTML or Text).

Format

Subscribe HTML/Text Communication Description

TechNet Flash (US, XX)

Communication Preferences

Choose how Microsoft may use your contact information. Please note: This section may not reflect your current permission settings in our systems. Please click here to review your settings.

I would like to hear from Microsoft products, services, and events, including the latest solutions, tips, and exclusive offers. I would like to hear from Microsoft Partners, or Microsoft on their behalf, about their products, services, and events. Share or use my details with Microsoft Partners.

E-Mail Address E-Mail Address

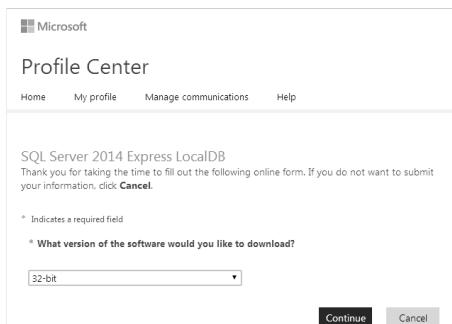
Note: These settings will not affect other newsletters or mandatory service communications from Microsoft. To learn how to set your contact preferences for other Microsoft sites, read the privacy statement.

By Downloading SQL Server 2014 Express software, you may receive emails from Microsoft with SQL Server 2014 Express resources.

Gambar 1-15. Membuat account Microsoft ID

Isilah semua isian yang diperlukan, pastikan tidak ada kesalahan. Gunakan email address yang aktif.

Tekan tombol **Continue** untuk melanjutkan.



Gambar 1-16. Menentukan versi Installer

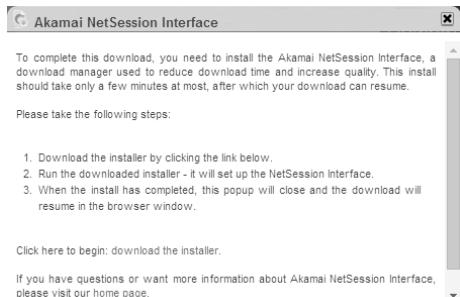
Di sini Anda hanya perlu untuk menentukan, apakah operating system Anda 64-bit ataukah 32-bit. Setelah Anda tentukan, tekan Continue.

A screenshot of the same SQL Server 2014 Express LocalDB download page. This time, the "Please select the language of your 32 Bit download" section is highlighted. It lists various languages with radio buttons: German, English (which is checked), Spanish, Italian, French, Japanese, Portuguese (Brazil), Chinese (Simplified), Chinese (Traditional), Korean, and Russian. At the bottom are "Back", "Continue", and "Cancel" buttons.

Gambar 1-17. Menentukan bahasa

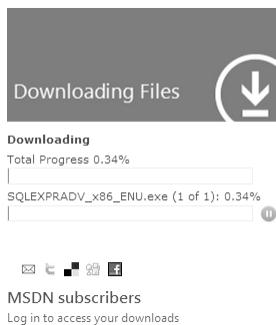
Selanjutnya, penentuan bahasa. Tidak ada **Bahasa Indonesia** di sini, jadi pilihlah bahasa lain yang Anda kuasai.

Setelah Anda menentukan bahasa yang akan digunakan, langkah selanjutnya adalah melakukan pengunduhan aplikasi download manager. Aplikasi ini digunakan di sini untuk membantu pengunduhan Anda, mengingat bahwa file yang akan diunduh berukuran besar.



Gambar 1-18. Akamai Download Manager

Tunggulah sampai proses pengunduhan selesai, dan kemudian jalankan aplikasi tersebut. Setelah Anda jalankan aplikasi tersebut, Anda bisa melanjutkan proses pengunduhan. Proses pengunduhan **SQL Server** ini akan dimulai secara otomatis begitu aplikasi download manager sudah terinstal. Namun, jika window download sebelumnya sudah Anda tutup, maka Anda harus mengulang lagi proses pengunduhan ini, dan setelah Anda menentukan bahasa, maka proses pengunduhan akan segera dimulai.



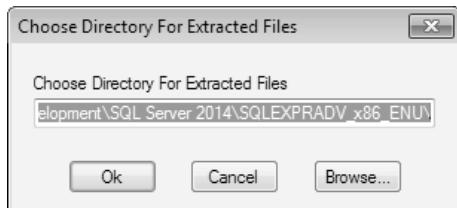
Gambar 1-19. Proses pengunduhan dimulai

Lamanya proses ini bervariasi, tergantung dari kecepatan internet Anda. Seperti sebelumnya, jika kecepatan tidak terlalu tinggi, maka akan lebih ideal apabila Anda melakukan pengunduhan ini pada malam hari, dan diharapkan keesokan harinya proses sudah selesai.



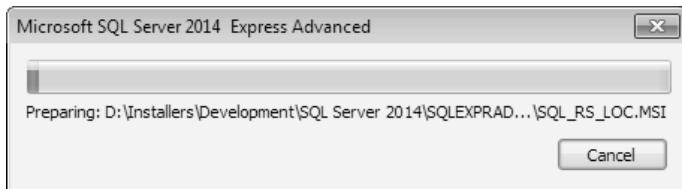
Gambar 1-20. Proses pengunduhan selesai

Setelah proses selesai, maka file tersebut akan tersedia pada lokasi penyimpanan Anda. Jalankan file installer tersebut.



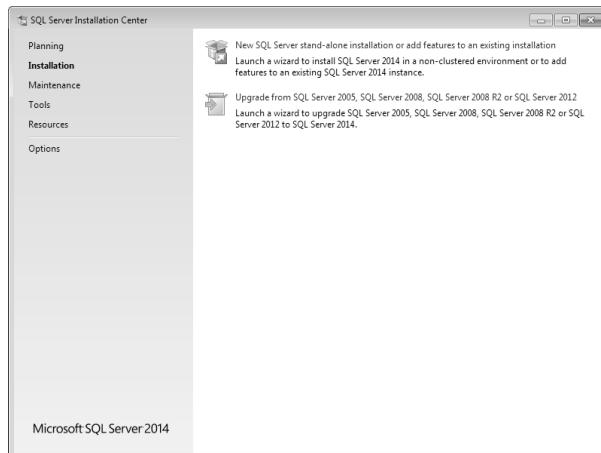
Gambar 1-21. Proses ekstraksi Installer

File akan diekstraksi, dan kemudian akan dijalankan.



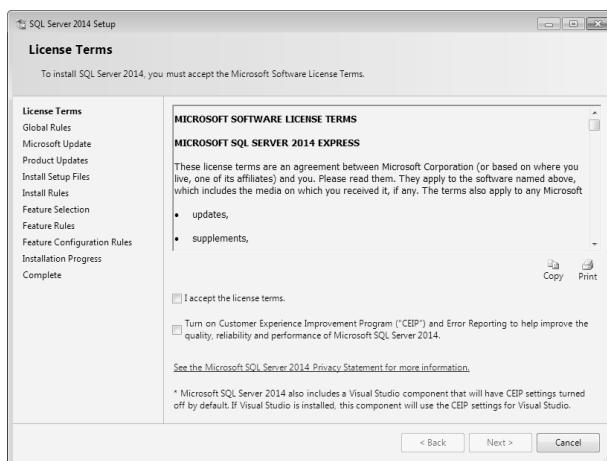
Gambar 1-22. Proses instalasi dimulai

Selanjutnya, proses ini akan menampilkan window **SQL Server Installation Center**.



Gambar 1-23. Pilihan instalasi

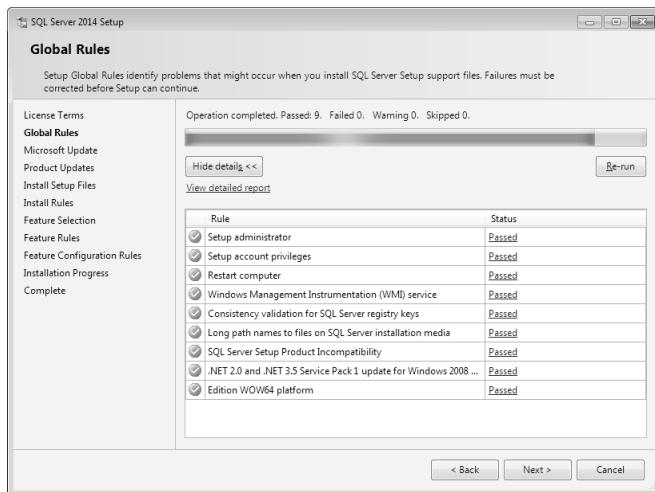
Pada kotak dialog yang pertama ini, Anda diminta untuk menentukan jenis instalasi, apakah akan melakukan instalasi baru, atau melakukan upgrade dari versi **SQL Server** yang lebih lama. Dan karena ini adalah instalasi baru, maka Anda bisa memilih pilihan pertama.



Gambar 1-24. Menyetujui kebijakan lisensi

Anda harus menyetujui kebijakan lisensi ini untuk bisa melanjutkan instalasi. Checkbox kedua mengatur Anda untuk mengaktifkan **Customer Experience Improvement Program** dan **Error Reporting**. Untuk mengetahui apa pengaruhnya dari instalasi Anda, Anda bisa memilih untuk memperlihatkan keterangannya dari link di bawahnya.

Karena itu, aktifkan checkbox persetujuan Anda pada lisensi, dan tekan tombol **Next**.

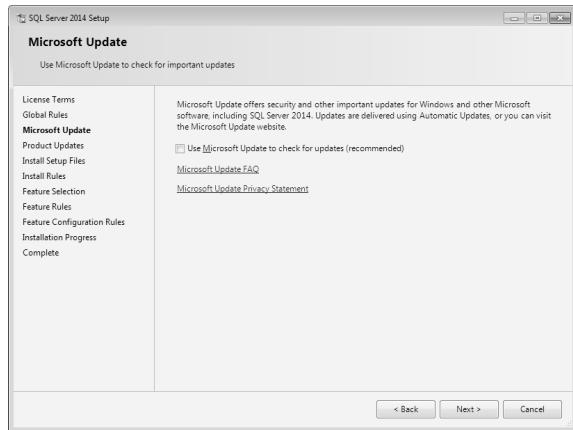


Gambar 1-25. Global Rules

Tahap selanjutnya akan memeriksa langkah-langkah yang harus diselesaikan sebelum proses instalasi bisa dilanjutkan. Jika masih ada hasil yang gagal, maka Anda harus memperbaikinya terlebih dahulu sebelum bisa melanjutkan kembali.

Mungkin ada sesuatu yang belum terinstal secara sempurna, hal itu akan menyebabkan Status-nya tertulis gagal dengan warna merah. Anda tidak perlu menutup window ini. Selesaikan langkah yang belum berhasil dilewati itu.

Setelah langkah tersebut Anda selesaikan, kembalilah pada window **Global Rules** ini lagi. Tekan tombol **Re-Run** untuk melakukan periksaan kembali pada semua langkah yang diperlukan. Dan jika memang langkah terakhir tersebut sudah Anda selesaikan dengan baik, maka tampilan **Global Rules** ini akan menunjukkannya dengan Status **Passed**.

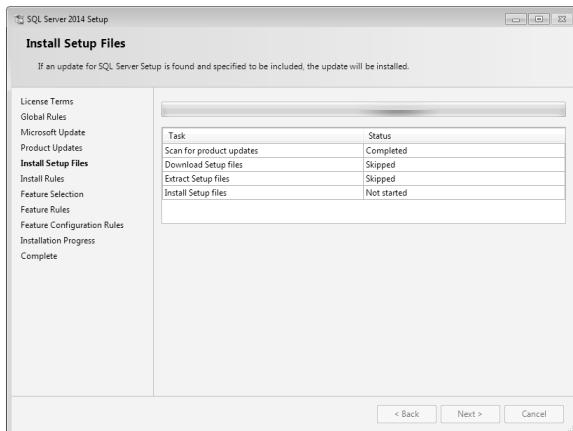


Gambar 1-26. Microsoft Update

Selanjutnya, Installer akan membuka sebuah window baru. Di sini, ada sebuah checkbox yang bisa Anda aktifkan, yang akan mengakibatkan Installer untuk mencari update dari **SQL Server 2014 Express** yang akan Anda instal.

Anda tentu saja bisa memilih untuk tidak mengaktifkan update, dengan cara meninggalkan checkbox tersebut tanpa memilihnya.

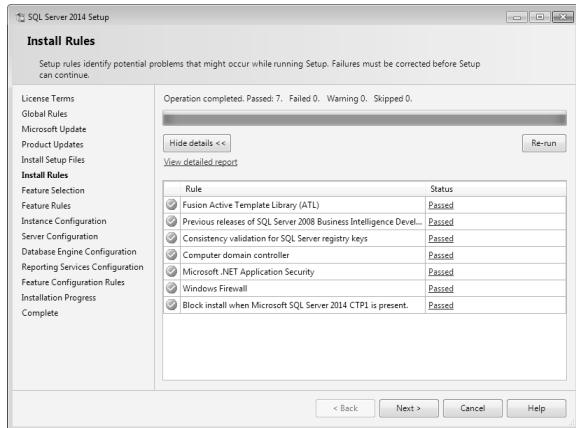
Proses selanjutnya adalah Installer mempersiapkan setup files yang diperlukan dalam instalasi.



Gambar 1-27. Mempersiapkan Setup Files

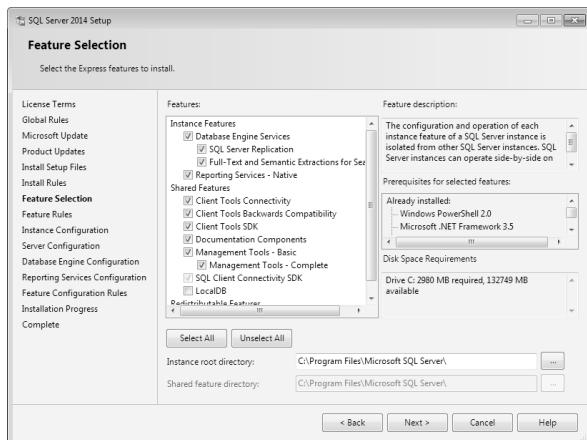
Langkah selanjutnya akan memeriksa apakah masih ada hal yang perlu dipersiapkan sebelum instalasi dilanjutkan.

Seperti sebelumnya, Anda harus mempersiapkan seluruh langkah tersebut, sebelum akhirnya Anda bisa melanjutkan instalasi.



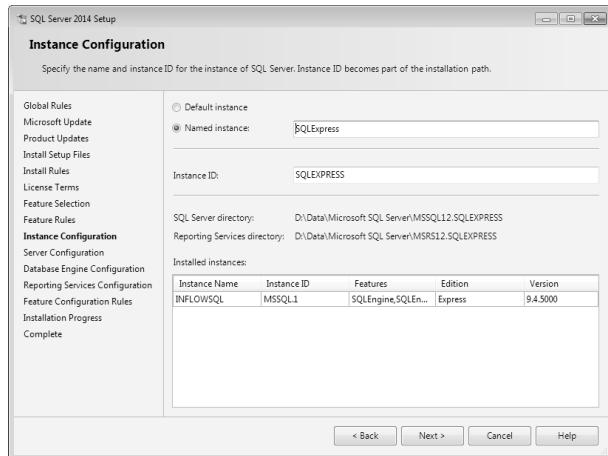
Gambar 1-28. Install Rules

Kemudian dilanjutkan dengan proses pemilihan fitur. Anda bisa memilih fitur apa pun yang ingin Anda instal, langsung dengan memilih dari checkbox yang tersedia untuk setiap fitur tersebut.



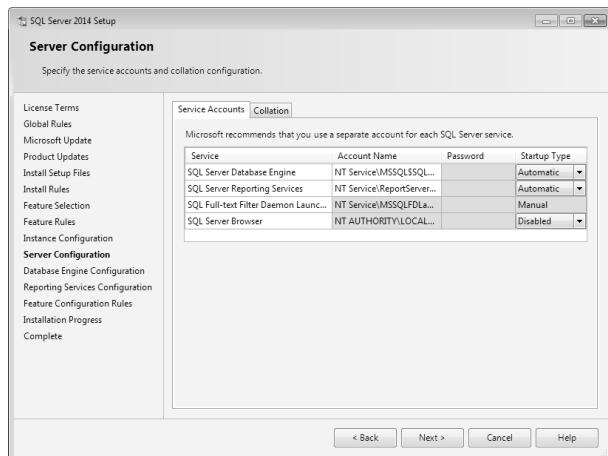
Gambar 1-29. Pemilihan fitur

Jika pada saat itu dalam komputer Anda sudah terinstal database **SQL Server** sebelumnya (mungkin termasuk dalam sebuah aplikasi lain), maka Installer akan menampilkan semua instance database yang sudah ada, sehingga Anda dapat memilih nama instance yang belum terpakai.



Gambar 1-30. Instance Configuration

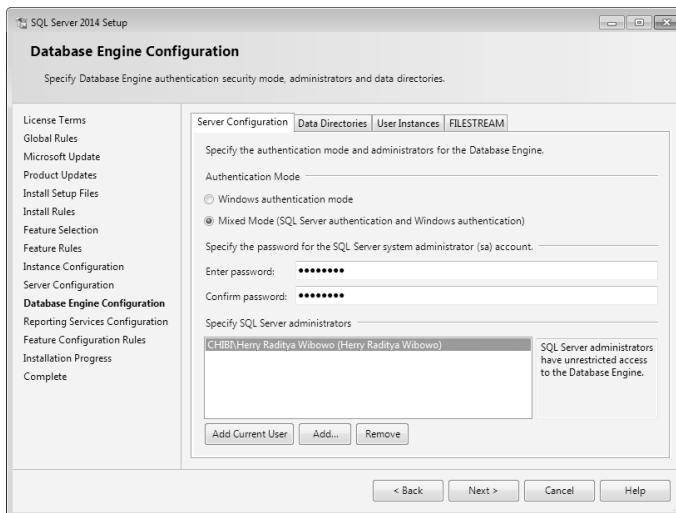
Setelah Anda menekan tombol **Next**, Anda akan sampai pada halaman **Server Configuration**.



Gambar 1-31. Server Configuration

Di sini, Anda menentukan bagaimana nanti masing-masing service dari **SQL Server** Anda akan dijalankan. Anda bisa mengaturnya untuk dijalankan begitu **Windows** Anda dinyalakan, atau Anda bisa mengaturnya untuk dijalankan secara manual kapan saja Anda perlukan.

Pilihan default cukup sesuai untuk saat ini, kecuali apabila Anda benar-benar ingin mengubah setting tersebut, dan Anda mengerti apa yang akan Anda lakukan.



Gambar 1-32. Konfigurasi database

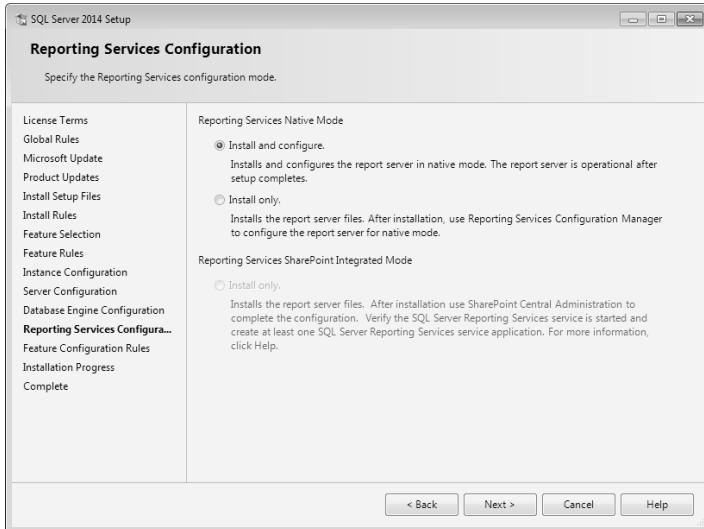
Pada kotak dialog ini, Anda harus menentukan tipe autentikasi database Anda. Sangat dianjurkan untuk menggunakan **Mixed Mode**, karena selain menggunakan autentikasi dari user account **Windows** Anda, juga digunakan user account database server Anda sendiri.

Jika Anda memilih **Mixed Mode**, Anda harus membuat sebuah password untuk account **SQL Server** system administrator (**sa**).

Setelah Anda buat password tersebut, Anda bisa memeriksa pada tab selanjutnya, dan periksalah bahwa semua setting yang ditampilkan sudah sesuai dengan keinginan Anda.

Setelah semuanya Anda periksa, dan setelah Anda memastikan semua sudah benar, Anda bisa melanjutkan pada tahap terakhir dari proses instalasi ini dengan menekan tombol **Next**.

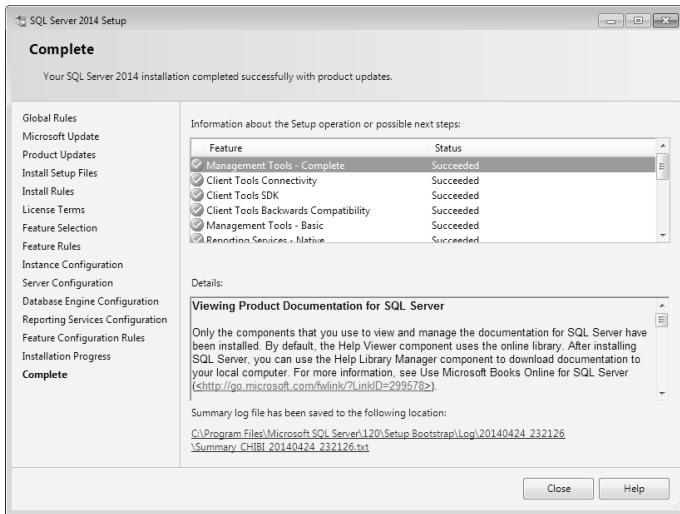
Jika sebelumnya Anda memilih untuk menginstal juga fitur Reporting **Services**, maka tampilan berikutnya akan meminta Anda untuk menentukan konfigurasi dari **Reporting Services** itu sendiri.



Gambar 1-33. Konfigurasi Reporting Services

Anda akan diminta untuk menentukan, apakah akan melakukan konfigurasi bersama dengan instalasi, ataukah hanya akan melakukan instalasi saja, tanpa konfigurasi.

Installer akan melanjutkan proses instalasi.



Gambar 1-34. Proses instalasi

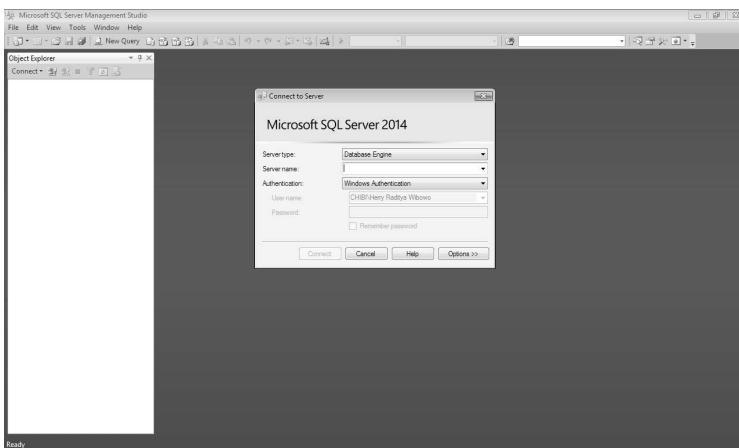
Setelah proses selesai, Anda bisa memastikan bahwa database Anda telah terinstal dengan sempurna dengan mencoba untuk membukanya. Untuk membukanya, Microsoft telah menyediakan sebuah aplikasi untuk membantu Anda, yaitu **Microsoft SQL Server Management Studio (SSMS)**.

Jalankah aplikasi tersebut, dan Anda akan disambut dengan sebuah banner pembuka.



Gambar 1-37. Banner pembuka

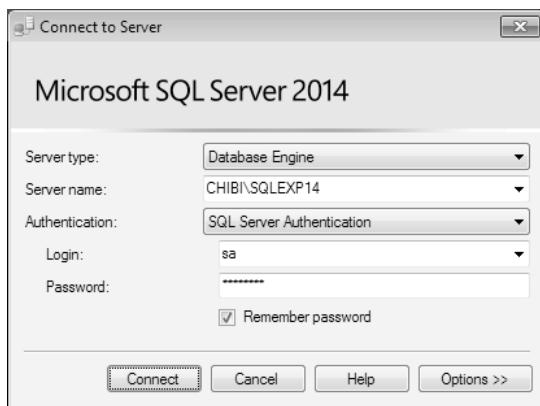
Dan kemudian masuk ke halaman awal.



Gambar 1-38. Halaman awal SSMS

Pada kotak dialog awal tersebut, Anda diminta untuk menentukan instance database mana yang akan Anda buka.

Selain itu, Anda juga bisa menentukan tipe autentikasi yang akan Anda gunakan. Jika sebelumnya Anda gunakan **Mixed Mode**, maka Anda bisa menggunakan kedua opsi autentikasi, yaitu **Windows Authentication**, di mana **SQL Server** akan menggunakan account **Windows** Anda untuk masuk, atau Anda bisa juga menggunakan **SQL Server Authentication**, di mana Anda masuk menggunakan user **sa** dengan password sebagai mana Anda atur pada halaman sebelumnya.



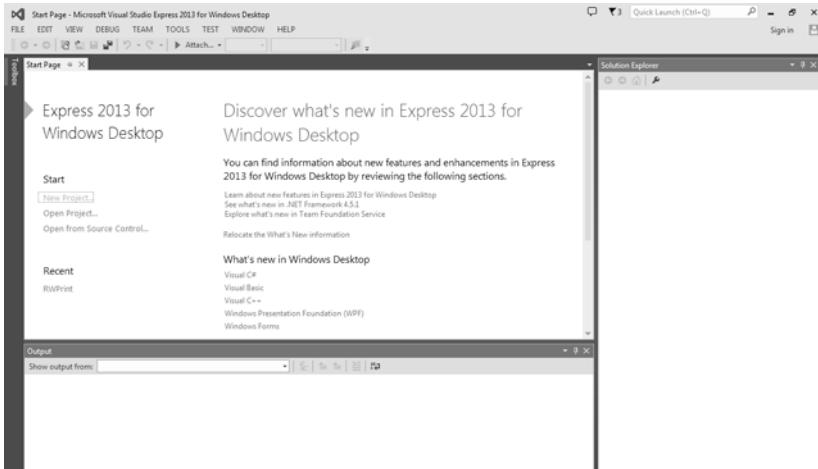
Gambar 1-39. SQL Server Authentication

BAB 2

Mengenal Visual Studio 2013 Express for Windows Desktop

Dalam bab sebelumnya, Anda telah melakukan instalasi aplikasi **Microsoft Visual Studio Express 2013 for Windows Desktop**. Carilah aplikasi tersebut dari dalam Start Menu Anda, dan jalankan.

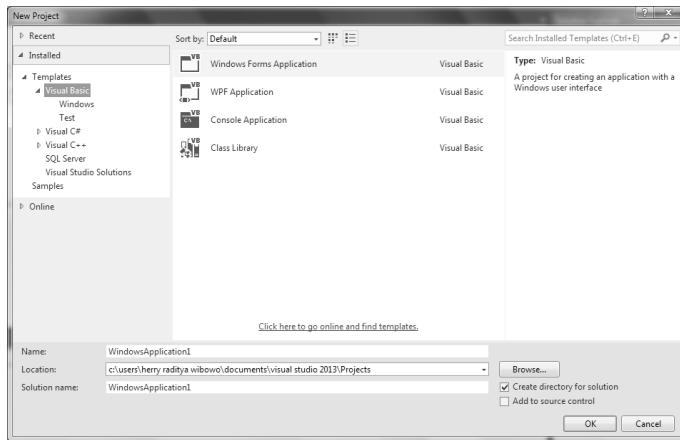
Pada saat pertama kali aplikasi tersebut terbuka, bisa Anda lihat tampilan awal sebagai berikut.



Gambar 2-1. Microsoft Visual Studio Express 2013 for Windows Desktop

Banyak sekali yang bisa dilakukan pada halaman awal ini. Namun, pada dasarnya di sinilah Anda bisa memulai sebuah project baru, atau membuka project yang sudah ada. Tab pertama, **New Project**, terpilih.

Tekan tab **New Project** tersebut untuk memulai. Pada saat Anda menekan tab tersebut, sebuah kotak dialog akan ditampilkan:



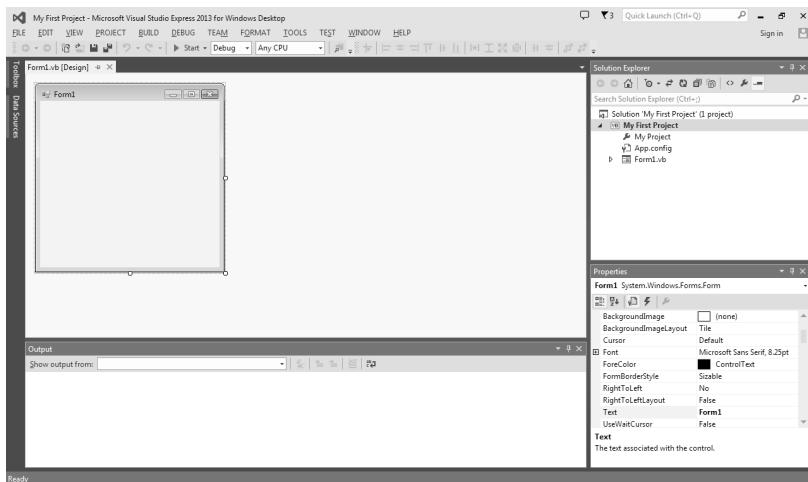
Gambar 2-2. Kotak dialog New Project

Sebagai awalnya, Anda akan memilih dari folder **Visual Basic** di sebelah kiri, opsi "**Windows Forms Application**". Ini berarti bahwa Anda akan membuat sebuah aplikasi yang akan dijalankan dalam sebuah komputer yang menggunakan sistem operasi Microsoft Windows.

Jika Anda perhatikan pada textbox **Name** di bagian bawah, Anda akan melihat bahwa saat ini isinya adalah **WindowsApplication1**. Ini adalah nama default dari project baru Anda. Tidak dianjurkan untuk tetap menggunakan nama tersebut (bayangkan jika pada akhirnya, nama-nama project Anda adalah "**WindowsApplication1**", "**WindowsApplication2**", dan seterusnya). Jadi, klik di bagian nama itu, dan gantilah nama project baru Anda menjadi **My First Project**.

Untuk **Location**, gunakan lokasi default-nya. Lokasi tersebut adalah sebuah folder di dalam folder "**My Documents**" Anda, bernama "**Visual Studio 2013**" dan dengan sub-folder "**Projects**". Sebuah folder baru akan dibuat, dan nama dari folder baru tersebut adalah sama dengan nama project baru Anda. Semua file yang nantinya akan ditambahkan dalam project Anda akan disimpan dalam folder ini.

Tekan tombol **OK**, dan IDE utamanya akan terbuka, dengan tampilan seperti berikut.



Gambar 2-3. Workspace utama Anda

Dalam workspace inilah proses pembuatan aplikasi **Windows** Anda akan dilakukan.

Default Form

Dalam lingkunan **VS Express 2013** ini, hal pertama yang perlu Anda perhatikan adalah sebuah kotak di bagian tengah-kiri. Itu adalah sebuah **Form**. Form ini adalah bagian tampilan dalam aplikasi Anda, di mana seorang user lain akan melihat jika aplikasi Anda dijalankan. Segera Anda akan belajar bagaimana cara membuat perubahan dari tampilan dasar form tersebut.

Untuk mencoba menjalankan form tersebut, jalankan langkah berikut:

- Dari menu bar, klik pada **Debug**
- Dari menu berikutnya, pilih **Start Debugging**
- Atau Anda bisa juga menekan tombol **F5** dari keyboard anda
- Aplikasi Anda akan dijalankan

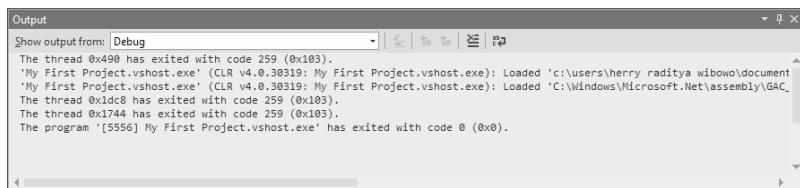
Selamat! Aplikasi pertama ada telah dibuat. Aplikasi itu akan memberikan tampilan sebagai berikut.



Gambar 2-4. Form1

Untuk menghentikan jalan aplikasi Anda, klik pada tombol X merah di sudut kanan-atas form tersebut. Kontrol akan kemudian dikembalikan lagi pada workspace pemrograman Anda.

Jika Anda perhatikan bagian bawah dari layar Anda, Anda akan melihat sebuah **Output Window**.



Gambar 2-5. Output Window

Jika Anda bandingkan antara form yang ditampilkan pada workspace Anda dengan form yang ditampilkan pada saat aplikasi berjalan, Anda akan bisa melihat bahwa kedua form tersebut hampir sama.

Pada dasarnya, **Visual Basic** memiliki dua lingkungan kerja yang berbeda, yaitu lingkungan **Design** dan lingkungan **Debug**. **Design** adalah waktu di mana Anda bisa bermain dengan form Anda, menambah textbox, tombol-tombol, text label, dan programnya sendiri. **Debug** adalah waktu di mana Anda bisa menguji program Anda dan melihat bagaimana aplikasi Anda berjalan (atau tidak berjalan).

Toolbox

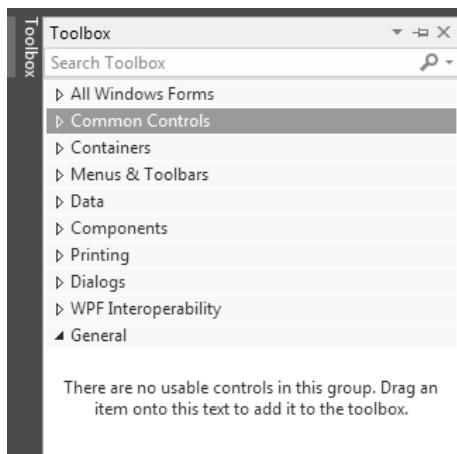
Benda-benda seperti halnya tombol, textbox, label, dan sebagainya adalah hal-hal yang bisa Anda tambahkan dalam form Anda. Mereka disebut dengan control, dan disimpan dalam **Toolbox** untuk kemudahan pencarian dan penggunaannya.

Toolbox bisa Anda dapatkan pada tepi layar sebelah kiri dan atas dari workspace Anda.



Gambar 2-6. Lokasi Toolbox

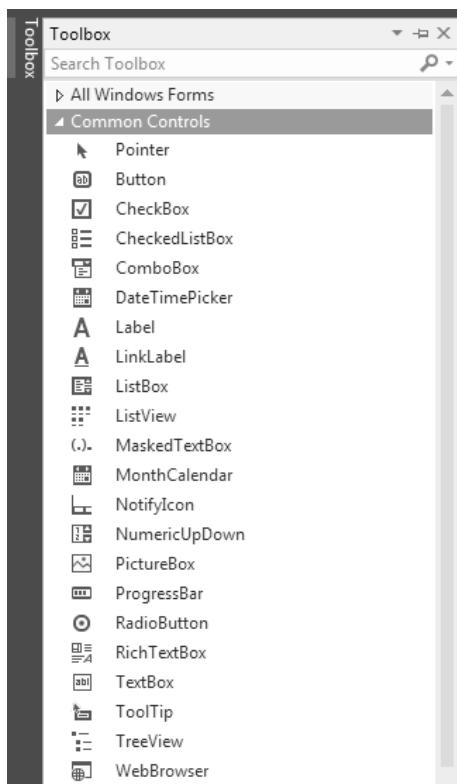
Untuk menampilkan semua isi dari **Toolbox**, pindahkan mouse Anda ke atas icon **Toolbox** tersebut, dan klik pada icon tersebut. Anda akan melihat **Toolbox** tersebut keluar dan tampil seluruhnya.



Gambar 2-7. Toolbox yang terbuka

Ada banyak kategori dalam **Toolbox** tersebut. Tool-tool yang akan paling sering Anda pakai adalah yang sudah dikelompokkan dalam kategori

Common Controls. Untuk melihat tool-tool apa saja di dalamnya, tekan pada panah kecil disamping **Common Controls**. Anda akan melihat daftar tool-tool tersebut.



Gambar 2-8. Tool dalam Common Controls

Jika Anda ingin untuk terus memperlihatkan **Toolbox** tersebut, tekan tombol **PIN** yang ada di samping tombol **X** pada ujung kanan atas dari **Toolbox** tersebut.

Menambahkan Tool pada Form

Cobalah sekarang untuk menambahkan sebuah textbox pada form Anda.

Dengan kategori **Common Control** terbuka, lakukan hal berikut:

1. Carilah tool **TextBox**.
 2. Klik-ganda pada tool tersebut.
 3. Sebuah textbox akan ditambahkan pada form Anda.
- Sebuah textbox akan ditambahkan langsung pada form Anda, dan akan diletakkan pada sudut kiri atas.



Gambar 2-9. TextBox pada Form

Untuk memindahkannya, cukup letakkan pointer mouse Anda pada textbox tersebut dan tariklah ke posisi yang lain.

Perhatikan bahwa ada kotak-kotak kecil di samping textbox tersebut. Itu adalah *sizing handles*, digunakan untuk mengubah ukuran textbox. Pin-dahkan mouse Anda pada salah satu kotak itu, dan pointer mouse Anda akan berubah menjadi sebuah panah dua arah. Tekan mouse Anda dan tariklah ke arah luar, dan ukuran dari textbox tersebut akan berubah.

Yang perlu Anda mengerti adalah bahwa Anda tidak bisa mengubah tinggi dari textbox tersebut, namun Anda bisa mengubah lebarnya. Hal ini karena secara default, sebuah textbox hanya memiliki satu baris text. Dan karena hanya satu baris saja, maka tinggi dari textbox tersebut tidak dapat diubah. Cara untuk mengubah tinggi sebuah textbox hanya bisa dilakukan jika textbox tersebut memiliki text yang multi-line (lebih dari satu baris).

Untuk persiapan pada langkah-langkah selanjutnya, silakan lakukan hal-hal berikut:

1. Buatlah dua buah textbox lain dan tambahkan pada form Anda.
2. Pastikan ukuran ketiga textbox adalah sama.
3. Urutkan satu sama lain ke bawah, dengan baris sela di antaranya.
4. Cobalah untuk membuat sesuai dengan contoh di bawah.

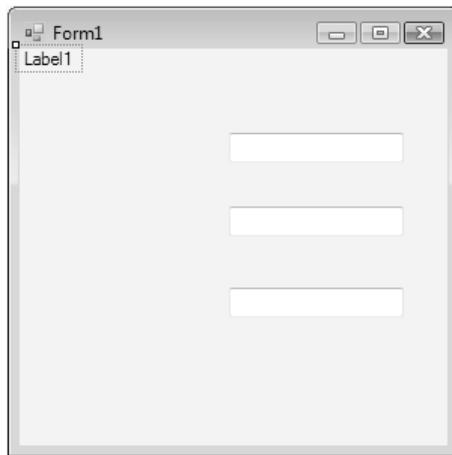


Gambar 2-10. Form dengan tiga textbox

Menambahkan Label pada Form

Di sini Anda akan menambahkan label pada form Anda agar user bisa mengerti kegunaan dari textbox yang ada.

1. Temukan tool **Label** dari **Toolbox**.
2. Klik-ganda pada tool tersebut.
3. Sebuah label akan ditambahkan pada form Anda.
4. Tampilan form Anda seharusnya akan seperti berikut:



Gambar 2-11. Label pada Form

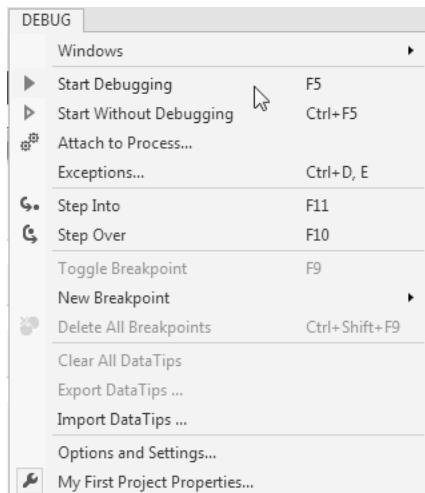
Klik pada label tersebut untuk meilihnya. Tahan, dan tarik label ke sebelah kiri dari textbox yang pertama

Buatlah dua buah label lagi, dan tempatkan pada sebelah kiri masing-masing textbox. Pastikan Anda memiliki tampilan form seperti di bawah ini.



Gambar 2-12. Posisi ketiga Label dan Textbox dalam Form

Untuk melihat bagaimana tampilan dari *form* Anda sebagai sebuah aplikasi, pilihlah dari menu **Debug – Start Debugging**. Atau Anda bisa juga tekan tombol **F5** dari keyboard Anda.



Gambar 2-13. Submenu Debug – Start Debugging

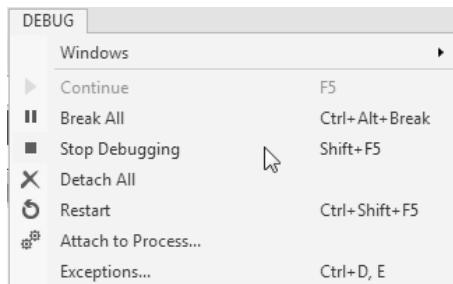
Aplikasi Anda akan dijalankan, dengan tampilan dasar sebagai berikut:



Gambar 2-14. Form in-action

Untuk menghentikan aplikasi, ada beberapa hal yang bisa Anda lakukan.

1. Tekan tombol **X** merah pada sudut kanan atas form Anda.
2. Pilih submenu **Debug – Stop Debugging**.
3. Tekan kombinasi tombol **Shift+F5** pada keyboard.



Gambar 2-15. Submenu Debug – Stop Debugging

Selain itu, Anda juga masih bisa untuk menghentikan aplikasi Anda dengan cara menekan tombol **Stop** pada **VB toolbar** di bagian atas *workspace* Anda, seperti ditunjukkan dalam gambar berikut.



Gambar 2-16. VB toolbar

Saat ini, Anda sudah memiliki sebuah form yang bisa berdiri sendiri, dengan tiga buah textbox di mana seorang user bisa mengisinya.

Namun, label yang ada tampak kurang bisa menjelaskan apa kegunaan dari form Anda. Dan juga, textbox yang ada hanya memiliki tipe data text yang bisa diisikan, secara default.

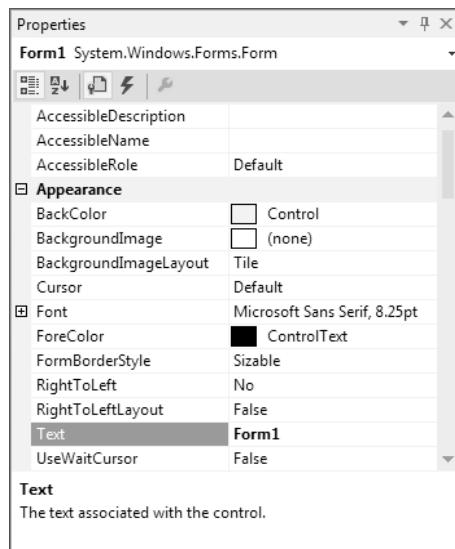
Bagaimana cara Anda untuk mengubah label supaya lebih informatif, dan juga mengubah isian textbox untuk tidak menggunakan tipe data text?

Untuk itu, Anda harus mempelajari mengenai **Property**.

Pengantar Property

Anda mungkin sudah mengetahui bahwa ada sebuah bagian di sebelah kanan workspace Anda, dengan banyak informasi, dan beberapa di antaranya menggunakan nama seperti “**AccessibleDescription**”, “**AccessibleName**”, dan seterusnya. Bagian itu adalah yang disebut window **Property**.

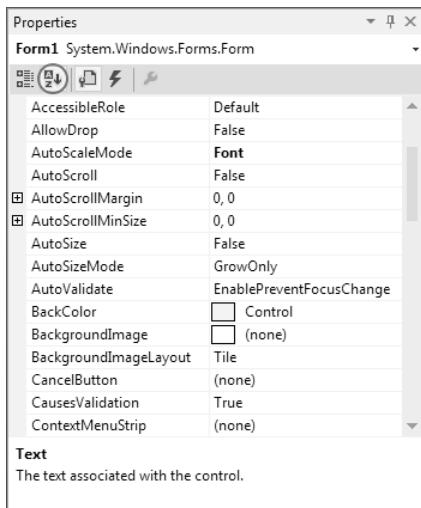
Kliklah pada bagian background form Anda, pada bagian yang berwarna abu-abu, dan bukan pada bagian *header*, *textbox*, atau *label*-nya. Kotak **Property** Anda akan tampil seperti berikut.



Gambar 2-17. *Property* untuk form Anda

Apa yang Anda lihat adalah sebuah daftar property yang dimiliki oleh form tersebut. Persis di sebelah kanan dari property tersebut adalah nilainya. Nilai-nilai tersebut adalah nilai default-nya, dan bisa diubah.

Sebelumnya, Anda bisa mengubah tampilan dari daftar property tersebut, dalam bentuk yang lebih mudah untuk dilakukan pencarian. Anda bisa mengubahnya dengan mengurutkan daftar tersebut. Tekanlah icon **Sort** yang terletak di baris sebelah atas dari window property tersebut.



Gambar 2-18. Property yang diurutkan

Hal ini akan membuat daftar property ini lebih mudah ditemukan.

Apa Itu Property?

Item-item yang Anda tambahkan ke dalam form Anda (*textbox* dan *label*), dan juga form itu sendiri, adalah sebuah **control object**. Anda bisa menganggap control sebagai sebuah benda, sesuatu yang bisa Anda lihat, pegang, dan pindahkan. Control memiliki property. Jika misalnya TV Anda adalah sebuah control, TV Anda juga memiliki property, seperti misalnya property tombol ON/OFF, property warna, property volume, property ukuran, dan sebagainya. Mengerti?

Property dari TV Anda akan memiliki nilai. Tombol ON/OFF hanya akan memiliki dua buah nilai, ON atau OFF. Ukuran memiliki sebuah rentang nilai, dari 22" sampai dengan 72", misalnya.

Dalam **Visual Basic**, Anda bisa mengubah property dari sebuah control dari dalam kotak **Properties** (nantinya, Anda juga akan bisa mengubahnya melalui pemrograman). Jika Anda kembali pada object Form Anda, dan semua daftar property maupun nilai-nilainya, Anda bisa mencoba untuk mengubah salah satu dari property yang ada.

Anda akan mengubah nilai dari property **Text** dari form tersebut.

Temukan property "**Text**" dari daftar, seperti pada **Gambar 2-17**.

Jangan bingung dengan nilai “**Form1**” yang tertulis di sebelah property “**Text**” tersebut. Hal itu hanya berarti bahwa saat ini, nilai dari property “**Text**” tersebut adalah kata “**Form1**”. Ini adalah nilai default-nya.

Untuk mengubahnya, klik mouse Anda pada bagian nilai di sebelah kanan “**Text**”, dan hapus kata “**Form1**”, kemudian tuliskan “**My First Form**”

Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	My First Form
TopMost	False
TransparencyKey	<input type="checkbox"/>
UseWaitCursor	False
WindowState	Normal

Gambar 2-19. Mengubah property form

Setelah mengubah nilainya, kembalilah pada form Anda, atau tekanlah tombol “**Enter**” setelah Anda mengisi nilainya.

Kata-kata “**My First Form**” akan tertulis pada bagian atas dari form Anda.



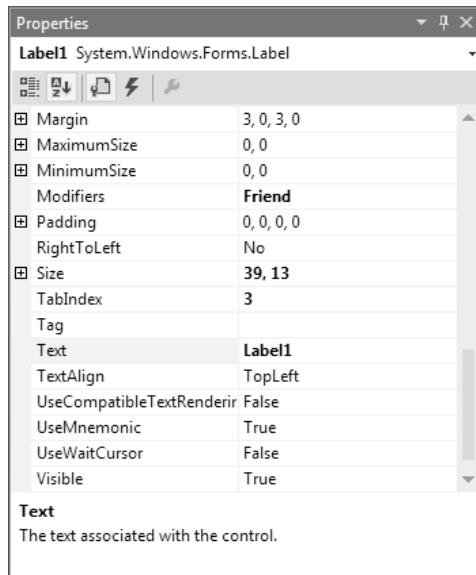
Gambar 2-20. Header form telah berganti

Sebagai yang bisa Anda lihat, kata “**My First Form**” tersebut sudah di-tempatkan langsung pada form Anda, pada bagian atasnya. Jadi, property

“Text” dan sebuah form adalah untuk menentukan apa yang akan Anda tampilkan pada bagian **title bar** dari form Anda.

Selanjutnya, Anda akan mengubah property dari label yang ada.

Pilihlah **Label1** dari form Anda. Kemudian periksalah kotak Property untuk label tersebut.



Gambar 2-21. Property untuk Label1

Anda bisa melihat bahwa property untuk Label cukup memiliki banyak perbedaan dibandingkan dengan property untuk Form. Kembali pada contoh TV Anda sebagai sebuah control. TV Anda akan memiliki tombol dan pengaturan yang berbeda dibandingkan dengan tombol dan pengaturan pada sebuah Lemari Es. Sebuah Label memiliki property yang berbeda dibandingkan dengan property dari sebuah Form.

Namun, Label memiliki banyak property yang sama dengan yang ada pada Form. Misalnya, dalam Label juga ada property “Text”. Gunanya juga sama, menambahkan/mengubah text yang ada pada tampilan.

Dengan **Label1** terpilih, carilah property “Text”, dan ketiklah nilai baru “First Name”. Setelah itu, kembalilah pada form Anda. **Label1** akan berganti menjadi “First Name”.



Gambar 2-22. Pengubahan Label1

Ubahlah property “Text” dari kedua label yang lain. Ubahlah menjadi nilai-nilai berikut:

Label2: Last Name

Label3: Telepon

Perhatikan bahwa ukuran label tersebut akan berubah menurut isinya, setelah Anda menekan tombol **Enter** pada saat Anda mengubah isinya. Anda mungkin perlu untuk mengatur letak dari label-label tersebut.

Setelah selesai melakukan pengubahan, form Anda akan tampak sebagai berikut.



Gambar 2-23. Setelah semua label diubah

Anda bisa mencoba untuk mengubah tampilan dari form Anda, posisi dari label maupun textbox Anda, dan lainnya.

Kliklah pada form tersebut, pada bagian yang bukan merupakan label ataupun textbox. Anda bisa melihat di sekelilingnya *sizing handle* untuk form tersebut. Pilih dan tariklah salah satunya untuk mengubah besar dari form Anda. Ubahlah posisi dari label maupun textbox Anda.



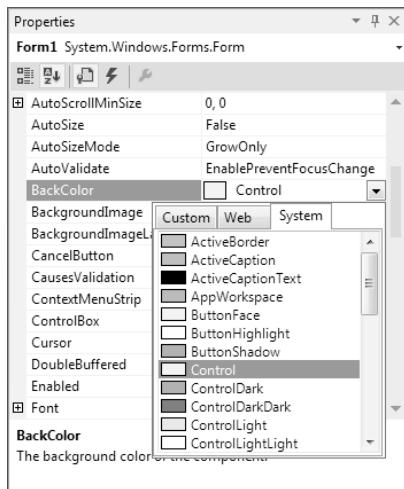
Gambar 2-24. Form setelah mengalami perubahan

Pilihlah **Debug > Start Debugging** untuk melihat tampilan dari form Anda. Pilihlah **Debug > Stop Debugging** untuk kembali ke lingkungan *workspace* Anda.

Menambahkan Warna

Mengubah warna dari **Form** berarti bahwa Anda harus mengubah salah satu dari property dari form tersebut, yaitu property **BackColor**.

Pilihlah property **BackColor** dari form tersebut. Kemudian tekanlah tombol panah kecil di sisi kanan dari property tersebut. Sebuah dropdown menu akan ditampilkan.

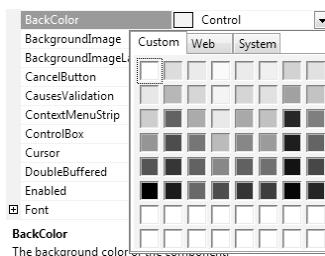


Gambar 2-25. Drop-down menu untuk property BackColor

Warna default yang terpasang saat ini adalah warna yang terpilih, yaitu **Control**. Warna itu ada di dalam tab **System**. Warna-warna **System** adalah warna-warna yang ada dalam setting pada saat Anda mengatur tampilan dari komputer Windows Anda.

Seperti yang ditunjukkan dalam gambar tersebut, Anda bisa memilih warna dari **Active Caption**. **Active Caption** adalah bagian di mana Anda mengubah tampilan text menjadi "**My First Form**" di atas.

Jika Anda tidak mau mengikuti warna dari tab **System**, Anda bisa memilih tab **Custom**, dan akan ditampilkan seperti di bawah ini.



Gambar 2-26. Pilihan untuk tab Custom

Pilihlah sembarang warna dari pilihan warna itu, dan warna dasar dari form Anda akan berubah juga. Di sini dipilih warna yang kebiruan.

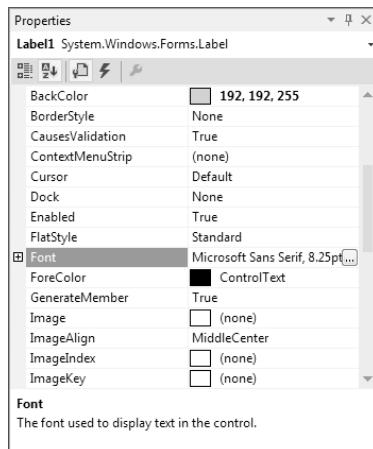
Selain dari tab **System** dan tab **Custom**, Anda bisa juga memilih tab **Web**. Sebuah warna disebut dengan **Web-Safe** adalah warna-warna yang akan ditampilkan dalam sebuah web-browser dengan benar. Anda mungkin perlu untuk memilih satu di antara warna-warna ini jika Anda akan membuat sebuah aplikasi untuk Internet. Namun, pada akhirnya pilihan tetap jatuh pada Anda.

Untuk mengubah warna dari label-label yang ada, pilihlah label yang akan diubah warnanya. Lihatlah pada bagian property-nya, dan carilah bagian **BackColor** seperti yang baru saja Anda lakukan pada object **Form**.

Ubahlah warna dari ketiga label tersebut. Untuk mengubah warna lebih dari sebuah label pada saat yang bersamaan, pilihlah salah satu label terlebih dahulu. Kemudian, tekan dan tahan tombol “**Ctrl**” dari keyboard Anda dan pilih sebuah label yang lain. Anda akan lihat kedua label ini akan terpilih menjadi satu. Masih dengan menahan tombol “**Ctrl**”, tekan lagi label yang ketiga, dan ketiga label tersebut akan terpilih. Kemudian Anda tinggal mengubah property **BackColor** untuk ketiga label tersebut pada saat yang bersamaan.

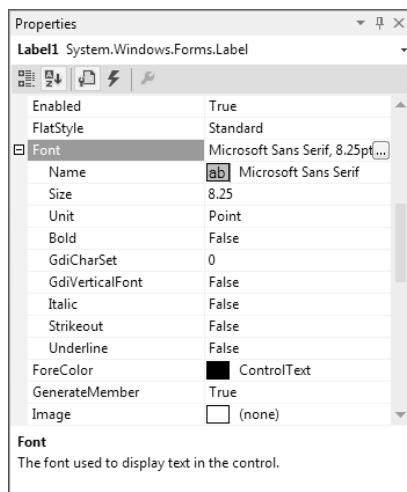
Jika Anda hendak mengubah ukuran dari **Font** pada label Anda, pilihlah salah satu label, misalnya **Label1**.

Carilah dalam daftar property, sampai Anda sampai pada **Font**. Pilihlah property tersebut. Anda bisa melihat bahwa **Font** default yang terpilih saat ini adalah **Microsoft Sans Serif**.



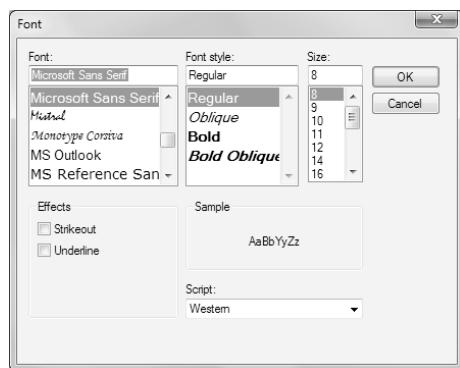
Gambar 2-27. Property **Font** dari **Label1**

Perhatikan bahwa property **Font** tersebut memiliki sebuah tanda “+” di sampingnya. Hal ini menunjukkan bahwa property tersebut bisa dikembangkan. Tekanlah tanda “+” tersebut untuk melihat hal berikut:



Gambar 2-28. Property Font yang dikembangkan

Bisa Anda lihat bahwa ada banyak yang bisa diubah hanya dari property **Font** saja, nama **Font**, ukuran, **Bold** atau biasa, dan sebagainya. Anda juga bisa menekan tombol dengan tiga buah titik di sebelah kanan. Tombol ini akan menampilkan sebuah kotak dialog di mana Anda juga bisa mengubah tampilan **Font**.



Gambar 2-29. Kotak dialog untuk mengganti font

Lakukan pengubahan berikut pada ketiga label yang ada:

Font: Arial

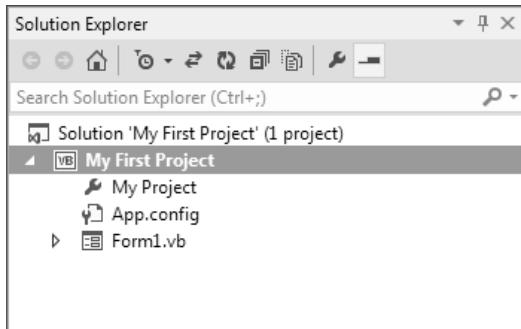
Font Size: 10

Font Style: Bold

Ubahlah Font untuk ketiga textbox juga supaya sama dengan Font pada ketiga label yang ada.

Menyimpan Project Anda

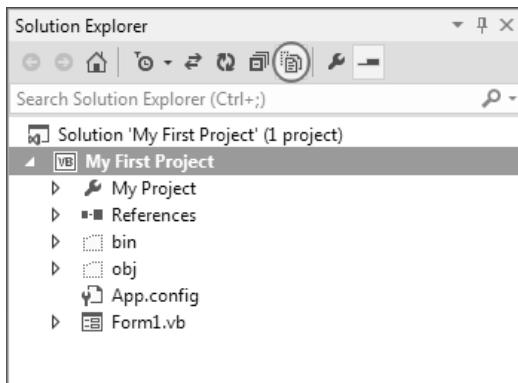
Jika Anda perhatikan, di sebelah kanan atas dari workspace Anda, akan Anda lihat **Solution Explorer** (jika Anda tidak dapat melihatnya, buka lah submenu **View > Solution Explorer**).



Gambar 2-30. Solution Explorer

Solution Explorer ini akan menampilkan file-file yang ada dalam project Anda.

Tidak terlalu banyak ada file di dalam project Anda. Tapi, jika Anda tekan tombol **Show All Files** di bagian toolbar dari **Solution Explorer**, seluruh file yang ada dalam project akan ditampilkan.



Gambar 2-31. Solution Explorer, tampilan lengkap

Jadi, ketika Anda menyimpan project Anda, seluruh file yang ada di sini akan ikut tersimpan.

Untuk menyimpan kerja Anda, pilih submenu **File > Save All**.

Seperti sudah dijelaskan di atas, project Anda akan disimpan dalam sebuah folder di dalam **My Documents**.

Untuk mempercepat proses penyimpanan, terutama pada saat Anda sedang melakukan pemrograman, Anda bisa menggunakan kombinasi tombol **Ctrl+Shift+S** dari keyboard Anda. Atau Anda bisa juga menggunakan tombol yang tampak sebagai dua buah floppy disk pada toolbar Anda. Dengan seringnya melakukan penyimpanan, kemungkinan kehilangan data akan bisa dikurangi.

BAB 3 | Mengenal Database

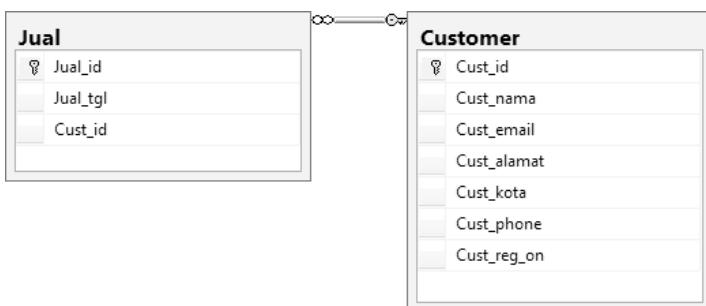
Setelah Anda menyelesaikan proses instalasi, dan mengerti cara penggunaan Visual Studio 2013 Anda, maka sekaranglah saatnya Anda mengerti mengenai database itu sendiri.

Database adalah suatu bagian dalam pemrograman komputer yang sangat luas cakupannya. Sebelum Anda bisa mengerti bagaimana cara untuk membangun suatu sistem database, Anda harus terlebih dahulu mengerti gambaran besarnya dari teknologi data access itu sendiri.

Database dan Database Management Systems

Sebuah Database adalah sebuah system yang digunakan untuk menyimpan informasi terstruktur, di mana informasi tersebut disusun dan disimpan sedemikian sehingga bisa diambil dengan mudah dan efisien.

Informasi tersebut dipecah dalam bentuk table, seperti ditunjukkan dalam gambar berikut.



Gambar 3-1. Contoh table

Setiap table menyimpan data yang berbeda (satu table menyimpan data customer, table lain menyimpan data penjualan, begitu seterusnya). Anda memecah-mecah semua informasi yang tersedia dalam potongan-potongan kecil sehingga Anda dapat mengaturnya dengan mudah. Anda juga bisa membuat aturan-aturan untuk melindungi database yang ada terhadap suatu tindakan tertentu dari user, dan dapat mengatur suatu program untuk memastikan aturan tersebut dijalankan (misalnya, menolak seorang customer tanpa isian nama). Aturan-aturan tersebut bisa Anda tekanan pada isi dari table **Customer**, misalnya, dan aturan tersebut tidak bisa dilaksanakan pada table **Barang**, atau table **Jual**. Begitu pula sebaliknya.

Selain dari table itu sendiri, Anda juga menentukan relationship/ hubungan antar-table. Relationship memungkinkan user untuk menggabungkan informasi dari beberapa table. Seperti yang bisa Anda lihat dalam gambar di atas, table **Jual** dan table **Customer** memiliki hubungan, dan hal yang menghubungkan mereka adalah sebuah Customer ID (**Cust_id**). Karena terjadi sebuah penjualan pada seorang customer, maka setiap catatan penjualan haruslah memiliki kaitan dengan customer tersebut. Dengan menentukan relationship antara kedua table tersebut, Anda bisa dengan cepat mendapatkan catatan penjualan untuk seorang customer. Tanpa relationship tersebut, Anda harus memeriksa semua catatan dalam table **Jual** untuk mencari catatan penjualan untuk seorang customer itu.

Sebuah database yang bergantung pada relationship antara table di dalamnya disebut dengan relational database, dan bisa direpresentasikan dalam bentuk visual diagram, seperti yang ditunjukkan dalam gambar di atas.

Table dan relationship, bersama dengan item-item lain dalam database, menjadikan unsur-unsur pendukung dalam desain sebuah database. Informasi di dalamnya disimpan di dalam dan diambil dari database tersebut oleh suatu program yang dikenal dengan database management system (**DBMS**).

DBMS akan mengatur semua informasi di dalam database, dan aplikasi dapat mengakses informasi tersebut melalui statement yang ditulis dalam *Structured Query Language (SQL)*. Statement semacam itu disebut dengan *query*, di mana terdapat dua macam query: *selection query*, yang akan mengambil informasi dari dalam database, dan *action query*, yang akan melakukan update pada database.

DBMS menyediakan fungsi-fungsi berikut:

- **DBMS** memperbolehkan aplikasi untuk menggunakan statement **SQL** untuk menentukan struktur dari database itu. Elemen dari statement **SQL** yang menentukan atau mengubah struktur ini disebut dengan *Data Definition Language (DDL)*. Jadi, walaupun semua **DBMS** menggunakan antarmuka visual dengan operasi point-and-click di seluruh tampilannya, operasi-operasi tersebut diterjemahkan ke dalam statement **DDL** yang sesuai.
- **DBMS** memperbolehkan aplikasi untuk menggunakan statement **SQL** untuk memanipulasi informasi yang disimpan dalam suatu database. Elemen dari statement **SQL** yang memanipulasi informasi ini disebut dengan *Data Manipulation Language (DML)*. Operasi-operasi dasar dalam **DML** yang akan banyak digunakan adalah: query, update, add, dan delete.
- **DBMS** melindungi integritas dari database dengan menekankan aturan-aturan tertentu, yang digabungkan pada design database itu sendiri. Anda dapat menentukan nilai-nilai awal, melarang nilai kosong pada suatu field, menghalangi penghapusan suatu record yang berhubungan dengan record lain, dan seterusnya. Sebagai contoh, Anda memerintahkan untuk menghapus data seorang customer, di mana customer ini berhubungan dengan satu atau dua buah invoice penjualan. Jika Anda bisa menghapus data customer tersebut, maka data invoice penjualan untuk customer tersebut menjadi ‘orphaned’, atau tidak memiliki elemen di atasnya lagi.

Selain dari itu, **DBMS** juga bertanggung-jawab pada keamanan database (melindungi database dari akses oleh user yang tidak memiliki hak).

SQL Server adalah sebuah database management system, dan bukanlah sebuah database. Sebuah database **SQL Server** adalah sebuah database yang di-maintain oleh **SQL Server**. **SQL** adalah sebuah bahasa universal yang digunakan untuk manipulasi database, dan didukung oleh hampir semua **DBMS**. **SQL** mengambil record-record tertentu dari dalam database, dan mengembalikannya pada user. Selain itu, bahasa tersebut juga bisa digunakan untuk manipulasi record, seperti penambahan, update, maupun penghapusan informasi dari dalam database.

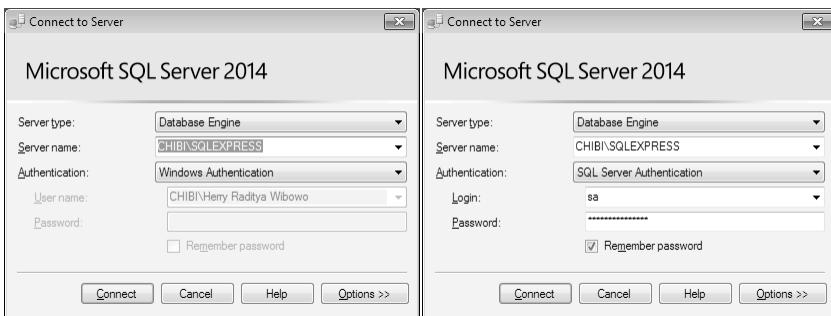
SQL Server Management Studio

SQL Server Management Studio (SSMS) adalah sebuah lingkungan kerja terintegrasi untuk mendapatkan akses, konfigurasi, manajemen, administrasi, maupun membangun seluruh komponen dalam **SQL Server**. Dalam bab ini, **SSMS** akan Anda gunakan untuk membangun suatu database yang akan digunakan dalam seluruh buku ini.

Jalankan **SSMS** dari Start Menu Anda, sampai Anda dihadapkan pada window **Connect to Server**.

Ada dua cara untuk melakukan koneksi, apabila pada saat instalasi sebelumnya, Anda memilih untuk menggunakan metode **Mixed Mode**. Anda bisa menggunakan account **Windows** Anda, dan Anda bisa juga menggunakan account system administrator pada **SQL Server** Anda.

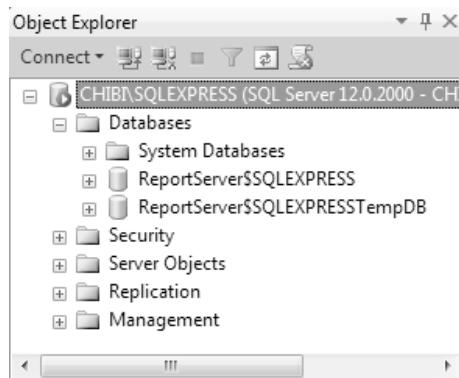
Kedua pilihan tersebut ditunjukkan dalam gambar berikut ini.



Gambar 3-2. Pilihan cara autentikasi

Apabila Anda menggunakan cara **SQL Server Authentication**, masukkan password untuk account **sa** Anda, dan tekan tombol **Connect**. Apabila Anda menggunakan cara **Windows Authentication**, Anda bisa langsung menekan tombol **Connect** tersebut.

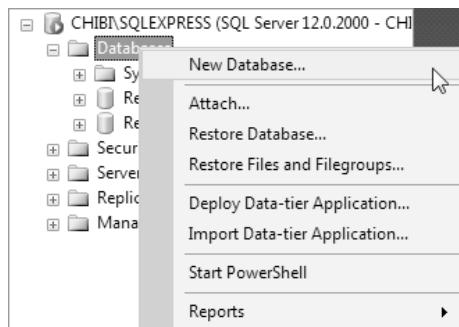
Setelah Anda berhasil masuk, perhatikan tool window **Object Explorer** di sebelah kiri workspace Anda. Di sana akan ditunjukkan semua bagian dari instalasi **SQL Server** Anda, namun terutama, perhatikan pada item **Database**. Lakukan klik pada icon tanda panah di sebelah kata **Database** untuk membukanya.



Gambar 3-3. Object Explorer

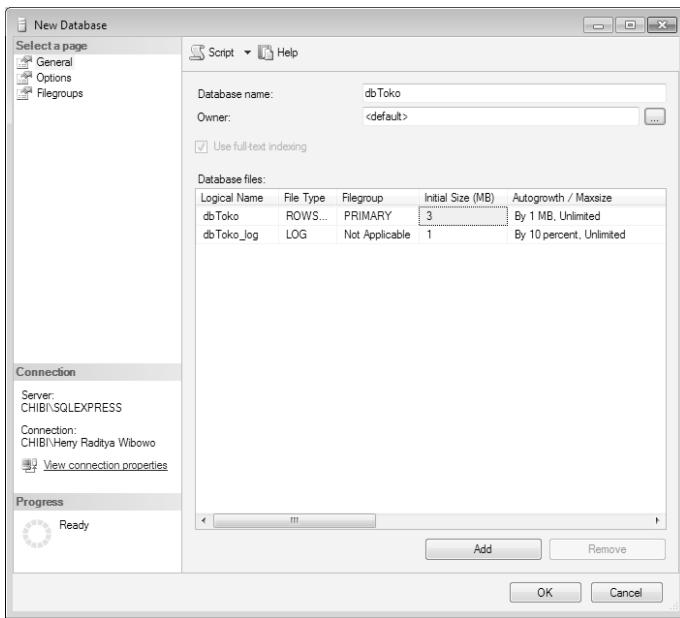
Pada saat instalasi pertama, belum ada database yang ada di sana, selain apa yang sudah ada sebelumnya. Di sini, Anda akan mencoba untuk membuat sebuah database baru, yang akan Anda gunakan dalam contoh pembuatan program dalam buku ini.

Pilihlah Database, dan lakukan klik-kanan.



Gambar 3-4. New Database...

Ada beberapa pilihan di sana, namun kali ini, Anda akan membuat sebuah database. Karena itu, pilihlah **New Database...**



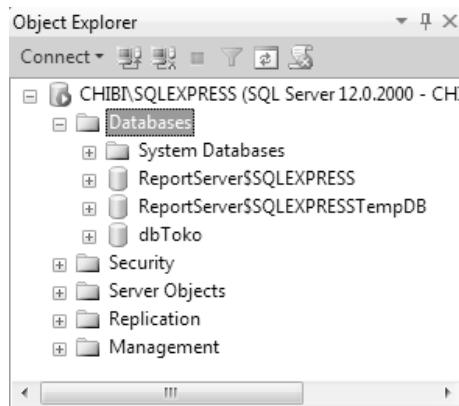
Gambar 3-5. Konfigurasi New Database

Pakailah nilai default di sini, hanya *Database name* yang perlu Anda ganti. Pakailah nama yang cukup sederhana dan mencerminkan apa yang akan Anda simpan di dalamnya. Dalam contoh ini, akan digunakan nama **dbToko** untuk database pertama Anda.

Perhatikan bahwa setelah Anda mengganti nama menjadi **dbToko**, maka untuk *Database files* di bawahnya pun akan ikut berganti nama sesuai dengan nama database Anda.

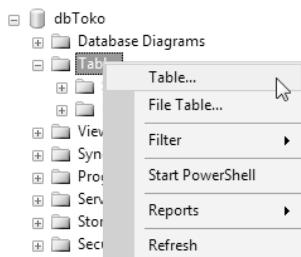
Tekanlah tombol **OK** untuk melanjutkan.

Proses pembuatan database akan dilaksanakan, dan beberapa saat ke mudian, database baru Anda akan nampak pada **Object Explorer**.



Gambar 3-6. dbToko pada Object Explorer

Anda bisa melihat, bahwa database baru Anda ada di sana. Bukalah database **dbToko** tersebut untuk melihat apa yang ada di dalamnya. Setelah itu, lakukan klik-kanan, dan pilihlah untuk membuat sebuah table baru dengan perintah **Table...**



Gambar 3-7. Membuat table baru

Pada tampilan baru yang Anda dapatkan, perhatikan window Properties di sebelah kanan workspace Anda, khususnya pada bagian **(Name)**. Di sini, Anda diperkenankan untuk memberikan sebuah nama untuk table Anda. Gunakan nama yang sesuai dengan peruntukannya, dalam contoh ini, table akan diberi nama dengan **Barang**.

Terimalah semua nilai default pada window Properties itu, dan sekarang beralihlah pada workspace utama Anda.

Anda akan diberikan suatu lembar semacam spreadsheet, namun hanya terdapat tiga buah kolom, yaitu:

- **Column Name**

Di sini Anda tuliskan nama dari field Anda. Nama field adalah seperti nama variable, dan dituliskan sesuai dengan apa yang akan Anda simpan. Misalnya jika sebuah table yang berisi informasi kontak, maka mungkin Anda akan tuliskan Column Name **NamaDepan**, **NamaKeluarga**, **TglLahir**, **Alamat**, **NoTelp**, dan **Email** untuk menyimpan data-data dalam database.

- **Data Type**

Kolom ini menentukan tipe dari informasi yang akan Anda simpan dalam setiap kolom. Misalnya, **NamaDepan** seseorang adalah berupa suatu rentetan huruf, maka Anda mungkin akan menggunakan varchar(30). Demikian juga untuk **TglLahir**, Anda boleh menggunakan tipe date.

- **Allow Nulls**

Kolom ini hanya berupa sebuah checkbox. Jika Anda mengaktifkannya, database akan mengizinkan Anda untuk menyimpan nilai kosong (null) untuk kolom tersebut.

Selesaikanlah pengisian atribut untuk table pertama Anda sesuai dengan gambar berikut.

Column Name	Data Type	Allow Nulls
Brg_id	nchar(10)	<input type="checkbox"/>
Brg_nama	nchar(35)	<input type="checkbox"/>
Brg_unit	nchar(5)	<input type="checkbox"/>
Brg_barcode	nchar(8)	<input type="checkbox"/>
Brg_hpp	int	<input type="checkbox"/>
► Kat_id	tinyint	<input type="checkbox"/>

Gambar 3-9. Atribut untuk table Barang

Untuk lebih Anda ketahui, dalam SQL Server ada enam kategori tipe data yang bisa digunakan, yaitu:

- **Exact Numeric** (angka presisi)

Tipe ini menyimpan nilai numerik di mana Anda mau menentukan presisi dari variable tersebut. Angka di sini bisa berupa angka bulat ataupun angka desimal.

- int
- bigint
- smallint
- tinyint
- decimal dan numeric
- money
- smallmoney

- **Approximate Numeric**

Tipe data ini kurang presisi dibanding dengan sebelumnya. Anda bisa menentukan berapa digit yang akan Anda simpan secara presisi, sementara sisa dari nilainya bisa mengalami pembulatan.

- float
- real

- **Date dan time**

Tipe ini memperbolehkan penyimpanan tanggal dan waktu.

- datetime
- datetime2
- smalldatetime
- date
- time
- datetimeoffset
- timestamp

- **Character string**

Untuk menyimpan nilai yang berdasar pada text.

- char(n)
- nchar(n)
- varchar(n)
- nvarchar(n)

- varchar(max)
 - nvarchar(max)
 - text dan ntext
- **Binary**

Tipe data yang disimpan adalah data biner, yaitu angka yang hanya terdiri dari 0 dan 1.

 - bit
 - binary(n)
 - varbinary(n)
 - varbinary(max)
 - image
 - **Tipe data lain**

Di sini digunakan untuk penyimpanan tipe yang lain, seperti identifier unik, cursor, table maupun data XML.

 - cursor
 - sql_variant
 - table
 - xml
 - uniqueidentifier

Untuk lebih jelasnya, tipe-tipe data di atas akan lebih dijabarkan pada bagian akhir buku ini.

Pada akhirnya, buatlah ke-tujuh table seperti detail di bawah ini:

Table Customer**Gambar 3-10**

Column Name	Data Type	Allow Nulls
Cust_id	smallint	<input type="checkbox"/>
Cust_nama	nchar(25)	<input type="checkbox"/>
Cust_email	nchar(25)	<input type="checkbox"/>
Cust_alamat	nchar(50)	<input type="checkbox"/>
Cust_kota	nchar(20)	<input type="checkbox"/>
Cust_phone	nchar(15)	<input type="checkbox"/>
Cust_reg_on	smalldatetime	<input type="checkbox"/>
		<input type="checkbox"/>
		<input type="checkbox"/>

Table Users**Gambar 3-11**

Column Name	Data Type	Allow Nulls
User_id	smallint	<input type="checkbox"/>
User_nama	nchar(8)	<input type="checkbox"/>
User_priv	tinyint	<input type="checkbox"/>
User_pwd	nchar(8)	<input type="checkbox"/>
		<input type="checkbox"/>
		<input type="checkbox"/>

Table Kategori**Gambar 3-12**

Column Name	Data Type	Allow Nulls
Kat_id	tinyint	<input type="checkbox"/>
Kat_nama	nchar(10)	<input type="checkbox"/>
		<input type="checkbox"/>

Table Jual**Gambar 3-13**

Column Name	Data Type	Allow Nulls
Jual_id	nchar(10)	<input type="checkbox"/>
Jual_tgl	smalldatetime	<input type="checkbox"/>
Cust_id	smallint	<input type="checkbox"/>
Jual_note	nchar(30)	<input checked="" type="checkbox"/>
User_id	smallint	<input type="checkbox"/>
		<input type="checkbox"/>
		<input type="checkbox"/>

Table Jual_details

Gambar 3-14

The screenshot shows the 'Jual_details' table structure in SQL Server Management Studio. The table has five columns: Jual_det_id (smallint), Jual_id (nchar(10)), Brg_id (nchar(10)), Jual_det_qty (tinyint), and Jual_det_harga (int). All columns allow null values.

Column Name	Data Type	Allow Nulls
Jual_det_id	smallint	<input type="checkbox"/>
Jual_id	nchar(10)	<input type="checkbox"/>
Brg_id	nchar(10)	<input type="checkbox"/>
Jual_det_qty	tinyint	<input type="checkbox"/>
Jual_det_harga	int	<input type="checkbox"/>

Table Barang

Gambar 3-15

The screenshot shows the 'Barang' table structure in SQL Server Management Studio. The table has six columns: Brg_id (nchar(10)), Brg_nama (nchar(35)), Brg_unit (nchar(5)), Brg_barcode (nchar(8)), Brg_hpp (int), and Kat_id (tinyint). All columns allow null values.

Column Name	Data Type	Allow Nulls
Brg_id	nchar(10)	<input type="checkbox"/>
Brg_nama	nchar(35)	<input type="checkbox"/>
Brg_unit	nchar(5)	<input type="checkbox"/>
Brg_barcode	nchar(8)	<input type="checkbox"/>
Brg_hpp	int	<input type="checkbox"/>
Kat_id	tinyint	<input type="checkbox"/>

Table Barang_details

Gambar 3-16

The screenshot shows the 'Barang_details' table structure in SQL Server Management Studio. The table has two columns: Brg_id (nchar(10)) and Brg_det_stock (tinyint). The Brg_id column is highlighted with a red border, indicating it is the primary key.

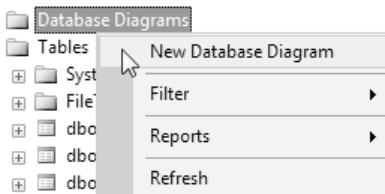
Column Name	Data Type	Allow Nulls
Brg_id	nchar(10)	<input type="checkbox"/>
Brg_det_stock	tinyint	<input type="checkbox"/>

Perhatikan tanda yang berupa anak kunci pada setiap table yang ada. Tanda itu berarti bahwa field di mana tanda anak kunci tersebut berada merupakan **Primary Key**. Sebagai sebuah **Primary Key**, maka nilai yang ada pada field tersebut haruslah unik untuk keseluruhan table. Tidak boleh ada nilai yang kosong, dan juga tidak boleh ada nilai yang sama.

Keberadaan **Primary Key** akan sangat membantu dalam hal pencarian, khususnya pada saat jumlah record dalam table tersebut terhitung banyak. Namun, selain itu, **Primary Key** juga digunakan untuk hubungan antar-table. Antar-table dapat saling berhubungan (seperti contoh pada awal bab, hubungan antara table **Jual** dan table **Customer**), selama pada kedua table tersebut ada sebuah field yang sama.

Langkah selanjutnya dari proses pembuatan database ini adalah menetapkan hubungan antar-table. Untuk mempermudah penetapan ini, Anda bisa membuat sebuah **Database Diagram**.

Temukan folder **Database Diagram** pada window **Object Explorer**, lakukan klik-kanan pada mouse Anda, dan pilihlah **New Database Diagram**.



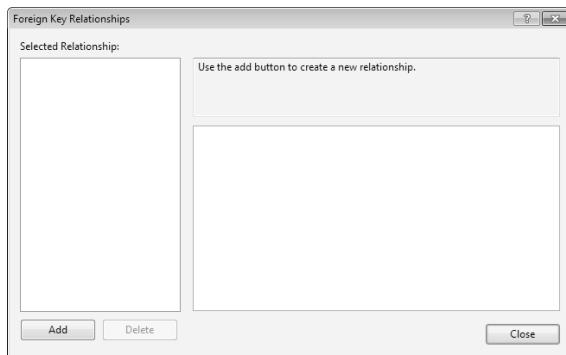
Gambar 3-17. New Database Diagram

Akan terbuka sebuah workspace baru, dengan sebuah kotak dialog **Add Table** ditampilkan. Pilihlah semua table Anda, dan tekan tombol **Add**. Kemudian tekan tombol **Close**.

Akan tersusun sebuah diagram dasar, dengan semua table Anda di dalamnya. Aturlah penempatannya seperti yang diinginkan, dan atur juga ukuran display, supaya lebih mudah untuk dilihat.

Yang paling mudah untuk dibuat hubungannya adalah table-table yang berupa parent-child, yaitu seperti **Barang** dengan **Barang_details**, dan **Jual** dengan **Jual_details**.

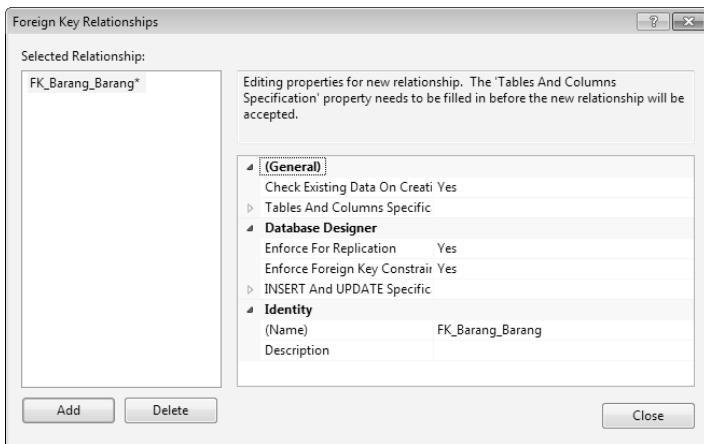
Pilihlah table **Barang**, tekan tombol mouse sebelah kanan, dan pilihlah opsi Relationships.



Gambar 3-19. Foreign Key Relationships

Apa itu **Foreign Key**? Apakah hubungannya dengan **Primary Key**? Sebuah **Foreign Key** adalah suatu field dalam sebuah table yang sama dengan **Primary Key** pada table lain. Jadi, hubungan antar-table adalah sama dengan hubungan antara **Primary Key** suatu table dengan **Foreign Key** pada table lain.

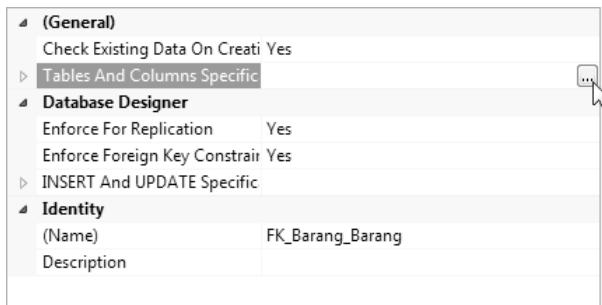
Saat ini, relationships masih kosong, karena memang belum dibuat. Tekan tombol **Add** untuk menambahkan suatu relationship baru.



Gambar 3-20. Penambahan suatu relationship

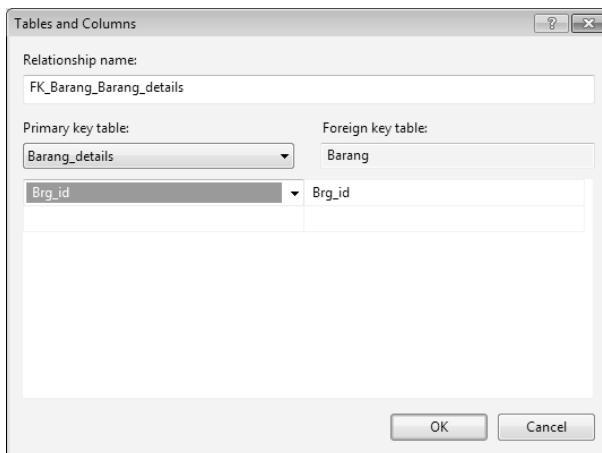
Secara default, system akan membuat sebuah relationship yang mengacu pada dirinya sendiri. Seperti yang Anda bisa lihat pada bagian sebelah kiri, kunci relationship tersebut mendapat nama temporary **FK_Barang_Barang**, yang artinya, sebuah **Foreign Key** untuk table **Barang** dengan table **Barang** juga. Namun di sini Anda hendak membuat sebuah relationship antara table **Barang** dengan table **Barang_details**.

Perhatikan pada bagian sebelah kanan. Pilihlah baris pada bagian **General** yang bertuliskan **Tables and Columns Specific**. Setelah Anda pilih baris tersebut, pada bagian sebelah kanannya akan ditampilkan sebuah tombol kecil, dengan isi tiga buah titik.



Gambar 3-21. *Tables and Columns Specific*

Tekanlah tombol tersebut. Anda akan mendapatkan sebuah *form* baru.



Gambar 3-22. *Hubungan antara table Barang dengan Barang_details*

Dalam form ini hanya bagian sebelah kiri yang bisa Anda ubah nama table-nya. Untuk kunci relationship Anda yang pertama, gantilah menjadi table **Barang_details**, dan kemudian ubah juga kunci field yang digunakan, untuk mengacu pada field **Brg_id**.

Tekan **OK**, dan selesailah relationship Anda yang pertama. Sebagai tanda bahwa relationship Anda sudah terbentuk, pada diagram akan ditampilkan sebuah garis penghubung antara table **Barang** dengan table **Barang_details**.



Gambar 3-23. Relationship yang pertama

Buatlah semua relationship yang ada, dengan acuan daftar berikut:

Barang – Barang_details, Brg_id

Barang – Kategori, Kat_id

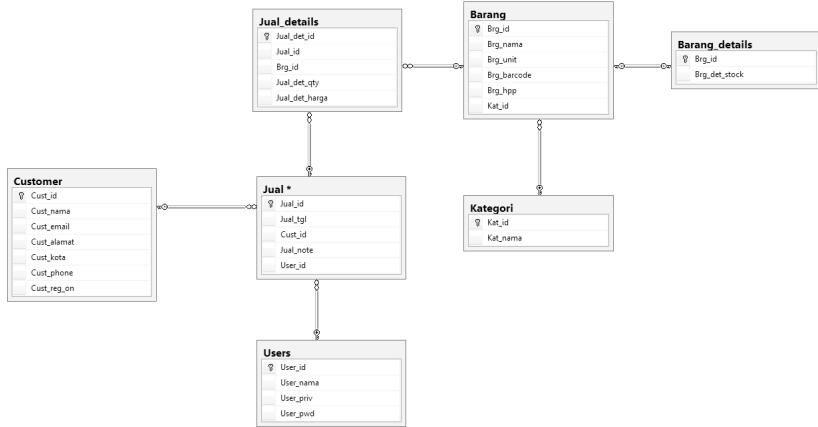
Barang – Jual_details, Brg_id

Jual – Jual_details, Jual_id

Jual – Customer, Cust_id

Jual – Users, User_id

Setelah dibuat semua, gambar pada diagram tampak seperti berikut:



Gambar 3-24. Relationship secara penuh

Seperti bisa dilihat pada gambar, Anda bisa menetapkan lebih dari satu hubungan untuk sebuah table.

Simpanlah diagram tersebut, dan keluarlah dari **SSMS**.

SQL Server 2014 Express Edition menawarkan kesempatan pada Anda untuk mendalami pengetahuan mengenai database maupun aplikasi database, sementara masih berupa sebuah aplikasi yang gratis.

Dalam bab ini, Anda akan mempelajari mengenai struktur dasar dari query maupun statement **SQL**. Perintah-perintah yang digunakan adalah merupakan bagian dari spesifikasi bahasa **Transact-SQL (T-SQL)**, dan merupakan unsur utama dalam penggunaan **Microsoft SQL Server**.

Meskipun ada banyak tool yang tersedia dalam merancang query secara visual, seperti misalnya **Visual Database Tool** yang tersedia dari **Microsoft Visual Studio** sendiri, namun adalah tetap penting untuk mengerti dan memahami bahasa **SQL** itu sendiri. Akan ada waktu di mana menulis sendiri statement **SQL** yang diperlukan adalah cara satu-satunya, ataupun yang paling cepat, untuk mendapatkan apa yang Anda butuhkan. Mempelajari **SQL** juga adalah suatu cara yang ideal untuk memahami penggunaan sebuah relational database seperti **SQL Express**.

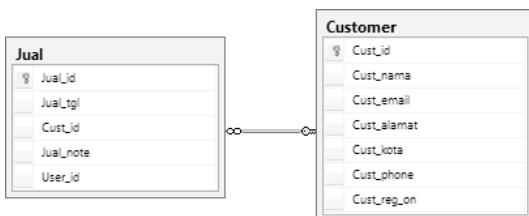
Relational Database

Seperti yang telah diperkenalkan pada bab sebelumnya, sebuah relational database terdiri dari satu atau lebih table, di mana setiap table memiliki 0 atau lebih record, atau baris, data. Data untuk setiap barisnya diatur dalam unit tertentu, yang disebut sebagai *field* atau kolom (*column*). Ketika kita hendak menunjukkan field dalam sebuah table, misalnya table **Customer**, biasanya akan ditunjukkan seperti gambar berikut.

Customer	
•	Cust_id
•	Cust_nama
•	Cust_email
•	Cust_alamat
•	Cust_kota
•	Cust_phone
•	Cust_reg_on

Gambar 4-1. Table Customer

Beberapa dari table-table yang ada dalam sebuah database akan memiliki relationship, atau hubungan, di antaranya. Hubungan itu bisa berupa hubungan *one-to-one* atau *one-to-many*. Elemen yang menghubungkan antar-table dimungkinkan oleh sebuah pasangan **Primary Key** dan **Foreign Key**, di mana sebuah field **Foreign Key** dari sebuah table adalah merupakan **Primary Key** dari table yang lain.



Gambar 4-2. Hubungan antar-table

Suatu contoh dari hubungan *one-to-many* adalah antara table **Customer** dan table **Jual**. Kedua buah table tersebut memiliki field **Cust_id**. Field **Cust_id** tersebut merupakan **Primary Key** dari table **Customer**, dan sekaligus merupakan **Foreign Key** dari table **Jual**. Kedua field yang berhubungan tersebut tidak perlu memiliki nama yang sama antar-keduanya, namun hal tersebut merupakan praktek yang dianjurkan.

Mengambil Data: Query SQL SELECT

Adalah sangat jarang apabila sebuah aplikasi database tidak menggunakan proses mengambil maupun menampilkan data. Sekali Anda sudah memiliki data dalam database, Anda akan menggunakan其nya dalam cara apa pun ke dalam aplikasi Anda. Anda akan perlu untuk melihat data tersebut dan menganalisisnya dalam cara-cara yang berbeda-beda, dengan cara mengubah-ubah variable filternya, pengurutannya, maupun

adanya perhitungan-perhitungan yang diaplikasikan pada data men-tahnya. Statement **SQL SELECT** merupakan perintah yang akan Anda gunakan untuk memilih, atau select, data yang Anda butuhkan dari dalam database ke aplikasi Anda.

Sebuah statement **SQL SELECT** dapat dipecah dalam beberapa elemen, di mana setiap elemen diawali dengan sebuah keyword. Meskipun bukan merupakan suatu keharusan, namun konvensi yang biasa diterapkan dalam penulisan keyword ini adalah dengan menggunakan huruf kapital.

Dalam awal bab ini, Anda akan mempelajari pada elemen yang paling banyak digunakan dari sebuah statement **SELECT**, yaitu:

SELECT
FROM
WHERE
ORDER BY

Clause SELECT ... FROM

Sebuah statement **SELECT** yang paling sederhana hanya memiliki dua bagian: (1) field apa yang akan Anda tampilkan dan (2) dari table apa field tersebut berasal.

Jika Anda ingin mendapatkan semua informasi dari semua customer dari dalam table **Customer**, maka Anda bisa menggunakan tanda **asterisk (*)** sebagai sebuah shortcut untuk semua field, dan query Anda akan terlihat seperti

```
SELECT * FROM Customer
```

Jika Anda hanya ingin mendapatkan beberapa field saja, Anda bisa secara eksplisit menuliskannya sebagai sebuah daftar yang dipisahkan dengan tanda koma, seperti

```
SELECT Cust_nama, Cust_email, Cust_reg_on, Cust_kota  
FROM Customer
```

yang akan memberikan data berisi field-field yang Anda tuliskan dalam perintah query:

	Cust_nama	Cust_email	Cust_reg_on	Cust_kota
1	Cash	-	2008-12-27 00:00:00	-
2	Satu Persatu	satu_p@email.com	2013-04-14 00:00:00	Batam
3	Dua Perdua	dua_p@email.com	2013-04-22 00:00:00	Bintan
4	Tiga Pertiga	tiga_p@email.com	2013-05-02 00:00:00	Batam
5	Empat Perempat	empat_p@email.com	2013-08-22 00:00:00	Dumai
6	Lima Perlama	lima_p@email.com	2013-12-03 00:00:00	Batam

Gambar 4-3. Hasil dari statement query

Menuliskan field yang dibutuhkan secara eksplisit juga akan mengizinkan Anda untuk mengatur pengurutan dari field yang akan diambil, sehingga jika misalnya Anda menginginkan data **Cust_reg_on** ditampilkan terlebih dulu dibanding semua field lainnya, Anda bisa menuliskan

```
SELECT Cust_reg_on, Cust_nama, Cust_email,Cust_kota
FROM Customer
```

Clause WHERE

Hal lain yang perlu dipelajari selanjutnya adalah membatasi, atau menggunakan filter, pada data yang Anda dapatkan dari database. Dengan menambahkan sebuah clause **WHERE** pada statement **SELECT**, berarti Anda menambahkan sebuah kondisi yang harus dipenuhi oleh data-data yang terpilih. Hal ini akan membatasi jumlah baris data yang merupakan jawaban dari query yang ditanyakan.

Anda bisa melanjutkan dari query sebelumnya, dan membatasi hasilnya untuk hanya menunjukkan customer yang tinggal di **Batam** saja

```
SELECT Cust_reg_on, Cust_nama, Cust_email,Cust_kota
FROM Customer
WHERE Kota = 'Batam'
```

yang akan memberikan hasil:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2013-04-14 00:00:00	Satu Persatu	satu_p@email.com	Batam
2	2013-05-02 00:00:00	Tiga Pertiga	tiga_p@email.com	Batam
3	2013-12-03 00:00:00	Lima Perlama	lima_p@email.com	Batam

Gambar 4-4. Hasil setelah diberikan filter

Jika yang Anda inginkan adalah kebalikannya, customer yang tidak tinggal di **Batam**, Anda akan menuliskan

```
SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota  
FROM Customer  
WHERE Kota <> 'Batam'
```

Tidaklah diperlukan untuk memastikan kesamaan tipe data; Anda bisa juga menggunakan operator sama dengan (atau tidak sama dengan) seperti yang Anda inginkan secara langsung. Misalnya, Anda hendak menampilkan semua customer yang terdaftar pada atau setelah tanggal tertentu, di mana Anda akan menuliskan

```
SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota  
FROM Customer  
WHERE Cust_reg_on >= '2-May-2013'
```

dan mendapatkan hasil sebagai berikut:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2013-05-02 00:00:00	Tiga Pertiga	tiga_p@email.com	Batam
2	2013-08-22 00:00:00	Empat Perempat	empat_p@email.com	Dumai
3	2013-12-03 00:00:00	Lima Perlina	lima_p@email.com	Batam

Gambar 4-5. Penggunaan filter

Tentu saja Anda bisa menuliskan kondisi yang lebih rumit. Cara yang paling jelas untuk melakukan itu adalah dengan menggunakan kondisi yang lebih dari satu pada clause **WHERE**. Jika Anda ingin mengetahui customer mana yang diregistrasi setelah tanggal tertentu, dan hanya yang tinggal di **Batam** saja, maka Anda bisa menuliskan

```
SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota  
FROM Customer  
WHERE Cust_reg_on >= ('2-May-2013') AND (cust_kota = 'Batam')
```

yang akan menghasilkan:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2013-05-02 00:00:00	Tiga Pertiga	tiga_p@email.com	Batam
2	2013-12-03 00:00:00	Lima Perlina	lima_p@email.com	Batam

Gambar 4-6. Kondisi yang lebih dari satu

Perhatikan juga bahwa **SQL** memiliki operator spesial **BETWEEN** yang akan menguji apakah nilai berada di antara dua buah nilai lain. Hal ini memperbolehkan Anda untuk menulis query lain sebagai

```

SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota
FROM Customer
WHERE Cust_reg_on BETWEEN '01-May-2013' AND '31-August-2013'
dengan hasil:

```

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2013-05-02 00:00:00	Tiga Pertiga	tiga_p@email.com	Batam
2	2013-08-22 00:00:00	Empat Perempat	empat_p@email.com	Dumai

Gambar 4-7. Operator BETWEEN

Anda juga bisa menambahkan operator **NOT**, untuk mendapatkan baris data yang tidak berada di antara dua tanggal:

```

SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota
FROM Customer
WHERE Cust_reg_on NOT BETWEEN '01-May-2013' AND '31-August-2013'

```

dengan hasil:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2008-12-27 00:00:00	Cash	-	-
2	2013-04-14 00:00:00	Satu Persatu	satu_p@email.com	Batam
3	2013-04-22 00:00:00	Dua Perdua	dua_p@email.com	Bintan
4	2013-12-03 00:00:00	Lima Perlima	lima_p@email.com	Batam

Gambar 4-8. Operator NOT BETWEEN

Selanjutnya, bagaimana jika Anda hendak menguji apakah nilai dari sebuah field sama untuk beberapa buah nilai lain? Anda bisa menggunakan operator **OR** untuk menggabungkan kedua kondisi, seperti

```

SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota
FROM Customer
WHERE (Cust_kota = 'Bintan') OR (Cust_kota = 'Dumai')

```

Namun, jika pada data terdapat banyak kondisi nilai yang ingin Anda bandingkan, Anda bisa menggunakan operator **IN** untuk menguji terhadap suatu kumpulan nilai. Misalnya, query di atas dapat ditulis ulang sebagai berikut:

```

SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota
FROM Customer
WHERE Cust_kota IN ('Bintan', 'Dumai')

```

yang akan menghasilkan baris data sebagai berikut:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2013-04-22 00:00:00	Dua Perdua	dua_p@email.com	Bintan
2	2013-08-22 00:00:00	Empat Perempat	empat_p@email.com	Dumai

Gambar 4-9. Operator IN

Dan seperti pada operator **BETWEEN**, di sini Anda juga akan mendapatkan lawan dari hasilnya, dengan melakukan query untuk kota yang tidak terdapat pada kumpulan kota:

```
SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota
FROM Customer
WHERE Cust_kota NOT IN ('Bintan', 'Dumai')
```

Akhirnya, operator **LIKE** mengizinkan Anda untuk melakukan suatu pengenalan pola dasar dengan menggunakan karakter *wildcard*. Untuk **Microsoft SQL Server**, karakter wildcard yang bisa digunakan adalah sebagai berikut:

Wildcard	Keterangan
_ (garis bawah)	sama dengan karakter tunggal.
%	sama dengan urutan satu atau lebih karakter.
[]	sama dengan karakter tunggal dalam range ([a-f] atau kumpulan ([abcdef]).
[^]	sama dengan karakter tunggal yang tidak dalam range ([^a-f]) atau kumpulan ([^abcdef]).

Beberapa contoh akan dapat menjelaskan aturan-aturan tersebut.

WHERE Cust_nama LIKE 'A_i' akan mendapatkan semua nama yang hanya terdiri dari tiga karakter, yang diawali dengan huruf 'A' dan diakhiri dengan huruf 'i' (misal **Ani**, **Aji**).

WHERE Cust_kota LIKE 'B%' akan mendapatkan semua customer yang tinggal di kota yang diawali dengan huruf 'B'.

WHERE Cust_nama LIKE 'A[jn]i' akan mendapatkan semua customer yang namanya diawali dengan huruf 'A' dan diakhiri dengan huruf 'i', namun huruf ditengahnya hanya yang berupa huruf 'j' dan 'n' saja.

WHERE Cust_nama LIKE 'A[^j]%' akan mendapatkan semua nama yang diawali dengan huruf 'A' di mana huruf keduanya bukan 'j'.

Di sini Anda juga masih bisa untuk menempatkan operator **NOT**; misalnya Anda hendak mendapatkan data customer yang namanya tidak diawali dengan huruf 'S' atau 'D', maka Anda bisa menuliskan

```
SELECT Cust_reg_on, Cust_nama, Cust_email, Cust_kota  
FROM Customer  
WHERE (Cust_nama NOT LIKE 'S%') AND (Cust_nama NOT LIKE 'D%')
```

yang akan menghasilkan baris data:

	Cust_reg_on	Cust_nama	Cust_email	Cust_kota
1	2008-12-27 00:00:00	Cash	-	-
2	2013-05-02 00:00:00	Tiga Pertiga	tiga_p@email.com	Batam
3	2013-08-22 00:00:00	Empat Perempat	empat_p@email.com	Dumai
4	2013-12-03 00:00:00	Lima Perlama	lima_p@email.com	Batam

Gambar 4-10. Penggunaan operator **NOT** pada operator **LIKE**

Clause ORDER BY

Sampai saat ini, Anda mempelajari cara untuk melakukan seleksi pada data; yaitu menentukan kondisi-kondisi untuk memilih baris data yang mana yang akan dimasukkan dalam baris data akhir untuk diambil dari database. Sekali Anda sudah menentukan field dan baris yang akan dimasukkan sebagai hasil dari query **SELECT** Anda, Anda mungkin mau mengatur urutan baris data yang akan ditampilkan.

Untuk mengurutkan baris data, Anda menggunakan clause **ORDER BY**. Clause **ORDER BY** ini memerlukan sebuah nama field yang akan digunakan untuk pengurutannya. Jika sekarang Anda kembali pada statement **SELECT** Anda yang pertama, Anda bisa mengurutkan hasilnya dengan kunci pada nama **Kota** dengan statement berikut:

```
SELECT Cust_nama, Cust_email, Cust_reg_on, Cust_kota  
FROM Customer  
ORDERBY Cust_kota
```

Secara default, pengurutan terhadap sebuah field adalah bersifat *ascending* (dari nilai terendah sampai nilai tertinggi), seperti ditunjukkan dalam hasil berikut:

	Cust_nama	Cust_email	Cust_reg_on	Cust_kota
1	Cash	-	2008-12-27 00:00:00	-
2	Satu Persatu	satu_p@email.com	2013-04-14 00:00:00	Batam
3	Tiga Pertiga	tiga_p@email.com	2013-05-02 00:00:00	Batam
4	Lima Perlama	lima_p@email.com	2013-12-03 00:00:00	Batam
5	Dua Perdua	dua_p@email.com	2013-04-22 00:00:00	Bintan
6	Empat Perempat	empat_p@email.com	2013-08-22 00:00:00	Dumai

Gambar 4-11. Hasil pengurutan terhadap Kota

Namun jika Anda menghendaki pengurutan yang *descending*, Anda bisa juga menambahkan keyword **DESC** setelah nama field tersebut.

Clause **ORDER BY** tidak dibatasi hanya pada sebuah field tunggal. Anda bisa menggunakan sebuah kumpulan field yang dibatasi dengan tanda koma, di mana baris data akan diurutkan berdasar field pertama yang diberikan pada statement, dan kemudian pada field berikut dalam statement. Jadi jika misalnya Anda hendak menambahkan field **Nama** customer pada statement **SELECT** tersebut, di mana Anda hendak mengurutkan berdasar pada **Kota** dan **Nama**, maka Anda bisa menuliskannya:

```
SELECT Cust_nama, Cust_email, Cust_reg_on, Cust_kota
FROM Customer
ORDERBY Cust_kota, Cust_namDESC
```

Jika Anda ingin membedakan pengurutan pada kata kuncinya, maka Anda bisa juga menuliskannya sebagai berikut:

```
SELECT Cust_nama, Cust_email, Cust_reg_on, Cust_kota
FROM Customer
ORDERBY Cust_nama ASC, Cust_kota DESC
```

Hasil dari query tersebut adalah sebagai berikut:

	Cust_nama	Cust_email	Cust_reg_on	Cust_kota
1	Cash	-	2008-12-27 00:00:00	-
2	Dua Perdua	dua_p@email.com	2013-04-22 00:00:00	Bintan
3	Empat Perempat	empat_p@email.com	2013-08-22 00:00:00	Dumai
4	Lima Perlama	lima_p@email.com	2013-12-03 00:00:00	Batam
5	Satu Persatu	satu_p@email.com	2013-04-14 00:00:00	Batam
6	Tiga Pertiga	tiga_p@email.com	2013-05-02 00:00:00	Batam

Gambar 4-12. Kunci pengurutan yang berbeda

Perhatikan bahwa sebuah field tidak perlu dimasukkan dalam hasil akhir baris data apabila Anda hendak menggunakankannya dalam clause **ORDER**

BY. Jadi jika Anda tidak perlu untuk melihat data **Kota**, tapi perlu untuk mengurutkan semua data berdasar field tersebut, maka Anda bisa menuliskan query Anda sebagai

```
SELECT Cust_nama, Cust_email, Cust_reg_on  
FROM Customer  
ORDERBY Cust_nama ASC, Cust_kota DESC
```

Di mana urutan hasil yang ditunjukkan adalah sama seperti pada query sebelumnya:

	Cust_nama	Cust_email	Cust_reg_on
1	Cash	-	2008-12-27 00:00:00
2	Dua Perdua	dua_p@email.com	2013-04-22 00:00:00
3	Empat Perempat	empat_p@email.com	2013-08-22 00:00:00
4	Lima Perlama	lima_p@email.com	2013-12-03 00:00:00
5	Satu Persatu	satu_p@email.com	2013-04-14 00:00:00
6	Tiga Pertiga	tiga_p@email.com	2013-05-02 00:00:00

Gambar 4-13. Pengurutan tanpa menampilkan field kunci

Penggunaan SQL pada Aplikasi

Dalam aplikasi yang akan Anda coba buat nantinya, penggunaan **SQL** tidak akan hanya terbatas pada statement **SELECT** saja. Ada saatnya Anda memerlukan **SQL** pada saat inisialisasi awal, seperti pembuatan table. Anda juga pasti membutuhkan suatu cara untuk menyimpan informasi yang Anda masukkan dalam aplikasi ke dalam database Anda.

Dalam aplikasi, nantinya akan ada form-form yang menampilkan data apa-adanya. Di sini, dimaksudkan adalah semua data pada table yang dimaksud ditampilkan semua. Untuk itu, statement yang dipakai masih tetap menggunakan statement **SELECT**.

Seerti pada form **Customer**, maka statement yang dibutuhkan adalah

```
SELECT * FROM Customer
```

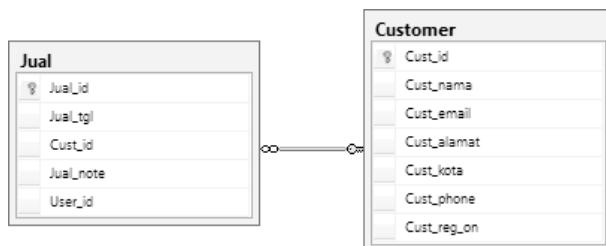
Sedangkan untuk form **Login**, dibutuhkan suatu statement **SELECT** dengan dua buah kondisi, yaitu bahwa **Nama** user ada di dalam table, dan pasangan **Password**-nya adalah benar. Untuk itu, digunakan sebuah statement **SELECT** dengan format:

```
SELECT*FROM Users  
WHERE User_nama = '<nama_user>' AND User_pwd = '<pwd_user>'
```

Statement ini akan mencari dari seluruh record yang ada dalam table **Users**, untuk sebuah record yang memenuhi dua kondisi, yaitu field **User_nama** dan field **User_pwd** sama dengan nilai yang diinputkan dalam form.

Mengambil Data dari Dua buah Table

Statement **SELECT** bisa digunakan untuk mengambil data dari lebih dari satu buah table. Seperti misalnya data penjualan, di mana dalam sebuah table **Jual**, ada hubungan ke table **Customer**, seperti ditunjukkan dalam gambar berikut.



Gambar 4-14. Hubungan antara dua buah table

Jadi, dari dalam table **Jual**, selain ada field **Jual_id** sebagai kunci (**Primary Key**), juga ada sebuah field lain, yaitu field **Cust_id**. Field ini disebut sebagai **Foreign Key** pada table **Jual**, sedangkan pada table **Customer**, field **Cust_id** tersebut adalah **Primary Key**.

Jadi, jika Anda tuliskan statement **SELECT** berikut:

```
SELECT Jual_id, Jual_tgl, Cust_id  
FROM Jual
```

Maka, hasil yang akan ditunjukkan adalah:

	Jual_id	Jual_tgl	Cust_id
1	0001.05.14	2014-01-27 00:00:00	3
2	0002.05.14	2014-02-03 00:00:00	1
3	0003.05.14	2014-05-20 00:00:00	1

Gambar 4-15. Hasil dari query table Jual

Sedangkan apabila Anda menampilkan data dari table **Customer**, maka hasil yang didapatkan adalah:

	Cust_id	Cust_nama
1	1	Cash
2	2	Satu Persatu
3	3	Dua Perdua
4	4	Tiga Pertiga
5	5	Empat Perempat
6	6	Lima Perlima

Gambar 4-16. Hasil dari query table Customer

Jadi, bagaimana caranya untuk menampilkan nama customer pada query table **Jual**? Di sini Anda akan belajar cara untuk melakukan **JOIN** pada kedua field pada masing-masing table.

Yang pertama harus Anda ketahui adalah bahwa Anda bisa meletakkan lebih dari satu buah table setelah clause **FROM**, dengan membatasi setiap nama table dengan tanda koma.

Dan karena jumlah table adalah lebih dari satu, maka penulisan nama field setelah clause **SELECT** haruslah menggunakan penunjuk dari table mana field tersebut dipilih. Format penulisannya adalah dengan menggunakan nama table, diikuti dengan tanda titik, dan kemudian nama field dari table tersebut. Hal ini sebenarnya tidak diperlukan, selama field tersebut hanya ada pada salah satu table saja. Namun sebaliknya, hal tersebut wajib dituliskan jika field tersebut ada pada kedua table.

Yang terakhir adalah penggunaan clause **WHERE**, di mana di sini ditentukan kondisi bahwa nilai dari field untuk kedua table tersebut adalah sama.

Jadi, query di atas bisa Anda tuliskan kembali sebagai berikut:

```
SELECT Jual_id, Jual_tgl, Cust_nama
FROM Jual, Customer
WHERE Jual.Cust_id = Customer.Cust_id
```

Dengan hasil query tersebut:

	Jual_id	Jual_tgl	Cust_nama
1	0001.05.14	2014-01-27 00:00:00	Dua Perdua
2	0002.05.14	2014-02-03 00:00:00	Cash
3	0003.05.14	2014-05-20 00:00:00	Cash

Gambar 4-17. Hasil dari dua buah table yang digabungkan

Mengambil Data dari Tiga buah Table

Bagaimana dengan penulisan **SQL** untuk data yang diambil dari tiga buah table? Seperti misalnya untuk daftar barang. Selain diambil dari table **Barang**, daftar ini juga perlu untuk mengambil data nama kategori dari table **Kategori**, dan data stock barang dari table **Barang_details**.

Perhatikan struktur table, dan bisa Anda lihat bahwa antara table **Barang** dengan table **Kategori** terdapat sebuah relationship, dengan pasangan **Primary Key-Foreign Key** pada field **Kat_id**. Sedangkan untuk table **Barang** dengan **Barang_details**, field kunci yang digunakan adalah field **Brg_id**.

Penulisannya sama saja dengan ketiga aturan di atas, sehingga **SQLSELECT** yang Anda tuliskan akan berformat:

```
SELECT Barang.Brg_id, Brg_nama, Brg_unit, Brg_barcode,
Brg_hpp, Kat_nama, Brg_det_stock
FROM Barang, Kategori, Barang_details
WHERE Barang.Kat_id = Kategori.Kat_id AND Barang.Brg_id =
Barang_details.Brg_id
```

Perhatikan bahwa Anda menggunakan operator **AND** untuk memastikan bahwa kedua kondisi tersebut harus dipenuhi, untuk kemudian memberikan hasil sebagai berikut:

Brg_id	Brg_nama	Brg_unit	Brg_barcode	Brg_hpp	Kat_nama	Brg_det_stock
82	T Almon Putih	PCS	A0004891	95000	SMPG	10
83	T Almon Ungu	PCS	A0002602	95000	SMPG	10
84	T Alisa Coklat	PCS	A0000401	77500	CKL	1
85	T Alisa Hitam	PCS	A0000400	77500	CKL	1
86	T Alisa Ungu	PCS	A0000399	65000	CKL	1
87	T Alun Abu	PCS	A0000441	72500	SMPG	10
88	T Alun Biru	PCS	A0001879	72500	SMPG	10

Gambar 4-18. Hasil penggabungan tiga buah table

Dalam aplikasi Anda nantinya ada beberapa form yang akan digunakan untuk menerima masukan dari user, dan kemudian akan menyimpan data yang didapat ke dalam database.

Menyimpan Data

Untuk menyimpan data baru pada sebuah table di dalam database, Anda tidak lagi menggunakan clause **SELECT ... FROM**, namun menggunakan clause **INSERT INTO ...**

Pada pengisian data Barang, misalnya, user akan diminta untuk memasukkan data-data barang baru yang akan dimasukkan. Selain itu, status stock barang tersebut juga dicatat, dalam hal ini, pada table **Barang_details**.

Table **Barang_details** ini hanya memiliki dua buah field, dan untuk memasukkan data, dibutuhkan format **SQL INSERT** sebagai berikut:

```
INSERT INTO Barang_details (Brg_id, Brg_det_stock)
VALUES ('<data_brg_id>', <data_brg_det_stock>)
```

Daftar field yang akan diisi setelah nama table tidak harus dituliskan, namun akan sangat membantu pembacaan bila dituliskan, jadi bisa menghindarkan kesalahan peletakan isi data yang akan disimpan.

Dengan tipe field data string, nilai yang akan disimpan agar diapit dengan tanda petik tunggal (misal, 'Jl. Bunga Raya'). Untuk tipe numeric, seperti Integer, tuliskan apa adanya.

Jadi, jika dicontohkan dengan statement di atas:

```
INSERT INTO Barang_details (Brg_id, Brg_det_stock)
VALUES ('PNC-UG', 6)
```

akan membuat sebuah record baru pada table **Barang_details**, dengan isi field **Brg_id** = 'PNC-UG' dan field **Brg_det_stock** = 6.

Mengubah Data

Pengubahan data diperlukan dalam sebuah aplikasi yang berbasis pada data. Bagaimana jika seandainya sebuah produk naik harga modalnya, disebabkan karena kenaikan ongkos kirim, atau penambahan pajak, dan sebagainya? Aplikasi Anda harus memiliki fitur untuk menemukan data yang mau dicari, dan kemudian mengubah salah satu, atau semua field datanya.

Format penulisannya cukup jelas. Sebagai contoh adalah Anda hendak mengubah stock barang, untuk barang dengan kode **Brg_id** = 'PNC-UG'. Untuk itu, Anda perlu melakukan **UPDATE** terhadap table **Barang_details**.

```
UPDATE Barang_details
SET Brg_det_stock = 8
WHERE Brg_id = 'PNC-UG'
```

Tanpa menggunakan kondisi **WHERE**, maka statement **UPDATE** ini akan mengubah data untuk semua record, tidak terkecuali. Misalkan, karena kenaikan harga BBM, maka perhitungan harga pokok penjualan untuk

semua barang dinaikkan sebesar 5%, maka perintah untuk itu adalah sebagai berikut:

```
UPDATE Barang  
SET Brg_hpp=Brg_hpp * 1.05
```

Menghapus Data

Penghapusan data, bagaimanapun juga, tetap diperlukan dalam suatu aplikasi. Misalnya, sebuah produk tidak lagi bisa didatangkan, karena sudah tidak diproduksi lagi oleh supplier. Atau seorang user tidak lagi menjadi karyawan toko Anda.

Untuk menghapus, digunakan statement DELETE.

```
DELETEFROM Barang  
WHERE Brg_id = 'PNC-UG'
```

Harap diperhatikan di sini, bahwa sebelumnya Anda sudah menetapkan relationship untuk table **Barang** ini terhadap table **Barang_details**. Record pada table **Barang_details** akan mengacu pada table **Barang**. Karena itu, apabila Anda menghapus sebuah record pada table **Barang**, maka **SQL Server** akan memberikan pesan kesalahan, dan proses penghapusan akan gagal. Kondisi ini disebut dengan **Referential Integrity**, yaitu konsep yang memastikan bahwa relationship antar-table tetap konsisten.

Dalam hal penghapus data **Barang** ini, yang harus Anda lakukan supaya tetap memenuhi aturan referential integrity tersebut adalah melakukan penghapusan untuk record tersebut dari table **Barang_details** terlebih dahulu. Setelah record pada table itu terhapus, maka proses penghapusan pada table **Barang** akan bisa dilakukan.

```
DELETEFROM Barang_details  
WHERE Brg_id = 'PNC-UG'
```

```
DELETEFROM Barang  
WHERE Brg_id = 'PNC-UG'
```

Dan seperti pada statement UPDATE di atas, statement DELETE tanpa menggunakan kondisi WHERE akan menghapus seluruh record dalam table tersebut.

Misalkan toko Anda melakukan stock-opname, dan Anda akan menghapus semua data stock, untuk kemudian memasukkan data baru. Maka, perintah penghapusan Anda bisa ditulis sebagai berikut:

```
DELETEFROM Barang_details
```

Jika kemudian Anda akan membuang seluruh table maupun database, Anda bisa menggunakan perintah DROP.

```
DROP TABLE Barang_details
```

Setelah Anda mempelajari mengenai dasar-dasarnya, sekarang Anda sudah siap untuk mulai melakukan pemrograman sesungguhnya.

Yang pertama harus dilakukan tentu saja adalah melakukan persiapan.

DOWNLOAD SCRIPT:

Seluruh script Visual Basic yang ada di dalam buku ini bisa diunduh lewat link berikut: <http://thinkjubilee.com/download/>.

Perancangan Aplikasi

Dalam buku ini, contoh aplikasi yang akan dibuat adalah aplikasi Penjualan untuk sebuah toko retail tas. Aplikasi ini akan mencakup beberapa modul, yaitu:

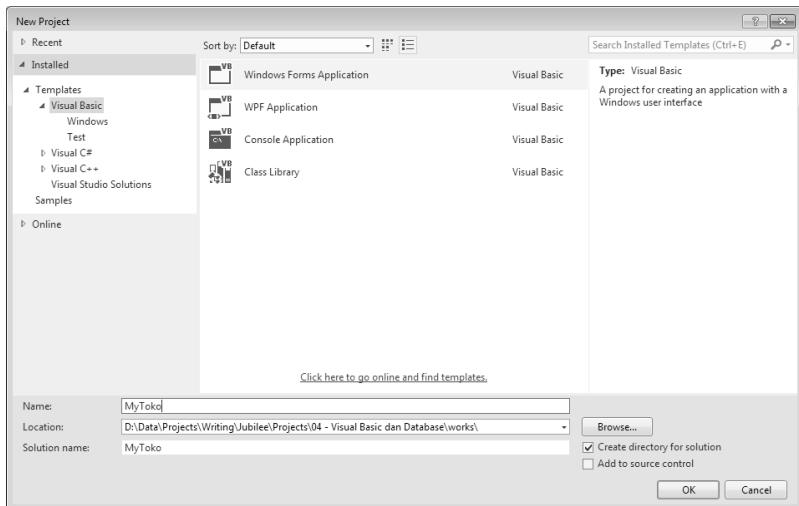
- Form Login, dengan hanya account khusus yang diperbolehkan untuk melakukan proses penambahan/pengubahan data dasar.
- Form untuk administrasi data dasar, yaitu **Customer, User, Kategori Barang, dan Barang**.
- Form untuk melakukan import data dasar dari format **.CSV (Comma-Separated Values)**.
- Form untuk mencatat proses penjualan barang, yang akan secara otomatis mengurangi stock untuk barang yang terjual

Aplikasi yang dimuat dalam buku ini bukanlah aplikasi lengkap sistem penjualan, namun modul-modul yang dimuat di dalamnya dibuat sebagai bahan pembelajaran, dan siap dikembangkan sesuai dengan kebutuhan Anda sebagai pengguna.

Persiapan Workspace

Persiapan ini adalah berupa mempersiapkan **Project** Anda pada **Microsoft Visual Studio 2013 Express**, dan kemudian menambahkan database Anda pada **Project**.

Jalankanlah program Anda, dan mulailah dengan membuat sebuah **Project** baru.



Gambar 5-1. Kotak dialog New Project

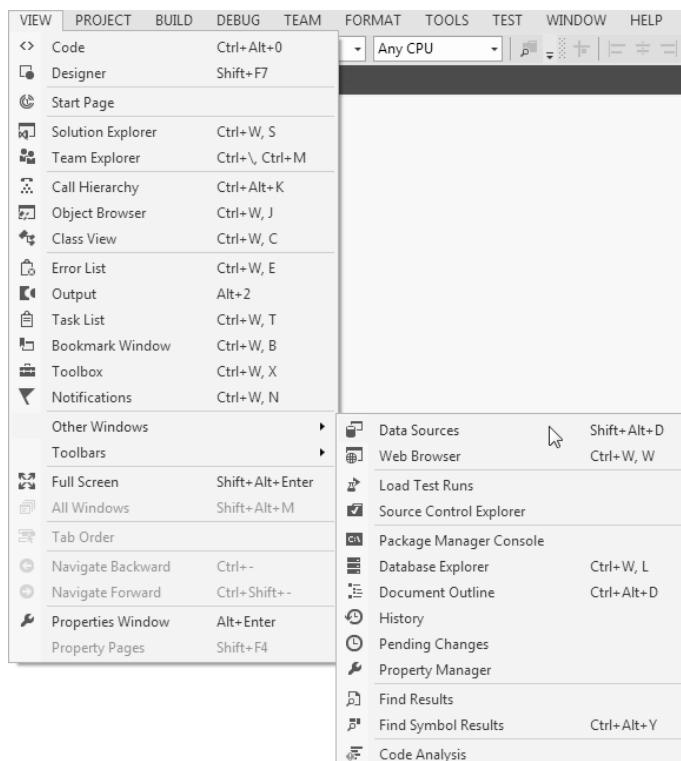
Pada kotak dialog ini, pastikan bahwa Anda memilih option yang benar. Sepanjang buku ini, pembahasan hanya akan menggunakan bahwa pemrograman **Visual Basic**. Oleh karena itu, pilihlah template **Visual Basic** dari bagian sebelah kiri. Kemudian, pilihlah untuk membuat **Windows Forms Application** di bagian tengah, dan diakhiri dengan menentukan nama aplikasi Anda.

Dalam contoh ini, akan digunakan mana aplikasi **MyToko**. Lokasi penyimpanan disesuaikan dengan kebutuhan Anda, lalu pastikan bahwa opsi untuk **Create directory for solution** di sebelah kanan bawah dipilih.

Tekan **OK** untuk melanjutkan. System akan memulai proses pembuatan project baru, dan setelah beberapa saat, workspace Anda akan ditampilkan.

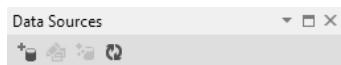
Langkah selanjutnya adalah menghubungkan database yang sudah Anda buat dengan aplikasi Anda. Untuk itu, Anda akan bekerja dengan sebuah dataset. Dataset adalah sebuah penampung yang akan menyimpan data-data baris dalam database Anda. Anda akan belajar untuk memanipulasi data dalam dataset, dan pada akhirnya menyimpannya secara permanen ke dalam database.

Bukalah tool window **Data Sources**. Jika pada workspace Anda belum ada tool window **Data Sources** itu, Anda bisa membukanya dari submenu **View > Other Windows > Data Sources**.



Gambar 5-2. Submenu untuk membuka Data Sources

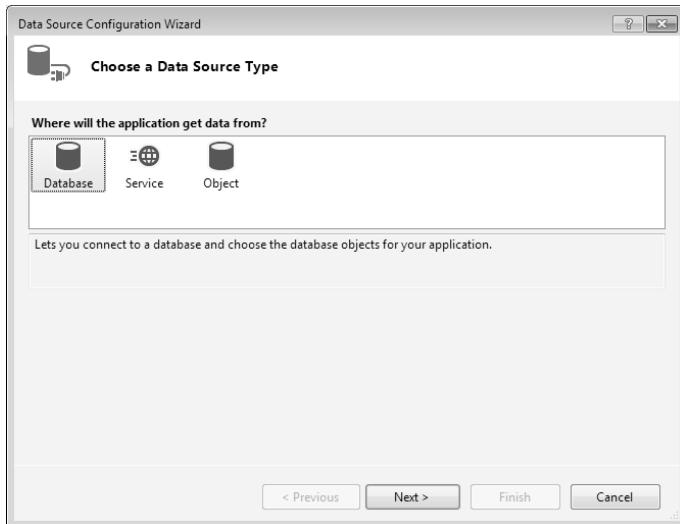
Setelah terbuka, biasanya di bagian sebelah kiri dari workspace Anda, Anda bisa melihat bahwa isinya kosong.



Gambar 5-3. Tool window Data Sources

Tambahkanlah sebuah **Data Source** melalui link di dalam tool window tersebut.

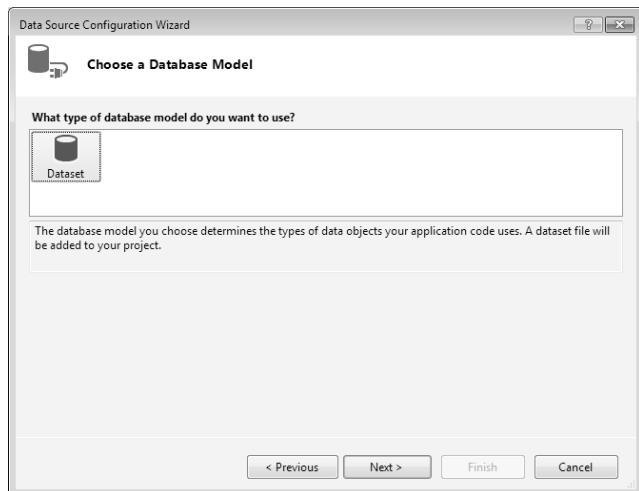
System akan membuka **Data Source Configuration Wizard**.



Gambar 5-4. Data Source Configuration Wizard

Pada pilihan ini, Anda akan memilih bahwa data Anda berasal dari sebuah database, yang telah Anda buat pada bab sebelumnya.

Tekan **Next** untuk melanjutkan.

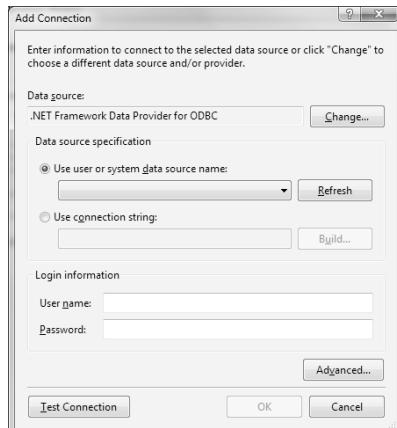


Gambar 5-5. Pemilihan tipe database

Pada langkah ini, opsi yang bisa Anda pilih hanyalah **Dataset**. Dan memang dataset inilah yang akan Anda gunakan dalam aplikasi Anda nantinya.

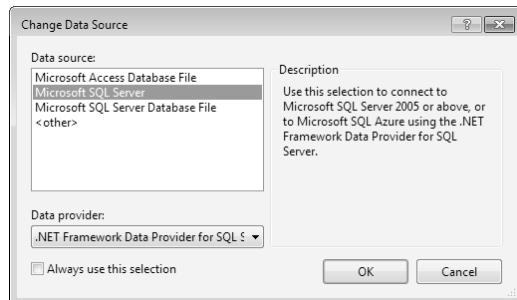
Jadi, pilihlah icon **Dataset**, dan tekan tombol **Next**.

Pada langkah berikut ini, sistem akan secara otomatis membuka kotak dialog **Add Connection**.



Gambar 5-6. Add Connection

Di sini, ada beberapa unsur yang harus Anda perhatikan. Yang pertama adalah **Data source** yang Anda gunakan. Secara default, sistem akan menampilkan pilihan data source pertama yang ada. Dalam contoh ini, sistem menampilkan **.NET Framework Data Provider for ODBC**. Apa pun yang terpilih di sini, pastikan Anda menggunakan yang benar. Pada pembahasan dalam buku ini, Anda menggunakan database dari **Microsoft SQL Server**, jadi jika pilihan saat ini bukan **Microsoft SQL Server**, tekan tombol **Change**, dan ganti menjadi **Microsoft SQL Server**.



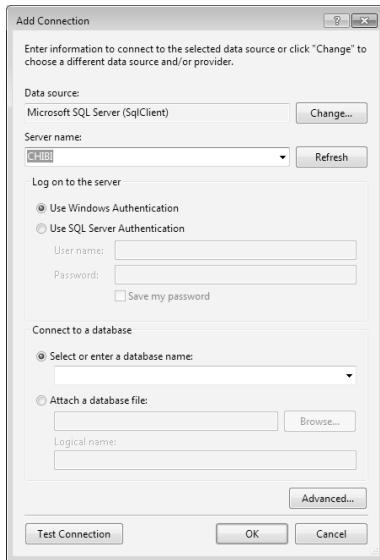
Gambar 5-7. Macam-macam pilihan data source

Setelah Anda memilih **Microsoft SQL Server**, maka Anda akan dikembalikan pada kotak dialog **Add Connection** seperti terlihat pada Gambar 5-8. Dan kotak dialog tersebut akan sedikit berbeda dibanding dengan sebelumnya. Perbedaan ini lebih difokuskan pada koneksi untuk **Microsoft SQL Server**.

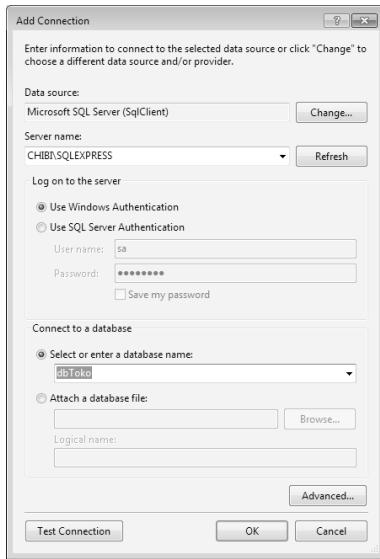
Pada bagian **Server name**, Anda perlu untuk memasukkan nama server bersama dengan instance **SQL Server Database** yang telah Anda instal sebelumnya.

Cara penulisannya adalah nama komputer, diikuti dengan tanda *backslash*, dan nama instance **SQL Server** Anda. Nama instance, jika Anda lupa, diatur pada **Bab 1** lalu, pada bagian **Instalasi SQL Server 2014 Express**.

Setelah Anda tuliskan dengan benar, pilihlah metoda logon, apakah menggunakan **Windows Authentication**, ataukah menggunakan **SQL Server Authentication**. Sekali lagi, informasi mengenai ini bisa Anda lihat kembali pada **Bab 1**.



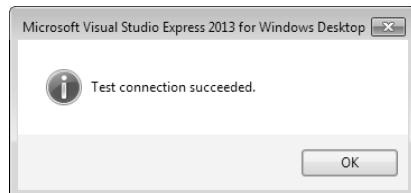
Gambar 5-8. Tampilan awal SQL Server connection



Gambar 5-9. Kotak dialog Add Connection setelah diisi dengan benar

Jika pengisian Anda sudah benar, maka pada bagian **Connect to a database**, lebih tepat lagi pada bagian **Select or enter a database name**, pada saat Anda membuka combo box tersebut, secara otomatis system akan menampilkan semua database yang ada dalam komputer Anda.

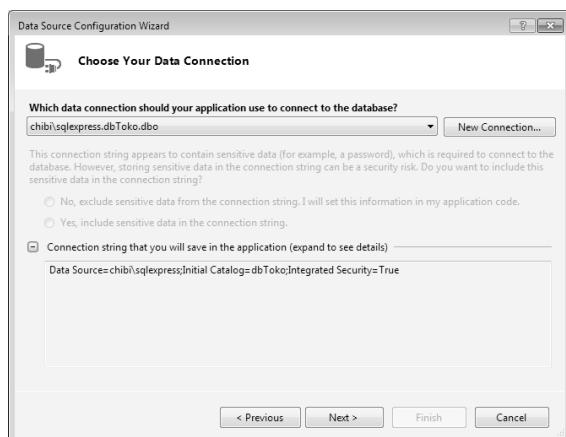
Setelah terisi dengan benar, Anda bisa melakukan **Test Connection** melalui tombol di bawah, untuk memastikan bahwa semua setting Anda sudah benar.



Gambar 5-10. Test connection succeeded

Tekan tombol **OK**, dan kemudian tekan tombol **OK** kembali pada kotak dialog **Add Connection**, untuk kembali ke **Data Soure Configuration Wizard**.

Namun, kali ini pada pilihan koneksi di atas sudah ada sebuah koneksi yang bisa Anda pilih.



Gambar 5-11. Data Source Configuration Wizard dengan isian data connection

Anda bisa menekan tanda tambah di bagian bawah sebelah kiri, untuk melihat *connection string* untuk data connection Anda.

Connection string adalah suatu string yang berisi data yang diperlukan untuk melakukan koneksi pada database. Anda cukup hanya menggunakan string tersebut untuk melakukan koneksi, seperti yang nanti akan Anda pelajari pada bagian selanjutnya.

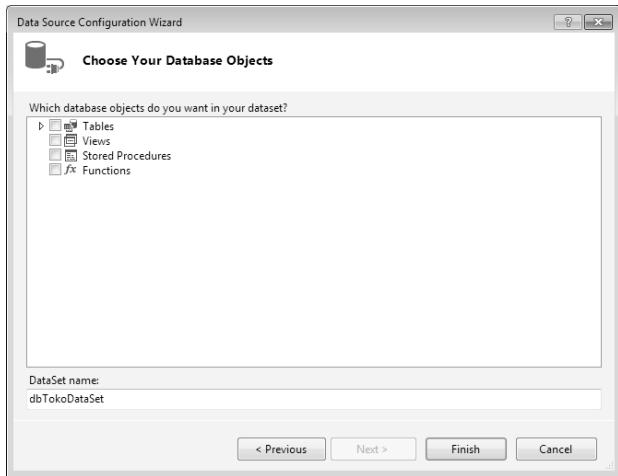
Langkah selanjutnya, adalah Anda menetapkan apakah Anda bersedia untuk menyimpan data connection string tadi ke dalam konfigurasi aplikasi Anda. Dengan menyimpannya dalam aplikasi, akan memudahkan dalam artian bahwa jika ada perubahan connection string, maka perubahan hanya perlu Anda ubah di satu tempat saja.

Pastikan checkbox di sebelah kiri Anda pilih, dan kemudian tuliskan nama connection string yang Anda inginkan dalam textbox yang tersedia. Nama usahakan yang cukup jelas, sehingga tidak menimbulkan kebingungan di waktu selanjutnya.



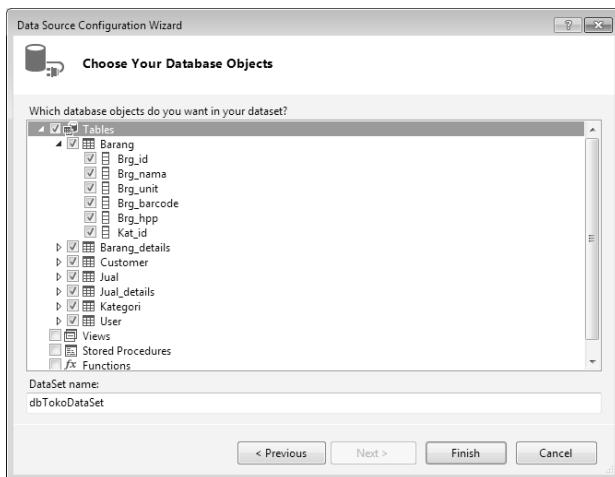
Gambar 5-12. Penyimpanan nama connection string

Tekan tombol **Next** untuk melanjutkan, di mana sistem akan mempersilahkan Anda untuk memilih object database mana yang akan Anda masukkan ke dalam dataset.



Gambar 5-13. Pemilihan object database

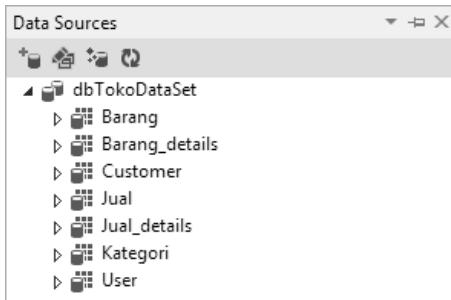
Ada beberapa macam object yang mungkin ada di dalam database, di antaranya akan bisa Anda lihat dalam gambar di atas. Namun, pada saat pembuatan database pada bab sebelum ini, Anda hanya membuat table saja. Karena itu, pilih checkbox di samping **Tables**. Tekan tanda anak panah kecil di samping checkbox tersebut untuk melihat table-table yang terdapat dalam database Anda.



Gambar 5-14. Table dan Column di dalamnya

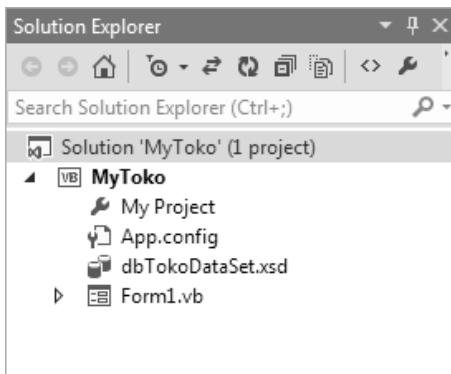
Selesailah sudah proses penambahan data source ke dalam aplikasi Anda.

Anda bisa melihat hasilnya pada tool window **Data Sources**, akan ditampilkan dataset Anda di sana.



Gambar 5-15. Dataset dalam Data Sources window

Selain dari itu, pada tool window **Solution Explorer** di sebelah kanan workspace Anda, dataset Anda juga akan ditampilkan di sana. Secara fisik, dataset Anda akan dibuatkan sebuah file dengan ekstensi **.xsd** pada folder di mana Anda menyimpan file aplikasi Anda.



Gambar 5-16. Dataset dalam Solution Explorer window

Setelah dataset Anda selesai dimasukkan ke dalam project Anda, maka selesailah proses persiapan ini. Anda sudah bisa untuk mulai melakukan pemrograman Anda.

###

BAB 6

Form Import Data

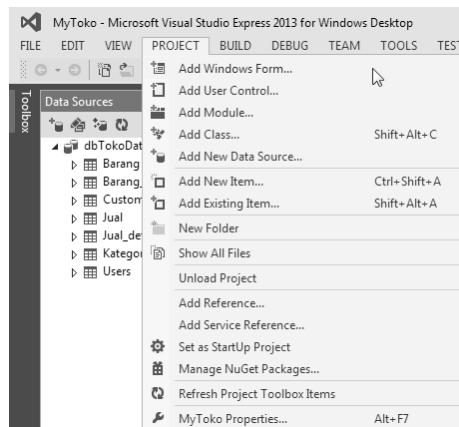
Ada baiknya form **Import Data** ini dibahas di awal pemrograman. Konsep dari pembuatan aplikasi ini adalah untuk membantu Anda belajar dalam hal pemrograman database. Dan dalam pembelajaran seperti ini, diasumsikan adalah bahwa Anda sudah mengerti dasar-dasar sistem **Visual Basic** maupun **SQL Server**. Karena itu, tidak seperti langkah pada saat Anda belajar bahasa **SQL**, yaitu mulai dari statement **SELECT**, di sini Anda tidak harus memulai dari form awal, yaitu form Login atau **Utama**. Alasan praktis yang bisa dikemukakan adalah dengan selesainya form **Import Data** ini terlebih dahulu, Anda bisa mulai memasukkan data dasar pada database, sehingga pemrograman seterusnya bisa dilanjutkan dengan lancar. Data dasar di sini digunakan untuk form-form lainnya, seperti basis data user (digunakan dalam form Login), basis data barang (digunakan dalam form **Jual**), basis data customer (digunakan dalam form **Jual**), dan basis data Kategori (digunakan dalam form **Barang**).

Dalam form ini, nantinya akan digunakan sebuah **ComboBox**, untuk memperbolehkan seorang user yang memiliki privilege sebagai Administrator untuk memilih table mana yang akan diisi datanya.

Selain itu, form ini juga akan memiliki sebuah **OpenFileDialog**, yang akan digunakan untuk membantu user tersebut untuk memilih file data, yang berupa text dengan data dipisahkan oleh tanda koma (**.CSV – comma separated values**).

Dan setelah user selesai menentukan table mana dan file **.CSV** sudah dipilih, maka user bisa menekan tombol **Import**, dan form akan membuka koneksi ke table tersebut, membaca text file, dan memasukkan data ke dalam table.

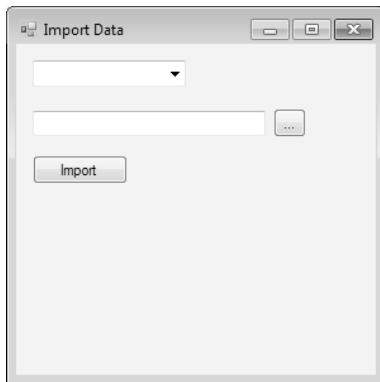
Untuk itu, mulailah untuk menambahkan sebuah form baru, untuk kemudian digunakan sebagai form **Import Data**.



Gambar 6-1. Add Windows Form

Pada dialog selanjutnya, pastikan yang terpilih adalah **Windows Form**, dan tuliskan namanya sebagai **frmImport.vb**.

Setelah form ditambahkan, form tersebut akan langsung ditampilkan dalam workspace Anda. Tambahkanlah empat buah object pada form tersebut, yaitu sebuah **ComboBox**, sebuah **TextBox**, dan dua buah **Button**. Aturlah supaya tampilan seperti dalam contoh berikut.



Gambar 6-2. Form Import Data

Dari form itu sendiri, yang harus Anda pastikan adalah bahwa nama dari form tersebut adalah **frmImport**. Form ini akan dipanggil dari form lain (form **Utama**), karena itu namanya harus Anda pastikan dengan benar.

Atur juga supaya property **Text** untuk form tersebut diisi dengan string “**Import Data**”.

Untuk ukuran dari form ini, ambil dari ukuran default, yaitu **300 x 300**.

Untuk item di dalam form tersebut, ikutilah detail berikut:

ComboBox

Name = **cbTables**

TextBox

Name = **tFilename**

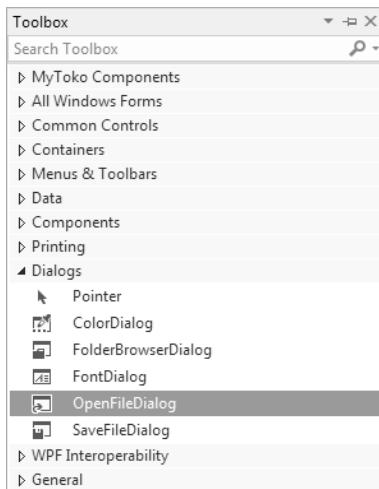
Button1

Name = **btnBrowse**

Button2

Name = **btnImport**

Di atas disebutkan bahwa dalam form ini akan ada sebuah object ber-nama **OpenFileDialog**. Untuk itu, tariklah dialog tersebut dari dalam Toolbox ke form Anda.



Gambar 6-3. OpenFileDialog

Setelah Anda tambahkan, object tersebut tidak akan ditampilkan dalam form Anda, tapi akan terlihat pada bagian tersendiri di bagian bawah

dari workspace Anda. Gantilah nama dari object tersebut menjadi **OpenFD** (dengan memilih object tersebut, dan mengganti property **Name**-nya).



Gambar 6-4. OpenFileDialog object

Langkah berikutnya adalah memberikan nilai awal pada item **ComboBox**. **ComboBox** ini akan memberikan pilihan pada user mengenai table apa yang akan diimpor datanya.

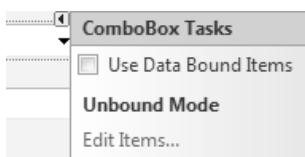
Pilihlah item **ComboBox** tersebut. Setelah terpilih, akan tampak sebuah icon kecil berisi anak panah.



Gambar 6-5. Anak panah kecil

Tanda itu akan menunjukkan context menu dari item yang dipilih. Untuk item **ComboBox**, pilihan data bisa berupa data yang diambil dari suatu table, atau bisa saja dengan menggunakan text statis. Dalam contoh ini, Anda akan menggunakan text statis.

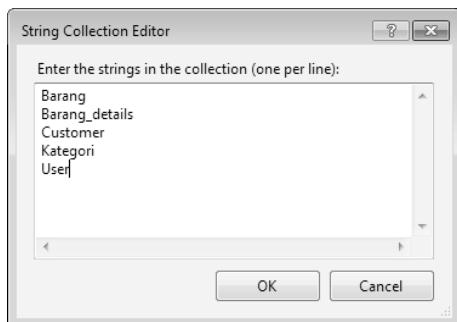
Karena itu, tekan tanda panah itu, dan pilihlah di bagian **Unbound Mode** di bawah, link untuk **Edit Items...**



Gambar 6-6. ComboBox Tasks

Link tersebut akan membuka sebuah text editor kecil, di mana Anda bisa memasukkan data pilihan yang akan dimunculkan dalam **ComboBox**.

tersebut. Di sini akan dimasukkan nama table-table dasar, yaitu **Barang** dan **Barang_details**, **Customer**, **Kategori** dan **Users**.



Gambar 6-7. Isian ComboBox

Sampai di sini, selesailah persiapan form **Import Data** ini. Selanjutnya, semuanya akan diselesaikan dengan pemrograman.

Pemrograman Database

Sebelum mulai masuk ke dalam pemrograman form ini, terlebih dahulu Anda akan dikenalkan dengan konsep koneksi data.

Di sini, Anda akan belajar mengenai object yang bisa Anda pakai untuk membuka maupun membaca data dari sebuah database, secara pemrograman (tanpa menggunakan wizard).

Untuk membuat suatu koneksi antara aplikasi Anda dengan database, Anda memerlukan apa yang disebut sebagai **Connection Object**. Salah satu yang Anda akan pelajari di sini adalah **OLE DB**.

OLE merupakan singkatan dari **Object Linking and Embedding**, yang adalah sekumpulan object yang dikelompokkan bersama secara khusus untuk melakukan koneksi ke suatu sumber data (bukan hanya database). Sumber data di sini bisa berupa database dari **SQL Server**, **Microsoft Access**, atau bahkan text file, email server, dan lain sebagainya.

Anda memerlukan beberapa buah variable dalam proses koneksi ini:

```
Dim con AsNew OleDb.OleDbConnection
```

Variable **con** ini sekarang menjadi tempat penyimpan dari connection object.

Sebagai sebuah object, tentu saja variable **con** memiliki banyak method maupun property. Saat ini, Anda akan mulai dengan sebuah property bernama **ConnectionString**.

Ingatkah pembahasan mengenai connection string sebelum ini? Connection string semacam itulah yang akan digunakan sebagai isi dari variable ini.

Di sini Anda memerlukan sebuah **Provider** dan sebuah **Data Source**, sebagai isi dari property **ConnectionString** ini. Provider adalah teknologi yang digunakan untuk melakukan koneksi, dan Data Source adalah penunjuk lokasi data Anda. Tambahkan deklarasi variable lain berikut di bawah deklarasi variable **con**, diikuti langsung dengan pemberian nilai untuk variable tersebut.

```
Dim dbProvider AsString  
Dim dbSource AsString  
  
dbProvider = "Provider=SQLOleDb;"  
dbSource = "Data Source=chibi\sqlexpress;Initial " &  
    "Catalog=dbToko;Integrated Security=SSPI"  
con.ConnectionString = dbProvider & dbSource
```

Pada bagian **dbProvider**, Anda menggunakan **SQLOleDB** sebagai provider-nya. Anda menggunakan string ini karena database Anda adalah **SQL Server** (jika misalnya Anda menggunakan database dari **Microsoft Access**, maka provider Anda akan menggunakan **Microsoft.Jet.OLEDB.4.0**, keterangan mengenai provider apa yang bisa Anda pakai bisa Anda dapatkan melalui Internet).

Sedangkan pada bagian **dbSource**, Anda menuliskan connection string yang bisa Anda dapatkan dari setting data source Anda.

Pada akhirnya, kedua string tersebut digabung menjadi satu untuk menjadi isi dari property **ConnectionString**.

Listing lengkap dari proses persiapan koneksi Anda akan menjadi seperti berikut:

```
Dim con AsNew OleDb.OleDbConnection  
Dim dbProvider AsString  
Dim dbSource AsString  
  
dbProvider = "Provider=SQLOleDb;"  
dbSource = "Data Source=chibi\sqlexpress;Initial " &  
    "Catalog=dbToko;Integrated Security=SSPI"  
con.ConnectionString = dbProvider & dbSource
```

Jadi, sampai pada proses ini, Anda sudah mempersiapkan connection string Anda, dan menyimpannya dalam variable object **con**. Langkah selanjutnya adalah membuka koneksi itu sendiri.

Tuliskan baris berikut untuk membuka koneksi tersebut,

```
con.Open()
```

Setelah dibuka, Anda bisa mulai melakukan pembacaan ataupun manipulasi data. Namun, setelah selesai, Anda harus menutupnya kembali,

```
con.Close()
```

Jadi, begitulah cara untuk melakukan koneksi pada database.

DataSet dan DataAdapter

Setelah Anda berhasil untuk membuka dan menutup database Anda, proses selanjutnya yang perlu dipelajari adalah proses penyimpanan data ke dalam tempat penyimpanan lokal, sehingga Anda dapat memanipulasi data tersebut.

Sebagai tempat penyimpan, anda akan menggunakan sesuatu yang disebut dengan **DataSet**. **DataSet** ini tidak berbentuk fisik, jadi tidak bisa Anda letakkan ke dalam form Anda seperti **Button** atau **TextBox**. **DataSet** adalah tempat penyimpan data di dalam memory. **DataSet** memiliki kolom dan baris, seperti pada tabel data pada umumnya. Jadi, setelah database berhasil Anda buka, Anda bisa mengambil datanya dan menyimpannya dalam **DataSet**. Kemudian, semua proses yang akan Anda lakukan pada data, Anda lakukan pada **DataSet** ini, dan setelah selesai, Anda dapat menyimpan **DataSet** ini kembali pada database.

Namun, connection object pada pembahasan sebelum, dengan dataset pada pembahasan ini, tidak bisa saling berkomunikasi secara langsung. Mereka membutuhkan perantara komunikasi, yang dalam hal ini dikenal sebagai **DataAdapter**. **DataAdapter** akan melakukan koneksi pada connection object, melakukan query padanya, dan kemudian menyimpan hasil query tersebut pada dataset.

Tambahkan deklarasi variable berikut:

```
Dim da As OleDb.OleDbDataAdapter  
Dim ds As New DataSet  
Dim Sql As String
```

DataSet dan **DataAdapter** adalah object, karena itu pendeklarasian mengikut pada aturan pendeklarasian object.

DataAdapter sendiri merupakan property dari object **OLEDB**, karena itu penulisannya bisa menggunakan pemisah tanda titik.

Variable **Sql** akan digunakan untuk menyimpan query dasar, misalnya untuk table **Customer**:

```
Sql = "SELECT * FROM Customer"
```

Langkah berikutnya:

```
da = New OleDb.OleDbDataAdapter(Sql, con)
```

Di sini, **DataAdapter** tersebut disimpan pada variable **da**, di mana proses penggunaan **DataAdapter** itu sendiri diberikan dengan dua buah parameter, yaitu sebuah variable string (di sini bernama **Sql**), dan variable connection object Anda, yaitu **con**.

Di atas dijelaskan sedikit, bahwa dalam proses pembukaan database, setelah Anda membuka koneksi pada database, Anda melakukan query padanya, dan menyimpan hasilnya pada dataset.

Dan untuk menyimpan hasil pada dataset tersebut, Anda menggunakan method **Fill** dari variable object **da**. Method ini meminta dua buah parameter, yang pertama adalah variable dataset Anda, dan yang kedua sebuah string yang digunakan sebagai penanda data ini. String ini bisa apa saja, digunakan hanya untuk menandai (memberi nama) **DataAdapter** ini.

```
da.Fill(ds, "Customer")
```

Dengan penulisan baris tersebut, variable **ds** sekarang menjadi terisi dengan data-data yang Anda pilih dengan perintah yang disimpan pada variabel **Sql** sebelumnya.

Menampilkan Data dalam DataSet

Kemudian, kembali pada program Anda, tambahkan baris program berikut:

```
MsgBox(ds.Tables("Customer").Rows(0).Item(0))
```

Anda bisa menambahkannya di manapun setelah pengisian variable **ds**, bahkan setelah statement **con.Close()**. Harap perhatikan bahwa koneksi ke database bisa ditutup setelah tidak lagi dibutuhkan. Anda bisa melihat bahwa koneksi tersebut sudah digunakan untuk mengisi variable **da**,

yang kemudian mengisi variable **ds** dengan **dataset**. Setelah pengisian berhasil, Anda bisa menutup koneksi ke database.

Perhatikan apa yang ditampilkan dalam **MsgBox** itu.

```
ds.Tables("Customer").Rows(0).Item(1)
```

Pertama, Anda menuliskan variable object utama dari data yang ada, yaitu variable dataset **ds**. Kemudian, Anda menggunakan property **Tables** dari dataset itu, sambil mengirimkan ID dari table yang akan dibuka. ID ini adalah ID yang dituliskan pada saat Anda melakukan pengisian variable **ds** tersebut dari **DataAdapter**, yaitu **da.Fill(ds, "Customer")**. Jadi, Anda tidak menggunakan nama table sesungguhnya dalam database Anda.

Setelah itu, masih dengan tanda titik, yang berarti adalah Anda menggunakan property **Rows** dari **Tables** yang sudah dibuka. Di sini **Rows** akan menunjuk kepada baris ke berapa dari dataset yang akan diambil. Baris akan dimulai dari baris **0** untuk baris pertama, begitu seterusnya.

Masih berlanjut, dari baris pertama yang dipilih, Anda akan mengambil hanya satu buah field, yaitu field **Cust_nama**. Jika Anda melihat di dalam table Anda, field ini adalah merupakan field ke dua. Untuk mengambil field tersebut, Anda menggunakan property **Item** dari object **Rows** yang sudah dipilih terlebih dahulu. Dan untuk Item ini pengurutan juga dimulai dari angka **0** untuk field pertama, dan **1** untuk field ke dua.

Jadi, statement

```
ds.Tables("Customer").Rows(0).Item(1)
```

Akan mengambil data dari dataset **Customer**, untuk baris pertama dan item field yang kedua, yaitu Nama Customer.

Jika misalnya Anda hendak mengambil data Email dari customer yang tersimpan pada baris ke empat, maka statement bisa Anda tuliskan sebagai berikut:

```
ds.Tables("Customer").Rows(3).Item(2)
```

Demikianlah cara untuk membuka koneksi pada sebuah table dalam database, dan kemudian mengambil isi baris datanya.

Sekarang kembali pada program Anda, klik-ganda pada form Anda untuk masuk ke dalam mode programming, dan langsung pada sub-routine **frmImport_Load()**.

Anda akan buat sedemikian sehingga koneksi ke database dibuka pada saat form dibuka.

```

PrivateSub frmImport_Load(sender As Object, e AsEventArgs)
HandlesMe.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial "&
"Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource

    con.Open()
End Sub

```

Kembali ke bagian awal program, di bagian paling atas. Tuliskan beberapa baris berikut ini, sebelum dan setelah deklarasi *Public Class*.

```

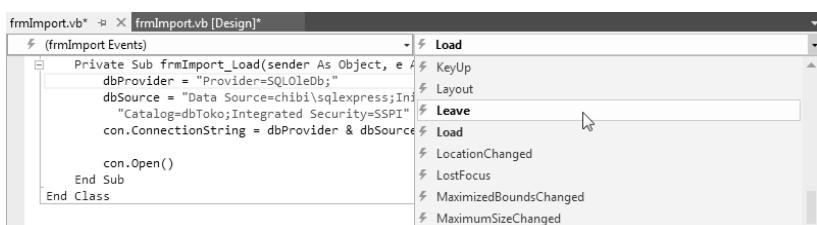
Imports System
Imports System.Data
Imports System.Data.OleDb
Imports Microsoft.VisualBasic.FileIO
PublicClassfrmImport
Dim con AsNew OleDb.OleDbConnection
Dim dbProvider AsString
Dim dbSource AsString
Dim ds AsNewDataSet
Dim da As OleDb.OleDbDataAdapter
Dim Sql AsString

```

Deklarasi variable dilakukan di luar sub-routine untuk memastikan bahwa semua sub-routine bisa menggunakannya.

Statement Imports digunakan untuk memberitahu system bahwa program Anda akan menggunakan Library **System**, **System.Data**, **System.Data.OleDb**, dan **Microsoft.VisualBasic.FileIO**.

Kembali ke sub-routine **Load**, gunakan pilihan object dan event di atas workspace Anda untuk mendapatkan sub-routine **Leave**. Sub-routine ini akan dijalankan pada saat form ini ditutup. Jadi, paling tepat digunakan untuk mengembalikan/menutup/menghapus variable yang digunakan.



Gambar 6-8. Event Form Load dan Leave

Dalam sub-routine baru ini, tuliskan baris pembersihan memory berikut:

```

PrivateSub frmImport_Leave(sender As Object, e AsEventArgs)
HandlesMe.Leave

```

```

    con.Close()
    con = Nothing
    ds = Nothing
    da = Nothing
EndSub

```

Kembalilah pada tampilan form, dan lakukan klik-ganda pada **btnBrowse**. Hal itu akan mengembalikan Anda pada tampilan program, dan pada sub-routine baru lagi.

Tuliskan baris program berikut:

```

PrivateSub btnBrowse_Click(sender AsObject, e AsEventArgs)
Handles btnBrowse.Click
    OpenFD.InitialDirectory = "D:\Data\ works\MyToko"
    OpenFD.Title = "Import CSV Data File"
    OpenFD.Filter = "CSV Files|*.csv"
Dim hasil AsInteger = OpenFD.ShowDialog()
If hasil <> DialogResult.Cancel Then
    tFilename.Text = OpenFD.FileName
EndIf
EndSub

```

Sub-routine ini akan dijalankan pada saat tombol tersebut ditekan. Dan apa yang dilakukannya adalah membuka object **OpenFileDialog** yang Anda tambahkan dalam form Anda.

Atur beberapa property-nya untuk membantu user dalam memilih file datanya. Setelah property ditentukan, dialog ini akan dipanggil dengan method **ShowDialog()**, dan jika user tidak menekan tombol **Cancel**, maka hasil filename yang dipilih akan disimpan dalam **TextBox** yang tersedia.

Langkah selanjutnya adalah langkah utama dalam form **Import Data** ini, yaitu proses **Import** itu sendiri. Sebelumnya, pastikan Anda sudah memiliki file datanya, dan disimpan dalam bentuk **.CSV**. Format penulisan kolom juga harus sesuai dengan isi dari table itu sendiri.

Misalnya Anda memiliki data Customer sebagai berikut:

A	B	C	D	E	F	G
1	1 Satu Persatu	satu_p@email.com	Jalan Persatu No. 1	Batam	0778-12345678	14/4/2013
2	2 Dua Perdua	dua_p@email.com	Jalan Perdua No. 2	Batam	0778-23456789	22/4/2013
3	3 Tiga Pertiga	tiga_p@email.com	Jalan Pertiga No. 3	Batam	0778-34567890	2/5/2013
4	4 Empat Perempat	empat_p@email.com	Jalan Perempat No. 4	Batam	0778-45678901	22/8/2013
5	5 Lima Perlima	lima_p@email.com	Jalan Perlima No. 5	Batam	0778-56789012	3/12/2013

Gambar 6-9. Data dasar untuk Customer

Tampilan data tersebut adalah dalam format **Excel**. Simpan data tersebut dengan ekstensi **.CSV**.

Proses pembacaan text file sendiri adalah sebagai berikut:

Sebagai awalnya, deklarasikan sebuah variable untuk menampung hasil pembacaan file.

```
Dim currentRow AsString()
```

Variable **currentRow** digunakan untuk menampungnya, dan memiliki tipe data array string, karena nantinya variable ini akan menampung data per kolom.

Selanjutnya, tuliskan perintah berikut untuk membaca datanya:

```
Using MyReader AsNewTextFieldParser(tFilename.Text)
    MyReader.TextFieldType = FieldType.Delimited
    MyReader.Delimiters = NewString() {",", "}"}
```

Di sini, Anda mendeklarasikan sebuah variable, yaitu **MyReader**, sebagai sebuah object dari library **Microsoft.VisualBasic.FileIO**, yaitu object **TextFieldParser()**. Object ini memiliki sebuah parameter, yaitu nama file yang akan dibuka.

Kemudian object ini akan diatur, dan ditentukan bahwa data di dalamnya adalah berupa data per kolom, yang dipisahkan oleh tanda koma.

Setelah itu, pasanglah sebuah looping **While ... End While**, dengan kondisi **Not MyReader.EndOfData**. Ini berarti bahwa proses akan terus dijalankan, selama belum sampai akhir dari file data.

```
WhileNot MyReader.EndOfData
```

Proses pembacaannya sendiri menggunakan statement berikut:

```
currentRow = MyReader.ReadFields()
```

Dengan contoh data seperti di atas, maka setelah proses ini dijalankan, maka isi dari variable **currentRow** tersebut adalah sebagai berikut:

```
CurrentRow(0) = 1
CurrentRow(1) = Satu Persatu
CurrentRow(2) = satu_p@email.com
CurrentRow(3) = Jalan Persatu No. 1
CurrentRow(4) = Batam
CurrentRow(5) = 0778-12345678
CurrentRow(6) = 14-04-2013
```

Jadi, setelah data mulai dibaca, maka langkah selanjutnya adalah mempersiapkan perintah untuk melakukan insert pada table, dan dataset untuk menampung data, dan kemudian menyimpannya.

```

Sql = "SELECT * FROM Customer"

Dim cmd As New OleDbCommand()
With cmd
    . CommandType = CommandType.Text
    . Connection = con
    . CommandText = Sql
End With

da = New OleDb.OleDbDataAdapter(cmd)

da.SelectCommand = cmd
Dim cb As New OleDbCommandBuilder(da)

da.Fill(ds, "Customer")

```

Sebagai awalnya, dideklarasikan sebuah variable cmd sebagai sebuah object OldDbCommand. Setelah ditentukan property-nya, kemudian di-deklarasikan, dan disimpan dalam variable da.

Variable **ds** akan diisikan dengan semua data dari table. Meskipun data belum ada, namun proses ini juga memiliki maksud lain, yaitu untuk mengatur dataset tersebut supaya memiliki format kolom yang sama dengan table.

Sekali lagi, karena table saat ini masih kosong, maka dataset setelah proses ini akan memiliki jumlah dan tipe kolom yang benar, namun tanpa data, atau masih kosong.

Tambahkanlah sebuah variable lagi.

```
Dim dsNewRow As DataRow
```

DataRow, seperti yang mungkin bisa Anda mengerti, adalah suatu object yang akan menyimpan satu baris data.

Setelah dideklarasikan, variable tersebut diberi nilai awal dari method **NewRow()**.

```
dsNewRow = ds.Tables("Customer").NewRow()
```

Di sini artinya adalah bahwa Anda menambahkan sebuah baris baru pada dataset "**Customer**", dan baris tersebut disimpan dalam variable **dsNewRow**.

Kemudian lakukan penyimpanan datanya, dengan menggunakan nilai dari variable **currentRow**.

```

dsNewRow.Item("Cust_id") = currentRow(0)
dsNewRow.Item("Cust_nama") = currentRow(1)
dsNewRow.Item("Cust_email") = currentRow(2)
dsNewRow.Item("Cust_alamat") = currentRow(3)
dsNewRow.Item("Cust_kota") = currentRow(4)

```

```
dsNewRow.Item("Cust_phone") = currentRow(5)
dsNewRow.Item("Cust_reg_on") = CDate(currentRow(6))
```

Perhatikan bahwa untuk kolom terakhir, digunakan sebuah fungsi **Cdate()**, untuk mengubah tipe data string dari input file menjadi tipe data Date.

Setelah variable **dsNewRow** memiliki data untuk ditambahkan, variable ini kemudian ditambahkan pada dataset.

```
ds.Tables("Customer").Rows.Add(dsNewRow)
```

Dan langkah terakhir adalah untuk memastikan data tersimpan dalam database, gunakan statement berikut.

```
da.Update(ds, "Customer")
```

Dengan itu, data akan tersimpan dalam dataset, dan kemudian disimpan juga dalam table sebenarnya pada database.

Pada akhirnya, tutup blok **While** dengan **End While**, dan **Using** dengan **End Using**.

```
EndWhile
EndUsing
```

Demikianlah cara membaca txtfile, dan kemudian mengimpor data yang dibaca dari txtfile tersebut ke dalam database.

Namun, selain dari cara tersebut di atas, masih ada cara lain lagi dengan menggunakan statement **SQL**.

Misalkan Anda membuat statement untuk mengimpor data table **Barang_details**.

Proses ini dimulai dengan deklarasi variable.

```
Dim DAInsert AsString
```

Proses pemberian nilai data adapter masih sama:

```
Sql = "SELECT * FROM Barang_details"
da = New OleDb.OleDbDataAdapter(Sql, con)
```

Namun, setelah itu, proses menjadi berbeda, di mana Anda kemudian mendeklarasikan statement **SQL** untuk melakukan **INSERT** data ke dalam table.

```
DAInsert = _
"INSERT INTO Barang_details (Brg_id, Brg_det_stock) VALUES(''_
& currentRow(1) & ','&CInt(currentRow(2)) & ")"
```

Di sini, statement **INSERT** menggunakan parameter nilai yang diambil dari kolom pertama dan kedua dari table **Barang_details**, yaitu kolom Kode Barang, dan kolom nilai stock.

Perintah untuk menjalankan statement **SQL** itu sendiri adalah sebagai berikut:

```
Dim objCmd AsNewOleDbCommand(DAInsert, con)
objCmd.ExecuteNonQuery()

objCmd = Nothing
```

Langkahnya adalah mendeklarasikan sebuah variable baru, yaitu **objCmd**, yang merupakan object perintah **OleDb**. Setelah itu, object menjalankan method **ExecuteNonQuery()** untuk menjalankan statement **SQL** tersebut.

Kemudian, setelah object tidak digunakan lagi, dihapus dari memory dengan cara memberinya nilai **Nothing**.

Demikianlah dua cara untuk mengimpor data ke dalam table database.

Listing lengkap dari sub-routine tersebut adalah:

```
PrivateSub btnImport_Click(sender AsObject, e AsEventArgs)
Handles btnImport.Click
Dim currentRowAsString()
IfNot (IsNothing(tFilename.Text)) Then
Using MyReader AsNewTextFieldParser(tFilename.Text)
    MyReader.TextFieldType = FieldType.Delimited
    MyReader.Delimiters = NewString() {","}
WhileNot MyReader.EndOfData
Try
    currentRow = MyReader.ReadFields()
SelectCase cbTables.Text
Case "Customer"
    Sql = "SELECT * FROM Customer"

Dim cmd AsNewOleDbCommand()
With cmd
    . CommandType = CommandType.Text
    . Connection = con
    . CommandText = Sql
EndWith

da = New OleDb.OleDbDataAdapter(cmd)
da.SelectCommand = cmd
Dim cb AsNewOleDbCommandBuilder(da)

da.Fill(ds, "Customer")

Dim dsNewRow AsDataRow
dsNewRow = ds.Tables("Customer").NewRow()

dsNewRow.Item("Cust_id") = currentRow(0)
```

```

        dsNewRow.Item("Cust_nama") = currentRow(1)
        dsNewRow.Item("Cust_email") = currentRow(2)
        dsNewRow.Item("Cust_alamat") = currentRow(3)
        dsNewRow.Item("Cust_kota") = currentRow(4)
        dsNewRow.Item("Cust_phone") = currentRow(5)
        dsNewRow.Item("Cust_reg_on") = _
            CDate(currentRow(6))

    ds.Tables("Customer").Rows.Add(dsNewRow)

        da.Update(ds, "Customer")
Case "Barang"
    Sql = "SELECT * FROM Barang"

Dim cmd As New OleDbCommand()
With cmd
    . CommandType = CommandType.Text
    . Connection = con
    . CommandText = Sql
End With

        da = New OleDb.OleDbDataAdapter(cmd)
        da.SelectCommand = cmd
Dim cb As New OleDbCommandBuilder(da)

        da.Fill(ds, "Barang")

Dim dsNewRow As DataRow

        dsNewRow = ds.Tables("Barang").NewRow()

        dsNewRow.Item("Brg_id") = currentRow(0)
        dsNewRow.Item("Brg_nama") = currentRow(1)
        dsNewRow.Item("Brg_unit") = currentRow(2)
        dsNewRow.Item("Brg_barcode") = currentRow(3)
        dsNewRow.Item("Brg_hpp") = currentRow(4)
        dsNewRow.Item("Kat_id") = currentRow(5)

    ds.Tables("Barang").Rows.Add(dsNewRow)

        da.Update(ds, "Barang")
Case "Barang_details"
Dim DAInsert As String
    Sql = "SELECT * FROM Barang_details"

        da = New OleDb.OleDbDataAdapter(Sql, con)

        DAInsert = "INSERT INTO Barang_details (Brg_id" &_
        ", Brg_det_stock) VALUES ('"& currentRow(1) &_
        "' , "&CInt(currentRow(2)) &")"

Dim objCmd As New OleDbCommand(DAInsert, con)
        objCmd.ExecuteNonQuery()

        objCmd = Nothing

End Select

Catch ex As MalformedLineException

```

```

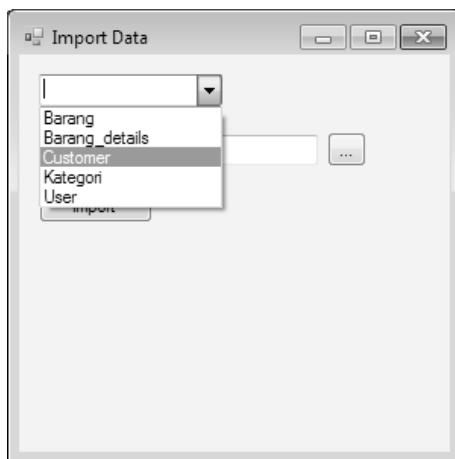
        MsgBox("Line "& ex.Message &_
    " is invalid. Skipping")
EndTry
EndWhile
EndUsing
EndIf
EndSub

```

Perhatikan beberapa hal berikut:

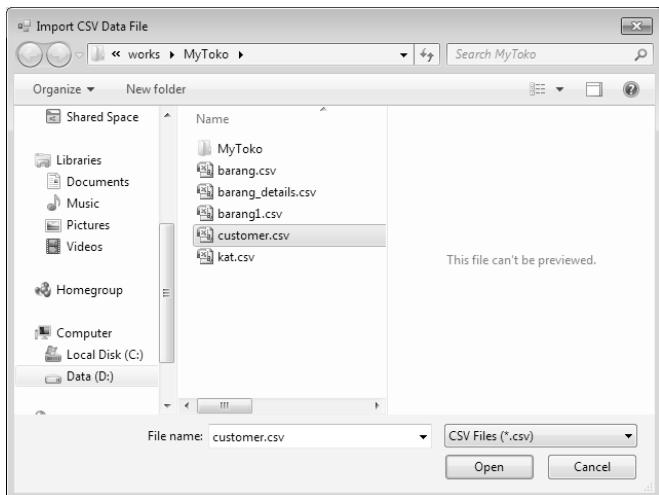
- Di awal modul, diberikan kondisi yang akan memastikan bahwa sub-routine dijalankan hanya apabila textbox sudah ditentukan.
- Setelah masuk di dalam looping **While**, digunakan sebuah **Try ... Catch** block untuk menangkap terjadinya *exception* (terutama karena proses pembacaan file), dan jika terjadi suatu kesalahan, maka sebuah *msgbox* akan ditampilkan, dan baris tersebut tidak akan digunakan.
- Digunakan sebuah block **Select ... Case ... End Select** untuk pilihan nama table (dari **ComboBoxTables**). Dalam listing dibuat fungsi **Import** untuk table data **Customer**, **Barang** dan **Barang_details** saja. Untuk data lain Anda bisa menambahkan sendiri dengan contoh dari buku ini.

Tampilan dari modul ini adalah sebagai berikut:



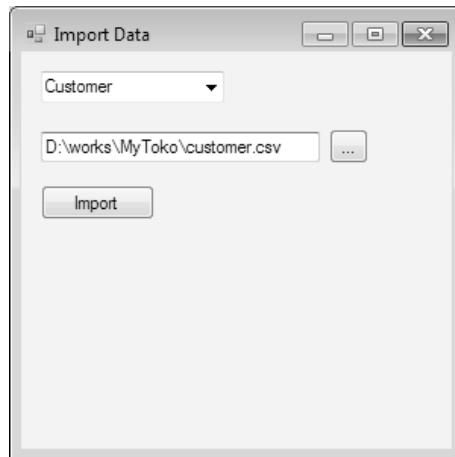
Gambar 6-10. Tampilan awal program dan ComboBox

ComboBox akan ditampilkan sesuai dengan data apa yang Anda masukkan sebelumnya.



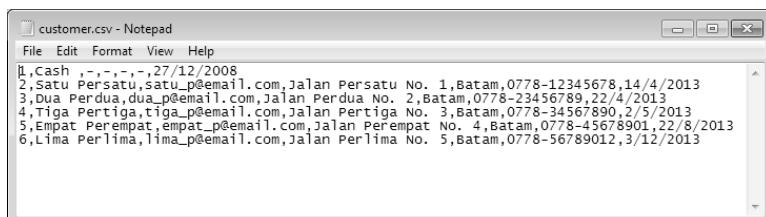
Gambar 6-11. OpenFileDialog

Jika Anda menekan tombol **btnBrowse**, maka akan terbuka standard **OpenFileDialog** dari **Windows**, yang digunakan untuk memilih textfile apa yang bisa digunakan. Sesuai dengan parameter yang Anda tentukan sebelumnya, **OpenFileDialog** ini hanya akan menerima masukan file dengan ekstensi **.CSV**.



Gambar 6-12. Setelah memilih textfile, form siap untuk Import

Pastikan terlebih dahulu sebelum melakukan proses ini bahwa data .CSV yang akan pergunakan sudah memiliki format yang sesuai. Format yang sesuai adalah dengan menggunakan pemisah field karakter tanda koma.



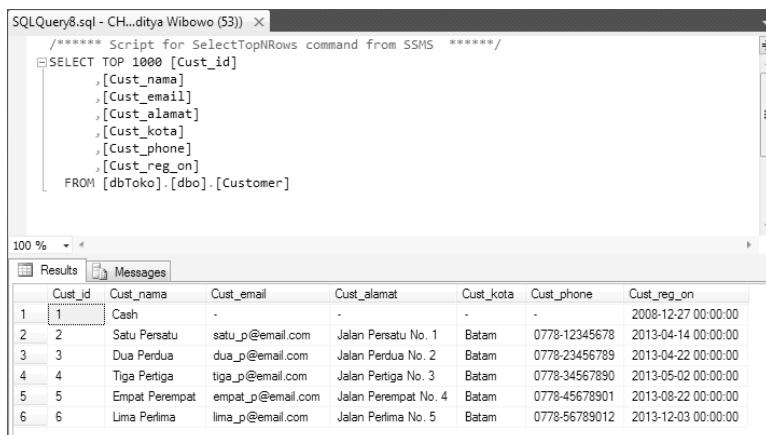
A screenshot of a Windows Notepad window titled "customer.csv - Notepad". The file contains the following CSV data:

```
1,Cash ,,-,-,-,27/12/2008
2,Satu Persatu,satu_p@email.com,Jalan Persatu No. 1,Batam,0778-12345678,14/4/2013
3,Dua Perdua,dua_p@email.com,Jalan Perdua No. 2,Batam,0778-23456789,22/4/2013
4,Tiga Pertiga,tiga_p@email.com,Jalan Pertiga No. 3,Batam,0778-34567890,2/5/2013
5,Empat Perempat,empat_p@email.com,Jalan Perempat No. 4,Batam,0778-45678901,22/8/2013
6,Lima Perlama,lima_p@email.com,Jalan Perlama No. 5,Batam,0778-56789012,3/12/2013
```

Gambar 6-13. Format .CSV untuk data Customer

Setelah data disiapkan, Anda bisa menekan tombol **Import**. Tergantung dari besar data yang Anda **Import**, proses ini bisa memakan waktu beberapa lama. Anda juga bisa memperkirakan, dari dua cara menyimpan data ke dalam dataset yang dibuat di sini, cara yang mana yang terlihat lebih cepat operasinya.

Demikianlah pembahasan untuk form **Import Data**. Setelah proses **Import** dijalankan, Anda bisa memeriksa dari **SQL Server Management Studio** mengenai isi table, sehingga Anda bisa memastikan bahwa data sudah dimasukkan.



A screenshot of the SQL Server Management Studio (SSMS) interface. The top pane shows a script window with the following T-SQL code:

```
===== Script for SelectTopNRows command from SSMS =====/
SELECT TOP 1000 [Cust_id]
,[Cust_nama]
,[Cust_email]
,[Cust_alamat]
,[Cust_kota]
,[Cust_phone]
,[Cust_reg_on]
FROM [dbToko].[dbo].[Customer]
```

The bottom pane shows the results of the query, displaying 6 rows of customer data:

Cust_Id	Cust_nama	Cust_email	Cust_alamat	Cust_kota	Cust_phone	Cust_reg_on
1	Cash		-	-	2008-12-27 00:00:00	
2	Satu Persatu	satu_p@email.com	Jalan Persatu No. 1	Batam	0778-12345678	2013-04-14 00:00:00
3	Dua Perdua	dua_p@email.com	Jalan Perdua No. 2	Batam	0778-23456789	2013-04-22 00:00:00
4	Tiga Pertiga	tiga_p@email.com	Jalan Pertiga No. 3	Batam	0778-34567890	2013-05-02 00:00:00
5	Empat Perempat	empat_p@email.com	Jalan Perempat No. 4	Batam	0778-45678901	2013-08-22 00:00:00
6	Lima Perlama	lima_p@email.com	Jalan Perlama No. 5	Batam	0778-56789012	2013-12-03 00:00:00

Gambar 6-14. Isi dari table Customer dilihat dari SSMS

###

BAB 7

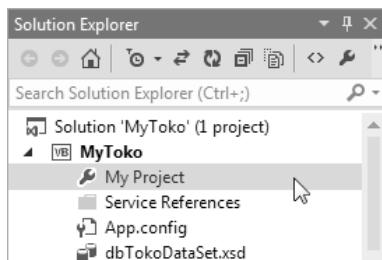
Form Barang Lihat Saja

Selanjutnya, akan dibahas mengenai **Form Barang – Lihat Saja**. Form ini, tidak seperti form data dasar yang lain, tidak akan memiliki kemampuan untuk menambah maupun mengubah data. Form ini hanya akan bisa menampilkan data, dan dibuat hanya untuk memberikan contoh apa yang bisa dilakukan dengan list data dasar.

Form Barang dengan konsep **Lihat Saja** ini akan dibuat sebagai sebuah form tampilan yang interaktif, di mana seorang user akan bisa menekan tombol untuk maju ke data berikut, atau mundur ke data sebelumnya. Selain itu, user juga bisa memasukkan input nomor baris untuk melompat langsung ke baris data yang diinginkan.

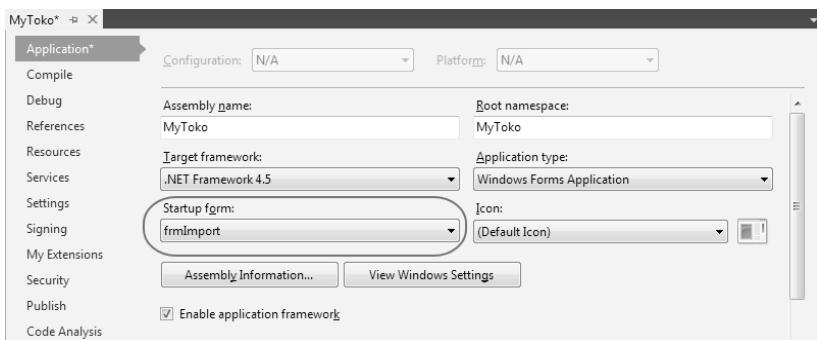
Tambahkanlah sebuah form baru, dengan nama **frmBarang1**. Mengapa **frmBarang1**? Karena setelah pembahasan bab ini selesai, Anda akan membuat lagi sebuah **frmBarang**, namun dengan kemampuan edit dan tambah data.

Perhatikan pada bagian **Solution Explorer** di sebelah kanan, ada sebuah item yang bernama **My Project**.



Gambar 7-1. My Project pada Solution Explorer

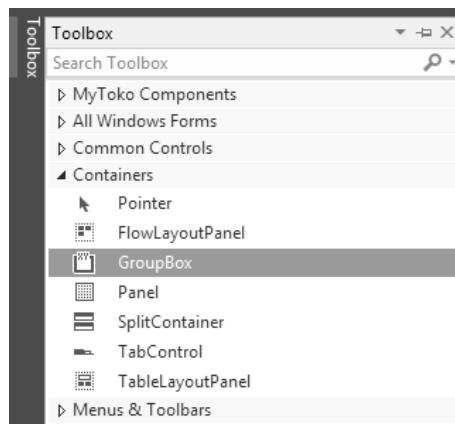
Lakukan klik-ganda pada item ini untuk membuka setting dari project Anda.



Gambar 7-2. My Project setting

Perhatikan pada bagian **Startup form**. Setting di sini yang memberitahu **Visual Studio** untuk membuka form awal, dalam gambar di atas masih merujuk pada form pertama yang Anda buat, yaitu **frmImport**. Gantilah menjadi **frmBarang1**, lalu simpan, dan kembali pada tampilan awal **frmBarang1** Anda.

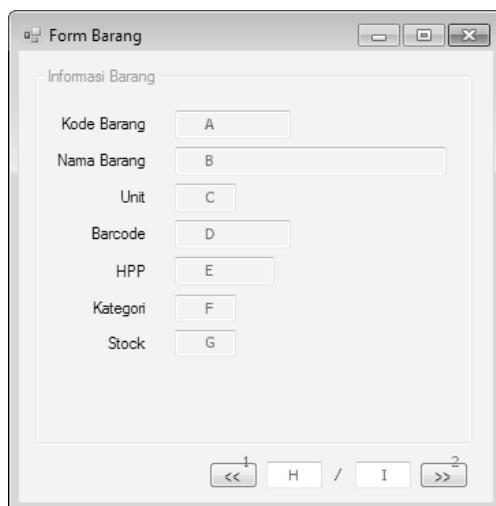
Pertama, tambahkanlah sebuah **GroupBox** pada form Anda, diambil dari sub-bagian **Containers** pada **Toolbox** di sebelah kiri.



Gambar 7-3. Containers – GroupBox

GroupBox ini digunakan untuk mengelompokkan kontrol yang akan Anda tambahkan dalam form Anda. Form ini akan menampilkan semua element dari table **Barang**, dan digabungkan dengan data **Kategori** dan data **Barang_details**. Untuk itu, sebuah **GroupBox** akan membantu untuk merapikan tampilan.

Setelah itu, tambahkan dan atur **8** buah **Label**, **9** buah **TextBox**, dan **2** buah **Button** menurut detail berikut:



Gambar 7-4. Tampilan frmBarang1

Selain dari nama form yang Anda tentukan, yaitu **frmBarang1**, ada beberapa **TextBox** dan **Button** yang juga harus Anda tentukan property-nya.

- A: Nama = tBrg_id
ReadOnly = True
- B: Nama = tBrg_nama
ReadOnly = True
- C: Nama = tBrg_unit
ReadOnly = True
- D: Nama = tBrg_barcode
ReadOnly = True

```
E: Nama = tBrg_hpp  
    ReadOnly = True  
  
F: Nama = tKat_nama  
    ReadOnly = True  
  
G: Nama = tBrg_det_stock  
    ReadOnly = True  
  
H: Nama = tCurRow  
  
I: Nama = tMaxRow  
    ReadOnly = True  
  
1: Nama = btnPrev  
  
2: Nama = btnNext
```

Setelah selesai setting tampilan, alihkan tampilan Anda pada bagian program.

Sebagai awal, buatlah deklarasi variable untuk koneksi data, seperti pada bab sebelum ini.

```
Public Class frmBarang1  
Dim con As New OleDb.OleDbConnection  
Dim dbProvider As String  
Dim dbSource As String  
Dim ds As New DataSet  
Dim da As OleDb.OleDbDataAdapter  
Dim Sql As String
```

Setelah itu, susun proses pembukaan maupun penutupan koneksi ke database pada event **Form_Load** dan **Form_Leave**.

```
Private Sub frmBarang1_Leave(sender As Object, e As EventArgs)  
Handles Me.Leave  
    con = Nothing  
    ds = Nothing  
    da = Nothing  
End Sub  
  
Private Sub frmBarang1_Load(sender As Object, e As EventArgs)  
Handles MyBase.Load  
  
    dbProvider = "Provider=SQLOleDb;"  
    dbSource = "Data Source=chibi\sqlexpress;Initial "&  
    "Catalog=dbToko;Integrated Security=SSPI"  
    con.ConnectionString = dbProvider & dbSource  
  
    con.Open()  
    Sql = "SELECT Brg.Brg_id, Brg.Brg_nama, Brg.Brg_unit,  
    Brg.Brg_barcode, Brg.Brg_hpp, Kat.Kat_nama,  
    Brg_det.Brg_det_stock FROM Barang Brg, Kategori Kat,
```

```

Barang_details Brg_det WHERE Brg.Kat_id = Kat.Kat_id AND
Brg.Brg_id = Brg_det.Brg_id"

da = New OleDb.OleDbDataAdapter(Sql, con)
da.Fill(ds, "Barang")

con.Close()
EndSub

```

Perhatikan bahwa dalam form ini, Anda menggunakan statement **SQL SELECT** yang mengambil data dari tiga buah Table.

Jika Anda eksekusi statement tersebut terhadap database, maka akan didapatkan data sebagai berikut:

Brng_id	Brng_nama	Brng_unit	Brng_barcode	Brng_hpp	Kat_nama	Brng_det_stock
244	C-HLN...	T Helena Orange	A0009865	100000	CKL	1
245	C-HLN...	T Helena Pink Fanta	A0009866	100000	CKL	1
246	C-HLN...	T Helena Ungu	A0009868	100000	CKL	1
247	CIN-BM	T R Cindy Biru Muda	A0005823	102500	RSL	8
248	CIN-BR	T R Cindy Biru	A000757	102500	RSL	8
249	CIN-CC	T R Cindy Coklat C...	A0005705	102500	RSL	8
250	CIN-CK	T R Cindy Coklat	A0004283	102500	RSL	8

Query executed successfully. CHIBN\SQLEXPRESS (12.0 RTM) CHIBN\Herry Raditya Wi... dbToko 00:00:02 2225 rows

Gambar 7-5. Baris data yang didapat dari SQL SELECT

Saat ini, baris data seperti dalam contoh di atas telah disimpan dalam variable dataset **ds**.

Jika Anda perhatikan tampilan form, maka di bagian bawah ada dua buah **TextBox**, yaitu **tCurRow** dan **tMaxRow**. Apa gunanya?

Form harus mengetahui posisi user pada suatu saat. Posisi di sini dimaksudkan dengan posisi baris data. Ingat bahwa Anda menginginkan user bisa mengubah posisi data yang ditampilkan, dengan menekan tombol **Previous** atau **Next**? Bagaimana jika saat itu user sudah berada pada tampilan data terakhir, namun tetap menekan tombol **Next**? Atau pada data pertama, dan menekan tombol **Previous**?

Karena itu, kedua **TextBox** di bawah tersebut digunakan untuk menuliskan posisi baris data saat itu (**tCurRow**), sedangkan **tMaxRow** adalah sebuah konstan yang berisi jumlah data. Karena jumlah data dalam form ini selalu tetap (form tidak memiliki fitur untuk menghapus, menambah ataupun mengubah data), maka **TextBox** ini diatur supaya bersifat **ReadOnly** (juga untuk **TextBox** lain di dalam **GroupBox**).

Tambahkan deklarasi tiga buah variable:

```

Dim inc As Integer
Dim maxRow As Integer
Dim curRow As Integer

```

Sedangkan untuk pengisian data, sisipan baris program berikut setelah proses pengisian *dataset* ds.

```
maxRow = ds.Tables("Barang").Rows.Count  
inc = 0  
curRow = inc
```

Variable **maxRow** mendapatkan nilai awal langsung dari dataset, yaitu dengan method **Count()** dari object baris data dari table **Barang**.

Pada saat permulaan form dibuka, diberikan nilai awal dari variable **inc** ini sebagai **0**. Variable **inc** ini, bersama dengan variable **curRow**, akan menunjukkan posisi baris data saat itu, yaitu pada baris data pertama.

Selanjutnya adalah penulisan baris data yang telah didapat tersebut dalam tampilan form.

Untuk itu, Anda akan menuliskan sebuah sub-routine baru.

```
PrivateSub Navigasi(curRow AsInteger)  
Dim dsTable AsDataTable = ds.Tables("Barang")  
    tBrg_id.Text = dsTable.Rows(curRow).Item(0)  
    tBrg_nama.Text = dsTable.Rows(curRow).Item(1)  
    tBrg_unit.Text = dsTable.Rows(curRow).Item(2)  
    tBrg_barcode.Text = dsTable.Rows(curRow).Item(3)  
    tBrg_hpp.Text = FormatNumber(dsTable.Rows(curRow).Item(4), 0)  
    tKat_nama.Text = dsTable.Rows(curRow).Item(5)  
    tBrg_det_stock.Text = _  
        FormatNumber(dsTable.Rows(curRow).Item(6), 0)  
    tCurRow.Text = curRow + 1  
    tMaxRow.Text = maxRow  
EndSub
```

Variable **dsTable** yang dideklarasikan di atas bersifat opsional, dan hanya digunakan untuk menyingkat penulisan pada baris di bawahnya. Tanpa penggunaan variable ini program tetap akan bisa berjalan, namun mengalami sedikit perubahan pada baris programnya, seperti misalnya baris untuk menuliskan **TextBrg_id** akan menjadi sebagai berikut:

```
tBrg_id.Text = ds.Tables("Barang").Rows(curRow).Item(0)
```

Seperti yang bisa Anda lihat, penulisan menjadi lebih panjang

Hal lain yang perlu Anda perhatikan adalah pada penulisan **TextBrg-CurRow**.

```
tCurRow.Text = curRow + 1
```

Di sini nilai **tCurRow** diambil dari variable **curRow**, dengan ditambah dengan nilai **1**. Hal ini disebabkan karena, jika Anda ingat, pengurutan nilai baris data adalah dimulai dari baris **0**. Jadi, jika ada **10** data, maka baris data pertama adalah baris **0**, dan baris terakhir adalah baris **9**.

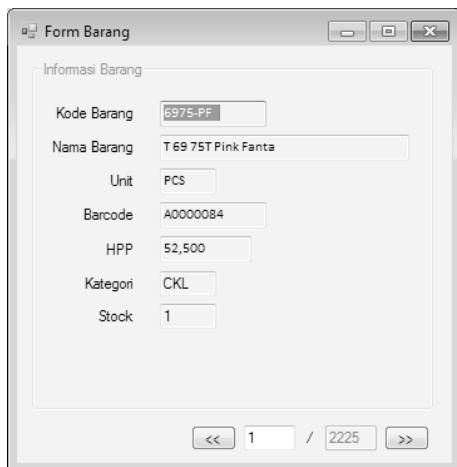
Sekarang, panggilah sub-routine tersebut, sebagai awal, letakkan pada bagian terakhir dari sub-routine untuk event **Form_Load**.

```
da = New OleDb.OleDbDataAdapter(Sql, con)
da.Fill(ds, "Barang")
maxRow = ds.Tables("Barang").Rows.Count
inc = 0
curRow = inc

con.Close()
Navigasi(curRow) ←
End Sub
```

Gambar 7-6. Pemanggilan sub-routine Navigasi()

Jika program ini dijalankan, maka tampilan akan serupa dengan gambar di bawah ini:



Gambar 7-7. Tampilan awal form Barang

Perhatikan bahwa karena sebagai awal Anda berikan **curRow = inc = 0**, berarti adalah bahwa data pertama yang akan dituliskan dalam form.

Langkah selanjutnya adalah untuk memproses penekanan tombol, maju untuk **btnNext** dan mundur untuk **btnPrev**.

Kembalilah ke tampilan form, dan lakukan klik-ganda pada tombol **btnPrev**.

Pada tampilan program, tuliskan baris program berikut:

```

PrivateSub btnPrev_Click(sender AsObject, e AsEventArgs)
Handles btnPrev.Click
If inc <>0Then
    inc -= 1
    Navigasi(inc)
EndIf
EndSub

```

Jadi, pada saat tombol tersebut ditekan, maka form akan memeriksa posisi baris data saat itu. Apabila data saat itu berada pada baris **0**, yang berarti form baru pertama dibuka, atau data yang ditampilkan sudah merupakan data baris pertama, yang berarti user tidak akan mendapatkan apa pun dengan penekanan tombol tersebut.

Demikian juga untuk tombol yang lain, yaitu tombol **btnNext**. Tombol ini digunakan untuk memajukan posisi baris data saat itu maju ke depan, selama posisi saat itu belum berada pada baris data terakhir.

```

PrivateSub btnNext_Click(sender AsObject, e AsEventArgs)
Handles btnNext.Click
If inc <> maxRow - 1 Then
    inc += 1
    Navigasi(inc)
EndIf
EndSub

```

Hal terakhir yang masih diperlukan adalah apabila user menuliskan nilai baris yang akan dituju pada **TextBoxtCurRow**.

Untuk itu, Anda akan memakai event **LostFocus** untuk kontrol **tCurRow** tersebut.

Isi dalam sub-routine ini cukup singkat:

```

PrivateSub tCurRow_LostFocus(sender AsObject, e AsEventArgs)
Handles tCurRow.LostFocus
    inc = CInt(tCurRow.Text) - 1
    Navigasi(inc)
EndSub

```

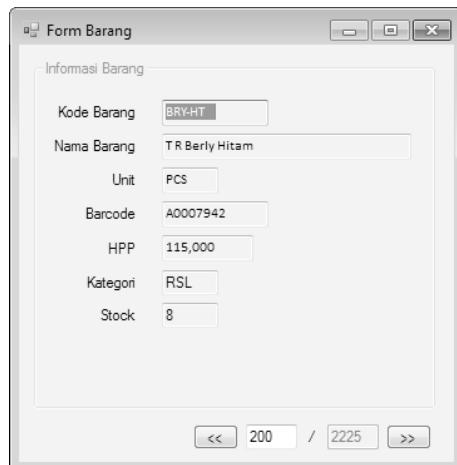
Jadi, setelah user mengisi **TextBox** tersebut, dan berpindah focus ke bagian lain dari form, maka event ini akan dijalankan.

Proses yang dijalankan juga cukup sederhana, yaitu memastikan bahwa data yang dituliskan user adalah data Numerik (tidak ada sub-routine untuk error checking, hanya digunakan fungsi **Cint()** untuk mengubah masukan **String** menjadi **Integer**). Setelah diubah menjadi numerik, nilai ditambah satu (sekali lagi, karena data pertama dimulai dari baris **0**), dan refresh tampilan dengan memanggil sub-routine **Navigasi()**.

Selesailah sudah proses pembuatan form ini. Form ini hanya memiliki satu tampilan, perbedaan hanya didasarkan pada perubahan data yang

ditampilkan. Tidak ada bagian form lain yang akan berubah, karena memang sifat dari interaksi form ini adalah hanya untuk display barang, dan bukan untuk melakukan manipulasi data.

Tampilan program ini akan sebagai berikut:



Gambar 7-8. Tampilan frmBarang1

###

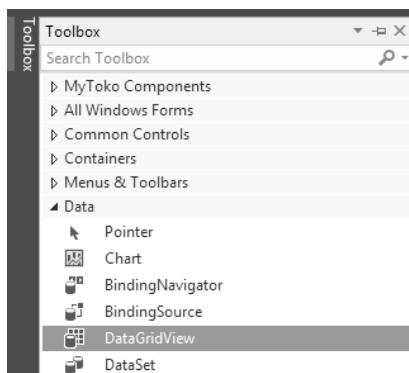
BAB 8

Form Barang

Dalam bab ini, Anda akan membuat satu lagi form Barang, namun form ini akan menggunakan format tampilan **DataGridView**, dan akan mengizinkan user untuk melakukan manipulasi data.

Tambahkanlah sebuah **Windows Forms** lagi pada Project Anda, dan berikan nama **frmBarang**. Kali ini tanpa angka **1** di belakang. Setelah itu, ubahlah setting pada item **My Project**, dan ubah **Startup form** untuk mengacu pada form baru ini.

Kemudian, tambahkanlah object**DataGridView** pada form Anda. Object diambil dari Toolbox, pada bagian Data.



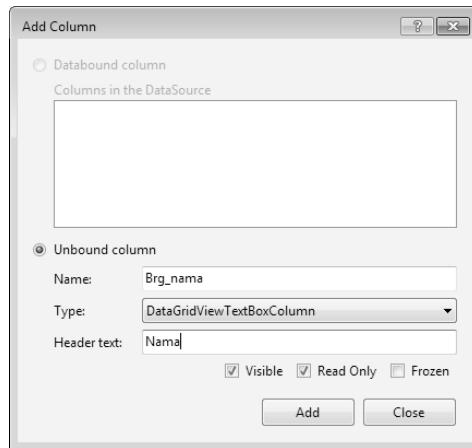
Gambar 8-1. Object Data - DataGridView

Setelah Anda tambahkan, ubahlah beberapa property-nya seperti dibawah ini:

```
Name = BarangDGV  
AllowUserToAddRow = False  
AllowUserToDeleteRow = False
```

```
AllowUserToOrderColumns = False  
AllowUserToResizeColumns = False  
AllowUserToResizeRows = False  
RowHeadersWidth = 25  
SelectionMode = FullRowSelect
```

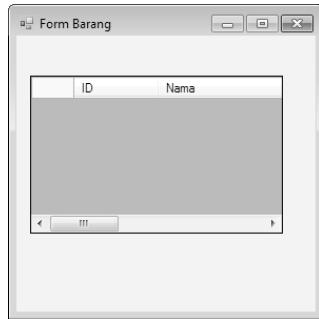
Setelah itu, tekanlah pada tanda anak panah kecil di sebelah kanan-atas dari **BarangDGV**, dan dari context-menu yang ditampilkan, pilihlah opsi untuk **Add Column ...**



Gambar 8-3. Add Column

Di sini, tambahkanlah tujuh buah column, dengan data sebagai berikut:

```
Name/Header text 1: Brg_id / ID  
Name/Header text 2: Brg_nama / Nama  
Name/Header text 3: Brg_unit / Unit  
Name/Header text 4: Brg_barcode / Barcode  
Name/Header text 5: Brg_hpp / HPP  
Name/Header text 6: Kat_nama / Kat  
Name/Header text 7: Brg_det_stock / Stock
```



Gambar 8-4. Tampilan Barang DGV

Setelah Anda tambahkan semua column itu, tekanlah sekali lagi tanda anak panah kecil itu, dan kemudian pilih opsi **Edit Columns ...**

Sekarang, Anda akan mengatur lebar kolom, agar form Anda lebih enak dilihat. Untuk setiap column, ubahlah property **Width**-nya, sesuai dengan data berikut:

ID, Width = 80

Nama, Width = 200

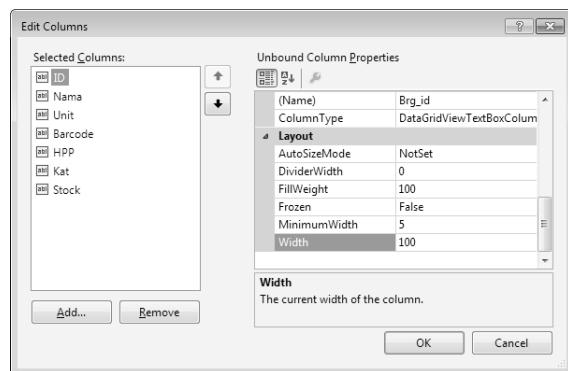
Unit, Width = 50

Barcode, Width = 80

HPP, Width = 100

Kat, Width = 50

Stock, Width = 40



Gambar 8-5. Property Width untuk semua Columns

Setelah itu, perbaikilah tampilan form Anda. Buat lebar yang cukup agar semua column bisa ditampilkan, tanpa menggunakan horizontal scroll-bar.



Gambar 8-6. Tampilan Form dengan object DataGridView

Setelah Anda selesaikan tampilan object **DataGridView** tersebut, sekarang marilah coba untuk memperlihatkan data barang pada object tersebut.

Pindahlah ke tampilan program, dan terlebih dulu tuliskan baris berikut untuk menggunakan beberapa Library yang ada.

```
Imports System
Imports System.Drawing
Imports System.Windows.Forms
Imports System.Data.OleDb
```

Setelah itu, deklarasikan beberapa variable untuk koneksi ke database.

```
Dim con As New OleDbConnection
Dim dbProvider As String
Dim dbSource As String
Dim ds As New DataSet
Dim da As OleDbDataAdapter
```

Hampir seperti form Barang sebelumnya, di sini Anda juga akan menggunakan event **Form_Load** dan **Form_Leave**. Ada hanya sedikit perbedaan antar kedua cara display ini.

```
Private Sub frmBarang_Leave(sender As Object, e As EventArgs)
Handles Me.Leave
    con = Nothing
    ds = Nothing
    da = Nothing
End Sub
```

```

PrivateSub frmBarang_Load(sender AsObject, e AsEventArgs)
Handles MyBase.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial "&
"Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource
    con.Open()

    da = New OleDb.OleDbDataAdapter("SELECT Barang.Brg_id,
Barang.Brg_nama, Barang.Brg_unit, Barang.Brg_barcode,
Barang.Brg_hpp, Barang.Kat_id, Kategori.Kat_nama,
Barang_details.brg_det_stock FROM Barang, Kategori,
Barang_details WHERE Barang.Kat_id = Kategori.Kat_id and
Barang.Brg_id = Barang_details.Brg_id", con)
    da.Fill(ds, "Barang_All")

    con.Close()

EndSub

```

Untuk menampilkan baris data pada display untuk form Barang sebelum ini, digunakan sub-routine **Navigasi()**. Untuk form ini pun menggunakan cara yang sama, namun untuk mempermudah, nama sub-routine akan menggunakan nama **setupDGV()**.

```

PrivateSub setupDGV()
Dim i AsInteger

    con.Open()
    da = New OleDb.OleDbDataAdapter("SELECT Barang.Brg_id,
Barang.Brg_nama, Barang.Brg_unit, Barang.Brg_barcode,
Barang.Brg_hpp, Barang.Kat_id, Kategori.Kat_nama,
Barang_details.brg_det_stock FROM Barang, Kategori,
Barang_details WHERE Barang.Kat_id = Kategori.Kat_id and
Barang.Brg_id = Barang_details.Brg_id", con)
    da.Fill(ds, "Barang_All")
    con.Close()

Dim dsTables AsDataTable = ds.Tables("Barang_All")
    BarangDGV.Rows.Clear()
For i = 0 To dsTables.Rows.Count - 1
    BarangDGV.Rows.Add()
    BarangDGV.Rows(i).Cells(0).Value = _
dsTables.Rows(i).Item("Brg_id")
    BarangDGV.Rows(i).Cells(1).Value = _
dsTables.Rows(i).Item("Brg_nama")
    BarangDGV.Rows(i).Cells(2).Value = _
dsTables.Rows(i).Item("Brg_unit")
    BarangDGV.Rows(i).Cells(3).Value = _
dsTables.Rows(i).Item("Brg_barcode")
    BarangDGV.Rows(i).Cells(4).Value = _
FormatNumber(dsTables.Rows(i).Item("Brg_hpp"), 0)
    BarangDGV.Rows(i).Cells(5).Value = _
dsTables.Rows(i).Item("Kat_nama")
    BarangDGV.Rows(i).Cells(6).Value = _
dsTables.Rows(i).Item("Brg_det_stock")
Next
End Sub

```

Seperti pada form Barang sebelumnya, letakkan statement pemanggil sub-routine ini pada bagian akhir dari sub-routine **Form_Load**.

```
da.Fill(ds, "Barang_All")  
con.Close()  
setupDGV()  
End Sub
```



Gambar 8-7. Letak pemanggil

Anda bisa melihat cara kerja dari sub-routine ini cukup sederhana. Langkah pertama, seperti sebelumnya, adalah mengisikan variable dataset **ds** dengan hasil dari query **SQL SELECT** dengan tiga table dasar.

Setelah variable **ds** terisi, kemudian form akan membersihkan object **BarangDGV** dari nilai sebelumnya, sehingga dimulai lagi dari Data Grid yang kosong.

Setelah itu, akan masuk ke dalam proses Looping, mulai dari data pertama sampai dengan data terakhir.

Untuk menambah baris baru dalam object **BarangDGV**, digunakan perintah

```
BarangDGV.Rows.Add()
```

Hal ini akan membuat sebuah baris baru, namun masih belum memiliki data apa pun, atau masih kosong.

Baris-baris setelah itu akan mengisi data kosong tersebut dengan data yang diambil dari dataset **ds**.

```
BarangDGV.Rows(i).Cells(0).Value = -  
dsTables.Rows(i).Item("Brg_id")  
BarangDGV.Rows(i).Cells(1).Value = -  
dsTables.Rows(i).Item("Brg_nama")
```

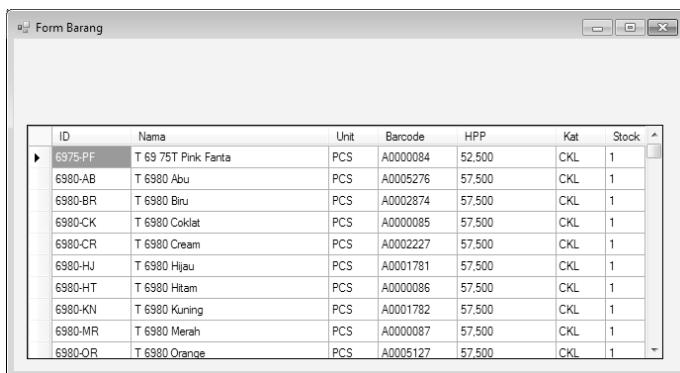
Dalam contoh di sini, baris ke-**i** (pada saat mulai Looping, nilai **i = 0**), akan diisi nilai dari Cell ke-**0** (kolom pertama dari baris pertama) dengan nilai dari dataset **ds**, baris yang pertama juga, dan diambil dari kolom dengan nama "**Brg_id**".

Begini juga seterusnya untuk baris-baris selanjutnya.

Khusus untuk kolom ke-**5**, yaitu kolom **HPP**, digunakan fungsi **FORMAT()** pada nilai yang akan dituliskan, untuk mengatur tampilan data menjadi tampilan numerik dengan pembagian ribuan.

```
FormatNumber(dsTables.Rows(i).Item("Brg_hpp"), 0)
```

Jika kemudian Anda menjalankan form ini, maka tampilan yang akan Anda dapatkan adalah sebagai berikut.



Gambar 8-8. Tampilang Barang DGV dengan isinya

Sampai di sini, Anda sudah memiliki form yang bisa menampilkan data barang, dan Anda bisa memilih bagian manapun dari form ini, meskipun belum ada yang bisa Anda lakukan di sini.

Untuk memberikan suatu fitur penambahan maupun pengubahan data, Anda akan belajar untuk menggunakan suatu object baru, yang disebut **Panel**.

Ambillah object **Panel** dari bagian **Containers** pada Toolbox Anda, dan letakkan di atas form Anda. Berilah nama **Panel** tersebut dengan nama **pnBarang**. Secara khusus, ubahlah property **Visible** untuk **Panel** ini dengan nilai **False**. Dengan demikian, pada saat form ini dijalankan, **Panel** ini akan tetap tersembunyi.

Setelah itu, tambahkanlah 7 buah **Label**, 6 buah **TextBox**, 1 buah **ComboBox**, dan 2 buah **Button** ke dalam **pnBarang** tersebut. Dengan demikian, apabila **pnBarang** masih dalam keadaan tersembunyi, semua kontrol di dalamnya juga akan tersembunyi.

Aturlah posisinya supaya terlihat rapi di dalam **pnBarang** Anda. Kedua tombol yang ditambahkan diletakkan di bagian bawah dari **pnBarang**, rata di tengah.

Selain itu, tambahkan juga sebuah tombol di luar **pnBarang**, di sisi atas dari **BarangDGV**.

Atur property dari kontrol-kontrol tersebut sebagai berikut:

- A: Name = txtBrg_id
- B: Name = txtBrg_nama
- C: Name = txtBrg_unit
- D: Name = txtBrg_barcode
- E: Name = txtBrg_hpp
- F: Name = cbKat_id
- G: Name = txtBrg_det_stock
- 1: Name = btnBatal
- 2: Name = btnTambah
- 3: Name = btnBrgBaru

Lakukan klik-ganda pada **btnBrgBaru**, dan tuliskan baris program berikut untuk event **Click**-nya.

```
PrivateSub btnBrgBaru_Click(sender AsObject, e AsEventArgs)
Handles btnBrgBaru.Click
    pnBarang.Visible = Not (pnBarang.Visible)
EndSub
```

Tugas dari tombol ini hanya akan mengubah status **Visible** dari **pnBarang**. Jika saat ditekan **pnBarang** tersembunyi, maka ditampilkan. Jika saat ditekan **pnBarang** terlihat, maka disembunyikan.

Lakukan juga untuk **btnBatal**, dan tuliskan baris berikut.

```
PrivateSub btnBatal_Click(sender AsObject, e AsEventArgs)
Handles btnBatal.Click
    pnBarang.Visible = False
EndSub
```

Jadi, jika tombol **btnBatal** ditekan, **pnBarang** juga bisa disembunyikan kembali.

Jalankan program, dan mainkan tombol yang ada.

	Unit	Barcode	HPP	Kat	Stock
PCS	A0000084	52.500	CKL	1	
PCS	A0005276	57.500	CKL	1	
PCS	A0002874	57.500	CKL	1	
PCS	A0000085	57.500	CKL	1	
PCS	A0002227	57.500	CKL	1	
PCS	A0001781	57.500	CKL	1	
PCS	A0000086	57.500	CKL	1	
PCS	A0001782	57.500	CKL	1	
PCS	A0000087	57.500	CKL	1	
PCS	A0005127	57.500	CKL	1	

Gambar 8-11. pnBarang terlihat di atas BarangDGV

Jika Anda lihat pada panel tersebut, ada salah satu yang berupa **ComboBox**, yaitu untuk **Kategori**. Dalam data barang, kategori dituliskan sebagai **ID**-nya saja, yaitu angka urut. Namun dalam form ini, data akan dituliskan sebagai namanya, sehingga lebih jelas bisa dilihat oleh user.

Kategori diambil juga dari sebuah table bernama table **Kategori** di dalam database yang sama. Anda perhatikan bahwa di bagian awal form ini dibuka, yaitu pada event **Form_Load**, form sudah membuka koneksi pada database, dan khususnya pada sebuah query yang mengambil data dari tiga buah table. Query tersebut disimpan dalam sebuah variable DataAdapter **da**. Yang harus diingat di sini, apabila Anda hendak menulis ke dalam database, Anda tidak bisa menggunakan variable DataAdapter yang sama untuk table yang berbeda. Karena itu, DataAdapter untuk table **Kategori** harus dideklarasikan secara terpisah. Sedangkan untuk dataset, masih bisa digunakan secara bersamaan.

```
Dim da, daKat As OleDbDataAdapter
```

Dan untuk deklarasi **DataAdapter**-nya sendiri.

```
daKat = _
New OleDb.OleDbDataAdapter("SELECT * FROM Kategori", con)
daKat.Fill(ds, "Kategori")
```

Kemudian, untuk mengisi **ComboBox** tersebut, Anda bisa menuliskan perintahnya pada sub-routine **setupDGV()**, karena **ComboBox** ini berada di dalam object panel yang akan ditampilkan oleh object **BarangDGV**.

Caranya adalah dengan membuat sebuah object **DataTable** baru, yang akan berisi dua buah field saja, yaitu **Kat_nama** dan **Kat_id**. Setelah itu, tambahkan data-data **Kategori** yang sudah diambil dan disimpan dalam variable dataset **ds**.

Object **DataTable** yang sudah terisi data **Kategori** itulah yang kemudian dijadikan sebagai **DataSource** dari item **ComboBox** itu sendiri.

Baris programnya sendiri adalah sebagai berikut:

```
Dim table AsNewDataTable  
table.Columns.Add("Kat_nama", GetType(String))  
table.Columns.Add("Kat_id", GetType(Integer))
```

Baris ini akan membuat object **DataTable**, dan memberikan dua buah column, yang akan memiliki tipe data **String** dan **Integer**.

Kemudian masuk ke dalam looping untuk setiap baris data **Kategori**, dan memasukkan pasangan data tersebut ke dalam object **DataTable**.

Untuk memasukkan ke dalam table, digunakan sebuah sub-routine kecil bernama **addrow()**.

```
PrivateSub addrow(table AsDataTable, keyAsString, value  
AsInteger)  
Dim row AsDataRow = table.NewRow  
    row("Kat_nama") = key  
    row("Kat_id") = value  
    table.Rows.Add(row)  
EndSub
```

Seperti pada pengisian **DataGridView**, untuk menambahkan item baris baru, harus dideklarasikan dulu variable-nya, dan diberikan nilai awal dengan method **NewRow**. Hal ini akan membuat sebuah baris baru pada table, namun dengan isi masih kosong.

Baris berikutnya yang akan memberikan nilai pada baris baru tersebut dengan nilai yang dijadikan parameter untuk sub-routine ini.

Kembali pada proses looping-nya, perintah untuk memanggil sub-routine **addrow()** tersebut adalah sebagai berikut:

```
For i = 0 To ds.Tables("Kategori").Rows.Count - 1  
    addrow(table, _  
        ds.Tables("Kategori").Rows(i).Item("Kat_nama"), _  
        ds.Tables("Kategori").Rows(i).Item("Kat_id"))  
Next
```

Langkah terakhir adalah mengisikan table tersebut sebagai sumber data dari item **ComboBox** itu sendiri.

```
cbKat_id.DataSource = table  
cbKat_id.DisplayMember = "Kat_nama"
```

```
cbKat_id.ValueMember = "Kat_id"
table = Nothing
```

Demikianlah cara untuk mengambil data dari table yang berbeda, dengan pemisahan DataAdapter. Jangan lupa untuk menghapus memory yang digunakan oleh variable DataAdapter baru tersebut dengan memberikan padanya nilai **Nothing** pada event **Form_Leave**.

Langkah berikutnya adalah membuat sub-routine untuk menangani event **VisibleChanges()** dari object **pnBarang** ini.

Event ini akan dijalankan pada saat status property **Visible** dari panel **pnBarang** berubah. Apakah berubah menjadi **True** atau **False**, namun jika berubah menjadi **False**, maka event ini tidak akan melakukan apa pun.

Buatlah sub-routine untuk event tersebut, dan tuliskan isinya. Berilah suatu kondisi yang akan memeriksa nilai dari property **Visible** **pnBarang** itu pada saat dijalankan.

```
PrivateSub pnBarang_VisibleChanged(sender AsObject, e
AsEventArgs) Handles pnBarang.VisibleChanged
If pnBarang.Visible Then
    txtBrg_id.Clear()
    txtBrg_nama.Clear()
    txtBrg_unit.Clear()
    txtBrg_barcode.Clear()
    txtBrg_hpp.Clear()
    cbKat_id.SelectedIndex = 0
    txtBrg_det_stock.Clear()
    btnTambah.Text = "Tambah"
EndIf
EndSub
```

Sub-routine ini akan membersihkan nilai-nilai yang ada di dalam panel **pnBarang** tersebut, pada saat panel dibuka.

Jika Anda jalankan, dan perlihatkan panel **pnBarang**, maka **ComboBox** di dalamnya akan sudah berisi data dari table **Kategori**.

ID	Unit	Barcode	HPP	Kat	Stock
6975-	PCS	A0000084	52.500	CKL	1
6980-	PCS	A0005276	57.500	CKL	1
6980-	PCS	A0002874	57.500	CKL	1
6980-	PCS	A0000085	57.500	CKL	1
6980-	PCS	A0002227	57.500	CKL	1
6980-	PCS	A0001781	57.500	CKL	1
6980-	PCS	A0000086	57.500	CKL	1
6980-	PCS	A0001782	57.500	CKL	1
6980-	PCS	A0000087	57.500	CKL	1
6980-OR	PCS	A0005127	57.500	CKL	1

Gambar 8-12. ComboBox sudah terisi data Kategori

Berikutnya, akan dibahas cara membaca posisi baris yang mendapatkan klik-ganda dari mouse. Tugas dari form ini nantinya adalah memperbolehkan user menambah barang baru dengan cara menekan tombol **Tambah** (tombol **btnBrgBaru**). Tapi selain itu, form ini juga memperbolehkan user untuk mengubah data suatu barang. Caranya adalah dengan melakukan klik-ganda pada baris barang yang hendak di-update.

Jadi, sebelum sampai tahap pengubahan itu sendiri, sekarang Anda akan belajar bagaimana cari mendapatkan posisi baris yang mendapatkan klik-ganda, kemudian menuliskan isi data dari baris tersebut ke dalam isian panel **pnBarang**.

Sebelumnya, deklarasikanlah sebuah variable yang akan menampung nomor baris barang saat itu.

```
Dim brgBaris As Integer
```

Kemudian, buatlah event **CellDoubleClick** untuk object **BarangDGV**, dan tuliskan baris program berikut.

```
PrivateSub BarangDGV_CellDoubleClick(sender AsObject, e AsDataGridViewCellEventArgs) Handles BarangDGV.CellDoubleClick
Dim dsTables AsDataTable = ds.Tables("Kategori")
pnBarang.Visible = Not (pnBarang.Visible)
With BarangDGV.Rows(e.RowIndex)
    txtBrg_id.Text = Trim(.Cells(0).Value)
    txtBrg_nama.Text = Trim(.Cells(1).Value)
    txtBrg_unit.Text = Trim(.Cells(2).Value)
    txtBrg_barcode.Text = Trim(.Cells(3).Value)
    txtBrg_hpp.Text = .Cells(4).Value
For i AsInteger = 0 To dsTables.Rows.Count - 1
If dsTables.Rows(i).Item("Kat_nama") = _
.Cells(5).Value Then
    cbKat_id.SelectedIndex = _
```

```

dsTables.Rows(i).Item("Kat_id") - 1
EndIf
Next
EndWith
txtBrg_det_stock.Text = _
ds.Tables("Barang_All").Rows(e.RowIndex).Item("Brg_det_stock")

brgBaris = e.RowIndex
btnTambah.Text = "Ubah"
EndSub

```

Jadi, untuk mendapatkan pada baris berapa object **DataGridView** mendapatkan klik-ganda, dipakai object **e.RowIndex**. Pada deklarasi subroutine ini, parameter **e** dideklarasikan sebagai object **DataGridViewCellEventArgs**. Dan dalam object ini ada sebuah property dengan nama **RowIndex**, yang menyimpan nilai baris saat mendapatkan event double-click.

Anda kemudian bisa menggunakan nilai baris tersebut dengan langsung mengisikan data dari dalam **BarangDGV** ke dalam item-item dalam panel **pnBarang**.

Khusus untuk dua item, yaitu **ComboBox** Nama Kategori dan **TextBox** Status Stock, penulisan agak berbeda, karena memang data tersebut diambil dari table yang berbeda.

Jika Anda perhatikan, Anda akan melihat bahwa tombol pada panel **pnBarang** bisa berubah antara **Tambah** dengan **Ubah**. Perubahan ini berdasarkan pada apa yang memanggil panel itu sendiri. Jika panel dibuka karena user menekan tombol **btnBrgBaru**, maka tombol dalam panel akan menjadi **Tambah**, karena itu berarti user akan menambahkan barang baru.

Namun, apabila panel itu ditampilkan karena user melakukan klik-ganda pada salah satu baris data dalam **BarangDGV**, maka baris untuk barang yang dipilih itu yang akan ditampilkan dalam panel, sehingga tombol dalam panel akan menjadi **Ubah**.

Kedua status tombol itu sendiri yang akan digunakan menjadi kondisi penentu antara perintah untuk **INSERT** data barang, atau untuk **UPDATE** data barang.

Sekarang, tiba saatnya untuk proses utama pada form ini, yaitu pada saat user menekan tombol **btnTambah** yang berada di dalam panel **pnBarang**.

Sebelumnya, persiapkan variable **DataAdapter** untuk kedua table yang lain, yaitu table **Barang** dan table **Barang_details**.

```
Dim da, daKat, daBrg, daBrg_Det As OleDbDataAdapter
```

Kemudian, buatlah bagian inisiasi variable dengan querySQL masing-masing.

```
daBrg = New OleDb.OleDbDataAdapter("SELECT Brg_id, Brg_nama,  
Brg_unit, Brg_barcode, Brg_hpp, Kat_id FROM Barang", con)  
daBrg.Fill(ds, "Barang")  
  
daBrg_Det = New OleDb.OleDbDataAdapter("SELECT * FROM  
Barang_details", con)  
daBrg_Det.Fill(ds, "Barang_details")
```

Anda bisa lihat di sini, meskipun variable **DataAdapter**-nya berbeda, namun hasil query tetap disimpan dalam variable dataset yang sama, yaitu variable **ds**. Dengan ini, variable **ds** memiliki empat buah **Tables**, yaitu **ds.Tables("Barang_All")**, **ds.Tables("Barang")**, **ds.Tables("Barang_details")** dan **ds.Tables("Kategori")**.

Setelah itu, persiapkan juga proses penghapusan data dalam variable **ds**, khususnya untuk table "**Barang_All**". Hal ini disebabkan karena table ini menjadi sumber data untuk **BarangDGV**, sehingga jika terjadi refresh tampilan (dengan melakukan penulisan ulang pada object **DataGridView**), isi dari **DataGridView** itu bisa berulang jika tidak dihapus terlebih dahulu.

Karena itu, letakkan proses penghapusan ini di bagian atas sub-routine **setupDGV0**, sebelum deklarasi ulang variable **da**.

```
If ds.Tables("Barang_All").Rows.Count > 0 Then  
    ds.Tables("Barang_All").Clear()  
EndIf
```

Sekarang, mulai masuk pada sub-routine utama. Lakukan klik-ganda pada tombol **btnTambah** untuk mulai menulis programnya.

Di bagian atas, deklarasikan dua buah variable tipe **DataRow**, untuk menampung baris data untuk masing-masing table, **Barang** dan **Barang_details**. Kemudian, lakukan error-checking sederhana, yang akan memeriksa nilai kosong pada **TextBox**.

```
PrivateSub btnTambah_Click(sender AsObject, e AsEventArgs)  
Handles btnTambah.Click  
Dim BrgBaru, BrgBaruStock AsDataRow  
If txtBrg_id.Text = ""Then  
    MsgBox("ID harus diisi.")  
    txtBrg_id.Focus()  
Elseif txtBrg_nama.Text = ""Then  
    MsgBox("Nama harus diisi.")  
    txtBrg_nama.Focus()  
Elseif txtBrg_unit.Text = ""Then  
    MsgBox("Unit harus diisi.")
```

```

    txtBrg_unit.Focus()
ElseIf txtBrg_barcode.Text = ""Then
    MsgBox("Barcode harus diisi.")
    txtBrg_barcode.Focus()
ElseIf txtBrg_hpp.Text = ""Then
    MsgBox("HPP harus diisi.")
    txtBrg_hpp.Focus()
ElseIf txtBrg_det_stock.Text = ""Then
    MsgBox("Jumlah stock harus diisi.")
    txtBrg_det_stock.Focus()
Else

```

Secara sederhana, apabila **TextBox** kosong, maka sebuah msgbox akan memberitahu user tentang itu, dan kemudian mengembalikan focus pada **TextBox** tersebut.

Jika semua error-checking tersebut terlewati, maka dimulai proses manipulasi datanya.

Awalnya, sekali lagi mendeklarasikan dua buah variable baru, satu untuk masing-masing table, sebagai sebuah object **OleDbCommandBuilder()**. Object ini dibutuhkan untuk memperbolehkan program Anda melakukan update terhadap table di dalam database. Tanpa deklarasi variable ini, data Anda tidak akan tersimpan.

Selain itu, di sini juga ditempatkan pemeriksaan kondisi, untuk memisahkan tindakan yang dilakukan, antara proses **Tambah** dengan proses **Ubah**.

```

Else
Dim cbBrg AsNew OleDb.OleDbCommandBuilder(daBrg)
Dim cbBrg_det AsNew OleDb.OleDbCommandBuilder(daBrg_det)
If btnTambah.Text = "Tambah"Then
    .
    .
    .
ElseIf btnTambah.Text = "Ubah"Then
    .
    .
    .
EndIf

daBrg.Update(ds, "Barang")
daBrg_det.Update(ds, "Barang_details")
pnBarang.Visible = False
setupDGV()
EndIf

EndSub

```

Setiap penambahan data pada table **Barang**, barang yang sama juga harus dimasukkan pada table **Barang_details**.

Setelah proses penambahan ataupun pengubahan data dijalankan terhadap dataset, maka seperti yang Anda lihat dalam listing di atas, dilakukan juga proses **Update** terhadap **DataAdapter** masing-masing.

Setelah proses itu selesai, kembali sembunyikan panel **pnBarang**, dan refresh tampilan dengan menjalankan sub-routine **setupDGV()**.

Untuk proses penambahan barang sendiri, tuliskan program berikut pada tempatnya.

```
BrgBaru = ds.Tables("Barang").NewRow
BrgBaruStock = ds.Tables("Barang_details").NewRow
BrgBaru("Brg_id") = Trim(txtBrg_id.Text)
BrgBaru("Brg_nama") = Trim(txtBrg_nama.Text)
BrgBaru("Brg_unit") = Trim(txtBrg_unit.Text)
BrgBaru("Brg_barcode") = Trim(txtBrg_barcode.Text)
BrgBaru("Brg_hpp") = txtBrg_hpp.Text
BrgBaru("Kat_id") = cbKat_id.SelectedValue
BrgBaruStock("Brg_id") = Trim(txtBrg_id.Text)
BrgBaruStock("Brg_det_stock") = txtBrg_det_stock.Text
ds.Tables("Barang").Rows.Add(BrgBaru)
ds.Tables("Barang_details").Rows.Add(BrgBaruStock)
```

Pertama-tama, dibuat sebuah baris record baru untuk setiap table-nya, dan disimpan dalam variable **DataRowBrgBaru** dan **BrgBaruStock**.

Kemudian, dilakukan proses penyimpanan data, diawali dengan variable **BrgBaru**, dan kemudian dilanjutkan dengan **BrgBaruStock**.

Setelah semua column terisi data (karena aturan database tidak memperbolehkan nilai **NULL**), dilakukan proses penambahan data, dari kedua variable **DataRow** tadi ke dalam dataset, untuk masing-masing table.

Demikian juga untuk proses pengubahan data.

```
ds.Tables("Barang").Rows(brgBaris).Item("Brg_id") = _
    Trim(txtBrg_id.Text)
ds.Tables("Barang").Rows(brgBaris).Item("Brg_nama") = _
    Trim(txtBrg_nama.Text)
ds.Tables("Barang").Rows(brgBaris).Item("Brg_unit") = _
    Trim(txtBrg_unit.Text)
ds.Tables("Barang").Rows(brgBaris).Item("Brg_barcode") = _
    Trim(txtBrg_barcode.Text)
ds.Tables("Barang").Rows(brgBaris).Item("Brg_hpp") = _
    CInt(txtBrg_hpp.Text)
ds.Tables("Barang").Rows(brgBaris).Item("Kat_id") = _
    cbKat_id.SelectedValue
ds.Tables("Barang_details").Rows(brgBaris).Item("Brg_id") = _
    Trim(txtBrg_id.Text)
ds.Tables("Barang_details").Rows(brgBaris).Item_(
    ("Brg_det_stock") = txtBrg_det_stock.Text
```

Dalam proses pengubahan ini tidak diperlukan variable **DataRow**, namun langsung menuju pada baris yang dipilih.

Dalam tampilannya, pertama-tama akan dibuka default tampilan **DataGridview**.

ID	Nama	Unit	Barcode	HPP	Kat	Stock
ETA-HJ	T Etania Hijau	PCS	A0006239	80.000	SMPG...	10
ETA-HT	T Etania Hitam	PCS	A0006211	80.000	SMPG...	10
ETA-KN	T Etania Kuning	PCS	A0006405	80.000	SMPG...	10
ETA-UG	T Etania Ungu	PCS	A0006159	80.000	SMPG...	10
FF	Frame Rarel	PCS	A0001774	17.500	PP	7
FL	T Flow WW	PCS	A0000606	67.500	CKL	1
FLCT	T Felicita	PCS	A0003121	77.500	SMPG...	10
FMK-10R	FMK 10R	PCS	A0000013	13.800	DKRS ...	2
FMK-2R	FMK 2R	PCS	A0000014	4.025	DKRS ...	2
FMK-2RR	FMK 2R R	PCS	A0000015	11.500	DKRS ...	2

Gambar 8-13. Tampilan awal

Dan jika melakukan klik-ganda pada salah satu baris data:

ID	ETA-HJ	Unit	PCS	Barcode	A0006239	HPP	80.000	Kat	SMPG...	Stock	10
ETA-H	T Etania Hijau	PCS	A0006211								
ETA-K		PCS	A0006405								
ETA-L		PCS	A0006159								
FF		PCS	A0001774								
FL		PCS	A0000606								
FLCT		PCS	A0003121								
FMK-1		PCS	A0000013								
FMK-2		PCS	A0000014								
FMK-2RR	FMK 2R R	PCS	A0000015								

Gambar 8-14. Panel pengubahan data

Namun, jika tombol **Tambah** di atas ditekan, maka panel pengubahan tidak akan berisi apa pun, dan mengharapkan user yang akan mengisi datanya.

Form Barang

ID	<input type="text"/>	Tambah			
Nama	<input type="text"/>	Unit	<input type="text"/>	HPP	<input type="text"/>
Barcode	<input type="text"/>	PCS	A0001928	25,500	KID
HPP	<input type="text"/>	PCS	A0000084	52,500	CKL
Kategori	<input type="text"/> CKL	PCS	A0005276	57,500	CKL
Stock	<input type="text"/>	PCS	A0002874	57,500	CKL
	Batal	PCS	A0000085	57,500	CKL
	Tambah	PCS	A0002227	57,500	CKL
		PCS	A0001781	57,500	CKL
		PCS	A0000086	57,500	CKL
		PCS	A0001782	57,500	CKL
		PCS	A0000087	57,500	CKL

Gambar 8-15. Panel penambahan data

Demikianlah form **Barang** telah selesai. Form ini menjadi dasar untuk form-form data dasar lainnya, yang menggunakan teknik yang hampir sama untuk setiap table yang digunakan.

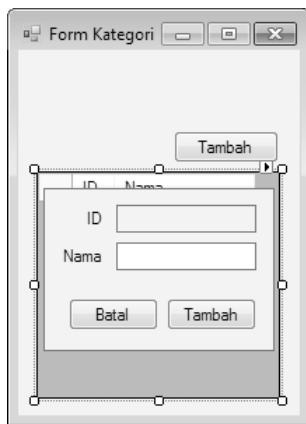
BAB 9

Form Kategori, User, dan Customer

Ketiga form ini, Form Kategori, Form User, dan Form Customer, adalah form administrasi data dasar yang terakhir, sama seperti Form Barang yang di bahas pada bab sebelumnya. Dan ketiga form ini disatukan dalam pembahasannya karena teknik maupun sistem pembuatannya serupa satu sama dengan yang lain, dan mengacu juga pada sistem pembuatan Form Barang.

Form Kategori

Dalam tampilannya, format form **Kategori** ini pasti tetap Anda kenal, karena memang dibuat sejalan dengan form dasar sebelumnya, yaitu form **Barang**.



Gambar 9-1. Tampilan Form Kategori

Yang harus Anda pastikan benar adalah hal-hal berikut:

Nama form adalah **frmKategori**

Nama DGV adalah **KatDGV**

Nama panel adalah **pnKategori**, **Visible = False**

Nama tombol di atas adalah **btnKatBaru**

Sedangkan **TextBox** yang ada:

txtKat_id – ReadOnly = True

txtKat_nama

Untuk pemrogramannya, akan dijelaskan dari bagian deklarasi di depan.

```
Imports System
Imports System.Drawing
Imports System.Windows.Forms
Imports System.Data.OleDb

Public Class frmKategori
    Dim con As New OleDbConnection
    Dim dbProvider As String
    Dim dbSource As String
    Dim ds As New DataSet
    Dim da As OleDbDataAdapter
    Dim katBaris As Integer
```

Deklarasi **Library** maupun variable bisa dilihat serupa dengan form sebelumnya, namun di sini dibuat sebuah variable **Integer** untuk men-catat baris, yaitu **katBaris**.

Kemudian dituliskan penanganan event **Form_Leave** dan **Form_Load**.

```
Private Sub frmKategori_Leave(sender As Object, e As EventArgs)
Handles Me.Leave
    con = Nothing
    ds = Nothing
    da = Nothing
End Sub

Private Sub frmKategori_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial &
"Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource
    con.Open()

    da = New OleDbDataAdapter("SELECT * from Kategori", con)
    da.Fill(ds, "Kat")
    con.Close()
    setupDGV()
End Sub
```

Di sini pun tidak banyak yang berbeda, hanya harus dipastikan bahwa query akan mengarah pada table data yang benar, yaitu table **Kategori**.

Kemudian, Anda tuliskan sub-routine **setupDGV()**.

```
PrivateSub setupDGV()
Dim i AsInteger
    KatDGV.Rows.Clear()
Dim dsTables AsDataTable = ds.Tables("Kat")
For i = 0 To ds.Tables("Kat").Rows.Count - 1
    KatDGV.Rows.Add()
With KatDGV.Rows(i)
    .Cells(0).Value = dsTables.Rows(i).Item("Kat_id")
    .Cells(1).Value = dsTables.Rows(i).Item("Kat_nama")
EndWith
Next
EndSub
```

Langkah yang dilakukan masih sama, yaitu membersihkan **KatDGV**, baru kemudian mengisikan ulang data di dalamnya.

Kemudian Anda bisa mempersiapkan event penekanan tombol. Kalau penekanan tombol **btnTambah**, yang berarti ada penambahan atau pengubahan suatu baris data, akan di bahas pada bagian akhir.

```
PrivateSub btnKatBaru_Click(sender AsObject, e AsEventArgs)
Handles btnKatBaru.Click
    pnKategori.Visible = Not (pnKategori.Visible)
    btnTambah.Text = "Tambah"
EndSub

PrivateSub btnBatal_Click(sender AsObject, e AsEventArgs)
Handles btnBatal.Click
    pnKategori.Visible = False
EndSub
```

Anda bisa lihat, bahwa di sini yang dilakukan hanya mengubah status property **Visible** dari panel Anda.

Berikutnya ada lebih sedikit teknik, yaitu apa yang harus dilakukan pada saat panel ditampilkan.

Apa yang ditampilkan di sini adalah, membersihkan isian **TextBoxNama**, tapi memberi isi pada **TextBoxID**. Isi di sini diambil dari dataset, dan dicari yang paling besar, kemudian sebelum diberikan pada **TextBox**, nilainya terlebih ditambah dengan angka **1**. Hal ini agar field **Kat_id** pada table **Kategori** akan selalu memiliki nilai yang unik, dan akan semakin besar, karena pada form ini, user tidak diberikan kemungkinan untuk membuat sendiri **ID**-nya.

```
PrivateSub pnKategori_VisibleChanged(sender AsObject, e
AsEventArgs) Handles pnKategori.VisibleChanged
If pnKategori.Visible Then
    txtKat_id.Text = CStr(getKat_id())
```

```

        txtKat_nama.Clear()
        btnTambah.Enabled = True
    EndIf
EndSub

PrivateFunction getKat_id() AsInteger
Dim i AsInteger
Dim tKat_id(ds.Tables("Kat").Rows.Count) AsInteger

For i = 0 To ds.Tables("Kat").Rows.Count - 1
    tKat_id(i) = CInt(ds.Tables("Kat").Rows(i).Item("Kat_id"))
Next
Return tKat_id.Max() + 1
EndFunction

```

Bisa Anda lihat, bahwa pengisian nilai untuk **txtKat_id.Text** adalah dengan memanggil fungsi **getKat_id()**. Sedangkan kerja dari fungsi tersebut adalah membaca satu per satu nilai field **Kat_id** dari dataset, dan menyimpannya dalam sebuah variable Array **tKat_id()**.

Pada akhirnya, nilai yang dikembalikan oleh fungsi itu adalah nilai terbesar dari Array tersebut ditambah dengan angka **1**.

Kemudian perlu juga dituliskan penanganan event double-click pada **KatDGV**. Seperti pada form sebelumnya, di sini Anda juga akan menggunakan nilai dari object **e.RowIndex** untuk menentukan posisi baris yang mendapatkan klik-ganda.

```

PrivateSub KatDGV_CellDoubleClick(sender AsObject, e
AsDataGridViewCellEventArgs) Handles KatDGV.CellDoubleClick
Dim i AsInteger
pnKategori.Visible = Not (pnKategori.Visible)
With KatDGV.Rows(e.RowIndex)
    txtKat_id.Text = .Cells(0).Value
    txtKat_nama.Text = Trim(.Cells(1).Value)
EndWith
    btnTambah.Text = "Ubah"
    katBaris = e.RowIndex
EndSub

```

Dan langkah terakhir dalam *form* ini adalah penanganan *event* penekanan tombol **btnTambah**.

```

PrivateSub btnTambah_Click(sender AsObject, e AsEventArgs)
Handles btnTambah.Click
Dim katBaru AsDataRow
If txtKat_nama.Text = ""Then
    MsgBox("Nama harus diisi.")
    txtKat_nama.Focus()
Else
Dim dsTables AsDataTable = ds.Tables("Kat")
Dim cb AsNew OleDb.OleDbCommandBuilder(da)
If btnTambah.Text = "Tambah"Then
    katBaru = ds.Tables("Kat").NewRow
    katBaru("Kat_id") = txtKat_id.Text
    katBaru("Kat_nama") = txtKat_nama.Text

```

```

        dsTables.Rows.Add(katBaru)
ElseIf btnTambah.Text = "Ubah" Then
    dsTables.Rows(katBaris).Item("Kat_nama") = _
        txtKat_nama.Text
EndIf
da.Update(ds, "Kat")
pnKategori.Visible = False
setupDGV()
EndIf
EndSub

```

Di sini Anda sudah menyelesaikan penulisan program untuk form ini. Anda bisa mencobanya untuk mengetahui hasilnya.



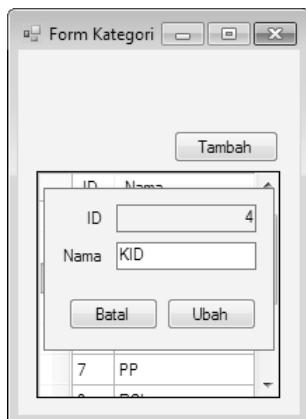
Gambar 9-2. Form Kategori

Pada saat tombol Tambah ditekan, akan menampilkan sebuah panel, dengan pengisian **Kat_id** otomatis.



Gambar 9-3. Penekanan tombol Tambah

Dan jika user melakukan klik-ganda pada data di baris ke-4:



Gambar 9-4. Klik-ganda pada baris data ke-4

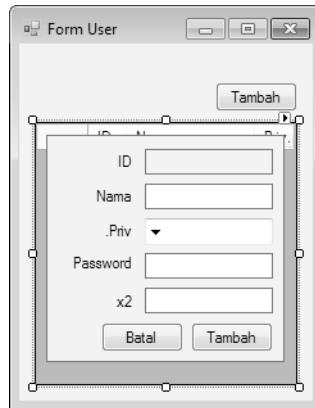
Form User

Form **User** tidak jauh berbeda dengan form **Kategori**. Namun, meskipun begitu, ada suatu fitur yang cukup mendasar pada form ini, dan itu berkaitan dengan fitur keamanan aplikasi.

Form ini pada dasarnya akan mencatat/menampilkan data user, mereka yang memiliki akses pada aplikasi ini. Akses ini ditentukan oleh form Login. Dan berhubungan dengan akses, fitur keamanan yang digunakan adalah dengan password sederhana.

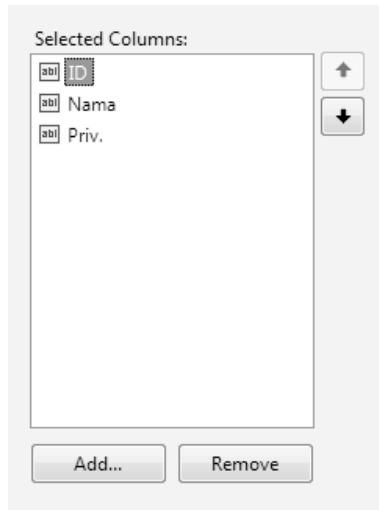
Dalam tampilannya pada **DataGridview**, tentu saja field password tidak akan ditampilkan. Namun, dalam perubahan data, ataupun penambahan data baru, isian password tidak boleh dilupakan, tapi malah dibuat dua kali, untuk memastikan tidak ada kesalahan ketik, karena password juga diberikan karakter yang berbeda, sehingga tidak bisa dibaca secara langsung.

Susunan form adalah sebagai berikut:



Gambar 9-5. Tampilan Form User

Dan perhatikan, hanya tiga buah field data yang akan ditampilkan dalam **DataGridView**. Field untuk **User_pwd** tidak ditampilkan.



Gambar 9-6. Tiga field dalam DataGridView

Data-data yang harus dipastikan:

Nama form adalah **frmUsers**

Nama DGV adalah **UsersDGV**

Nama panel adalah **pnUsers**, **Visible = False**

Nama tombol di atas adalah **btnUserBaru**

Sedangkan **TextBox** dan **ComboBox** yang ada:

TextBox txtUser_id – **ReadOnly = True**

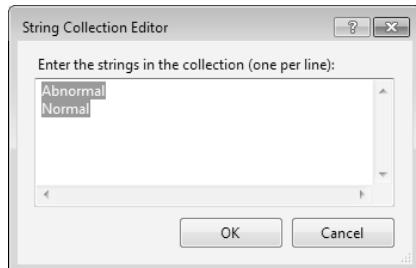
TextBox txtKat_nama

ComboBox cbUser_priv

TextBox txtUser_pwd – **PasswordChar = ***

TextBox txtUser_pwd2 – **PasswordChar = ***

Untuk **ComboBox cbUser_priv**, diberikan pilihan isian awal sebagai berikut:



Gambar 9-7. Isian untuk ComboBox

Seperti sebelumnya, dilakukan deklarasi terlebih dahulu.

```
Imports System
Imports System.Drawing
Imports System.Windows.Forms
Imports System.Data.OleDb

Public Class frmUsers
    Dim con As New OleDbConnection
    Dim dbProvider As String
    Dim dbSource As String
    Dim ds As New DataSet
    Dim da As OleDbDataAdapter
    Dim userBaris As Integer
```

Perhatikan variable **userBaris** di bagian akhir. Variable ini juga digunakan untuk mencatat posisi baris yang terpilih.

Jangan dilupakan untuk membersihkan memory dari penggunaan, saat form ditutup.

```

PrivateSub frmUsers_Leave(sender AsObject, e AsEventArgs)
HandlesMe.Leave
    con = Nothing
    ds = Nothing
    da = Nothing
EndSub

PrivateSub frmUsers_Load(sender AsObject, e AsEventArgs)
Handles MyBase.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial & _"
    "Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource
    con.Open()

    da = New OleDbDataAdapter("SELECT * from Users", con)
    da.Fill(ds, "Users")

    con.Close()
    setupDGV()
EndSub

```

Di sini Anda menyimpan data baris yang dibaca dari table ke dalam variable **ds**, dengan nama table "**Users**".

Untuk sub-routine **setupDGV**, perhatikan bahwa data yang ditampilkan dalam **UsersDGV** hanya untuk tiga buah field.

```

PrivateSub setupDGV()
    Dim i AsInteger
    Dim dsTables AsDataTable = ds.Tables("Users")
    UsersDGV.Rows.Clear()
    For i = 0 To ds.Tables("Users").Rows.Count - 1
        UsersDGV.Rows.Add()
        With UsersDGV.Rows(i)
            .Cells(0).Value = dsTables.Rows(i).Item("User_id")
            .Cells(1).Value = dsTables.Rows(i).Item("User_nama")
            .Cells(2).Value = dsTables.Rows(i).Item("User_priv")
        EndWith
    Next
EndSub

```

Untuk password, yang diwakili oleh nama field **User_pwd**, biarlah tetap berada dalam dataset**ds**. Nanti data inilah yang akan diambil oleh panel, dan bukan hanya data dari **DataGridview**.

Penekanan tombol yang hanya mengubah status **Visible** dari panel juga tetap dibuat sama.

```

PrivateSub btnUserBaru_Click(sender AsObject, e AsEventArgs)
Handles btnUserBaru.Click
    pnUsers.Visible = Not (pnUsers.Visible)
EndSub

PrivateSub btnBatal_Click(sender AsObject, e AsEventArgs)
Handles btnBatal.Click

```

```
    pnUsers.Visible = False
EndSub
```

Pada saat status panel berubah menjadi **Visible = True**, maka akan dijalankan suatu event untuk mengisi data awal. Dan dalam data awal ini, ada sebuah isian untuk **txtUser_id** (yang ditentukan sebagai `ReadOnly`) yang mengambil dari sebuah fungsi `getUser_id()`.

```
PrivateSub pnUsers_VisibleChanged(sender AsObject, e
AsEventArgs) Handles pnUsers.VisibleChanged
If pnUsers.Visible Then
    txtUser_id.Text = CStr(getUser_id())
    txtUser_nama.Clear()
    cbUser_priv.SelectedIndex = 1
    txtUser_pwd.Clear()
    txtUser_pwd2.Clear()
    btnTambah.Text = "Tambah"
EndIf
EndSub

PrivateFunction getUser_id() AsInteger
Dim i AsInteger
Dim tUser_id(ds.Tables("Users").Rows.Count) AsInteger

For i = 0 To ds.Tables("Users").Rows.Count - 1
    tUser_id(i) = _
    CInt(ds.Tables("Users").Rows(i).Item("User_id"))
Next
Return tUser_id.Max() + 1
EndFunction
```

Seperti pada form sebelumnya, field **User_id** ini juga akan dibuat numerik dan berurut, sehingga nilai awal untuk field ini adalah nilai maksimal dari semua nilai yang lain, ditambah dengan angka **1**.

Satu lagi yang perlu Anda perhatikan dalam form ini adalah item **ComboBoxcbUser_priv**. Jika Anda lihat ke dalam, isi dari pilihannya hanya ada dua, yaitu **Normal** dan **Abnormal**. Istilah ini digunakan hanya untuk memberi contoh perbedaan. Nantinya, akan dibuat sehingga seorang user dengan privilege **Normal** tidak akan diperbolehkan untuk melakukan **Import Data**.

Sekarang mari lanjutkan dengan apa yang perlu ditampilkan pada saat user melakukan klik-ganda pada UsersDGV.

```
PrivateSub UsersDGV_CellDoubleClick(sender AsObject, e
AsDataGridViewCellEventArgs) Handles UsersDGV.CellDoubleClick
Dim dsRowAsDataRow = ds.Tables("Users").Rows(e.RowIndex)

    pnUsers.Visible = Not (pnUsers.Visible)
    With UsersDGV.Rows(e.RowIndex)
        txtUser_id.Text = .Cells(0).Value
        txtUser_nama.Text = Trim(.Cells(1).Value)
        cbUser_priv.SelectedIndex = .Cells(2).Value
    EndWith
EndSub
```

```

        txtUser_pwd.Text = Trim(dsRow.Item("User_pwd"))
        txtUser_pwd2.Text = Trim(dsRow.Item("User_pwd"))
    EndWith
    userBaris = e.RowIndex
    btnTambah.Text = "Ubah"
EndSub

```

Cara yang sama seperti yang Anda lakukan dalam form **Kategori** yang lalu, cuma berbeda pada item datanya saja.

Dan akhirnya, proses penyimpanan data pada event penekanan tombol **btnTambah**.

```

PrivateSub btnTambah_Click(sender AsObject, e AsEventArgs)
Handles btnTambah.Click
Dim dsRow AsDataRow = ds.Tables("Users").Rows(userBaris)
Dim userBaru AsDataRow
If txtUser_nama.Text = ""Then
    MsgBox("Nama harus diisi.")
    txtUser_nama.Focus()
ElseIf txtUser_pwd.Text = ""Or txtUser_pwd2.Text = ""Then
    MsgBox("Password harus diisi.")
    txtUser_pwd.Focus()
ElseIf txtUser_pwd.Text <> txtUser_pwd2.Text Then
    MsgBox("Password harus diketik dua kali.")
    txtUser_pwd.Focus()
Else
    Dim cb AsNew OleDb.OleDbCommandBuilder(da)

    If btnTambah.Text = "Tambah"Then
        userBaru = ds.Tables("Users").NewRow
        userBaru("User_id") = txtUser_id.Text
        userBaru("User_nama") = txtUser_nama.Text
        userBaru("User_priv") =
        IIf(cbUser_priv.SelectedIndex = 0, "0", "1")
        userBaru("User_pwd") = txtUser_pwd.Text
        ds.Tables("Users").Rows.Add(userBaru)
    ElseIf btnTambah.Text = "Ubah"Then
        dsRow.Item("User_nama") = txtUser_nama.Text
        dsRow.Item("User_priv") =
        IIf(cbUser_priv.SelectedIndex = 0, "0", "1")
        dsRow.Item("User_pwd") = txtUser_pwd.Text
    EndIf
    da.Update(ds, "Users")

    pnUsers.Visible = False
    setupDGV()
EndIf
EndSub

```

Di sini, salah satu kondisi yang harus dipastikan adalah bahwa masukan password antara kedua **TextBox** harus sama.

```

ElseIf txtUser_pwd.Text <> txtUser_pwd2.Text Then
    MsgBox("Password harus diketik dua kali.")
    txtUser_pwd.Focus()

```

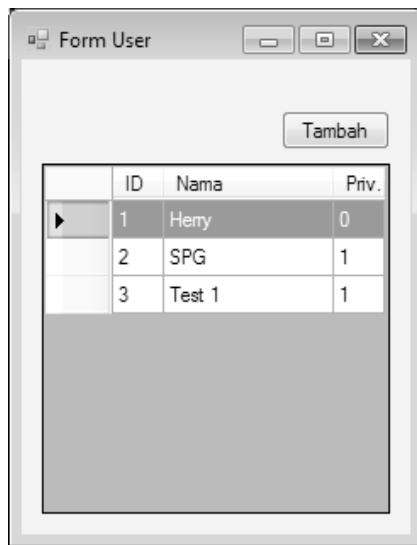
Kemudian, pada saat pengisian field **User_priv**, digunakan fungsi **IIF()**.

```
userBaru("User_priv") =  
    IIf(cbUser_priv.SelectedIndex = 0, "0", "1")
```

Karena pada awalnya, privilege “**Normal**” adalah menggunakan **ID = 1**, dan “**Abnormal**” menggunakan **ID = 0**, dan dalam penampilannya, secara default digunakan tampilan privilege **Normal**.

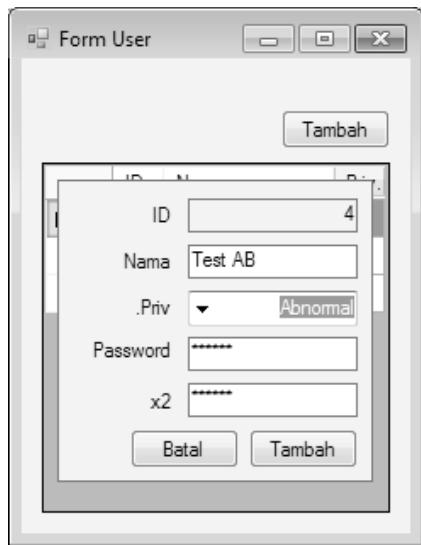
Maka dari itu, fungsi **IIF()** akan memeriksa apakah nilai yang terpilih dari **ComboBoxcbUser_priv** adalah sama dengan **0** (yang berarti “**Abnormal**”), kalau iya, maka akan diberikan nilai “**0**”, sedangkan kalau tidak (yang berarti user memilih **Normal**), maka akan diberikan nilai “**1**”.

Demikianlah akhir dari pemrograman form **Users** ini. Jika Anda jalankan, akan memberikan tampilan sebagai berikut:



Gambar 9-8. Tampilan awal form *Users*

Dan pada saat user menekan tombol **Tambah**, dan mengisi seorang user baru.



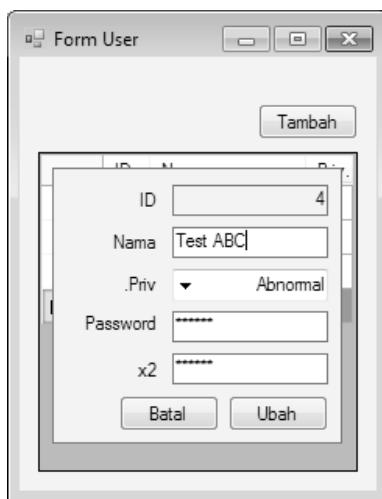
Gambar 9-9. User menambahkan seorang user baru

Setelah user menekan tombol **Tambah** (yang berada di dalam panel), maka user baru tersebut akan ditambahkan dalam **Data Grid**.

	ID	Nama	Prv.
▶	1	Henry	0
	2	SPG	1
	3	Test 1	1
	4	Test AB	0

Gambar 9-10. User baru Test AB dalam Data Grid

Dan kemudian mendapatkan klik-ganda.



Gambar 9-11. Pengubahan Nama user

Dan setelah user menekan tombol **Ubah**.

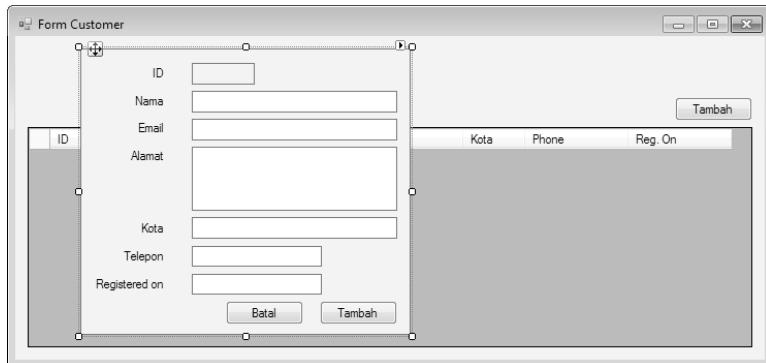
A screenshot of the "Form User" window showing a list of users in a table. The table has columns for ID, Nama, and Priv. The data is as follows:

	ID	Nama	Priv.
▶	1	Henry	0
	2	SPG	1
	3	Test 1	1
	4	Test ABC	0

Gambar 9-12. Data user berhasil diubah

Form Customer

Tidak banyak perbedaan antara form Customer ini dengan form Users. Yang paling mencolok adalah jumlah item data pada form ini. Karena itu, ukuran form inipun harus disesuaikan dengan jumlah field, sehingga memang menjadi lebih lebar.



Gambar 9-13.Tampilan Form Customer

Data-data yang harus dipastikan dalam form ini adalah:

Nama form adalah **frmCustomer**

Nama DGV adalah **CustDGV**

Nama panel adalah **pnCustomer, Visible = False**

Nama tombol di atas adalah **btnCustBaru**

Sedangkan **TextBox** yang ada:

TextBox txtCust_id – ReadOnly = True

TextBox txtCust_nama

TextBox txtCust_email

TextBox txtCust_alamat – MultiLine = True

TextBox txtCust_kota

TextBox txtCust_phone

TextBox txtCust_reg_on

Pemrograman untuk form ini dimulai dari deklarasi variable.

```

Imports System
Imports System.Drawing
Imports System.Windows.Forms
Imports System.Data.OleDb

Public Class frmCustomer
Dim con As New OleDbConnection
Dim dbProvider As String
Dim dbSource As String
Dim ds As New DataSet
Dim da As OleDbDataAdapter
Dim custBaris As Integer

```

Seperti sebelumnya, ada sebuah variable bernama **custBaris** yang digunakan untuk mencatat baris data yang terpilih.

Dilanjutkan dengan penanganan event**Form_Load** dan **Form_Leave**.

```

Private Sub frmCustomer_Leave(sender As Object, e As EventArgs)
Handles Me.Leave
    con = Nothing
    ds = Nothing
    da = Nothing
End Sub

Private Sub frmCustomer_Load(sender As Object, e As EventArgs)
Handles Me.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial &
"Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource
    con.Open()

    da = New OleDbDataAdapter("SELECT * from Customer", con)
    da.Fill(ds, "Customer")

    con.Close()

    setupDGV()
End Sub

```

Di sini, data baris disimpan juga dalam variable dataset **ds**, dan dalam table "**Customer**".

Seperti yang lain, event **Form_Load** ini juga memanggil sub-routine **setupDGV()**.

```

Private Sub setupDGV()
Dim i As Integer
Dim dsRow As DataRow

CustDGV.Rows.Clear()
For i = 0 To ds.Tables("Customer").Rows.Count - 1
    CustDGV.Rows.Add()
    dsRow = ds.Tables("Customer").Rows(i)
    With CustDGV.Rows(i)
        .Cells(0).Value = dsRow.Item("Cust_id")
    End With
Next i
End Sub

```

```

.Cells(1).Value = Trim(dsRow.Item("Cust_nama"))
.Cells(2).Value = Trim(dsRow.Item("Cust_email"))
.Cells(3).Value = Trim(dsRow.Item("Cust_alamat"))
.Cells(4).Value = Trim(dsRow.Item("Cust_kota"))
.Cells(5).Value = Trim(dsRow.Item("Cust_phone"))

.Cells(6).Value = _
Format(dsRow.Item("Cust_reg_on"), "dd-MM-yyyy")
EndWith
Next
EndSub

```

Pada sub-routine ini, masih digunakan sistem yang sama. Cuma pada kolom data terakhir dalam **Data Grid**, penulisan data tanggal menggunakan format yang sudah ditentukan.

```

.Cells(6).Value = _
Format(dsRow.Item("Cust_reg_on"), "dd-MM-yyyy")

```

Diikuti dengan penekanan tombol yang mengakibatkan perubahan status **Visible** dari panel, dan penanganan event **VisibleChanged** dan fungsi **getCust_id()** yang dipanggilnya.

```

PrivateSub btnCustBaru_Click(sender AsObject, e AsEventArgs)
Handles btnCustBaru.Click
    pnCustomer.Visible = Not (pnCustomer.Visible)
EndSub

PrivateSub btnBatal_Click(sender AsObject, e AsEventArgs)
Handles btnBatal.Click
    pnCustomer.Visible = False
EndSub

PrivateSub pnCustomer_VisibleChanged(sender AsObject, e
AsEventArgs) Handles pnCustomer.VisibleChanged
If pnCustomer.Visible Then
    txtCust_id.Text = CStr(getCust_id())
    txtCust_nama.Clear()
    txtCust_email.Clear()
    txtCust_alamat.Clear()
    txtCust_kota.Clear()
    txtCust_phone.Clear()
    txtCust_reg_on.Text = CDate(Date.Today)
    btnTambah.Text = "Tambah"
EndIf
EndSub

PrivateFunction getc

```

Penanganan event **CellDoubleClick** juga tidak memiliki banyak perbedaan dari sebelumnya, hanya pada nama table maupun variable yang digunakan.

```
PrivateSub CustDGV_CellDoubleClick(sender AsObject, e AsDataGridViewCellEventArgs) Handles CustDGV.CellDoubleClick
    pnCustomer.Visible = Not (pnCustomer.Visible)

    With CustDGV.Rows(e.RowIndex)
        txtCust_id.Text = .Cells(0).Value
        txtCust_nama.Text = Trim(.Cells(1).Value)
        txtCust_email.Text = Trim(.Cells(2).Value)
        txtCust_alamat.Text = Trim(.Cells(3).Value)
        txtCust_kota.Text = Trim(.Cells(4).Value)
        txtCust_phone.Text = Trim(.Cells(5).Value)
        txtCust_reg_on.Text = .Cells(6).Value
    EndWith
    custBaris = e.RowIndex
    btnTambah.Text = "Ubah"
EndSub
```

Dan penanganan penekanan tombol **btnTambah**, yang merupakan inti dari form ini.

```
PrivateSub btnTambah_Click(sender AsObject, e AsEventArgs)
Handles btnTambah.Click
Dim custBaru AsDataRow
Dim dsRow AsDataRow = ds.Tables("Customer").Rows(custBaris)
If txtCust_nama.Text = ""Then
    MsgBox("Nama harus diisi.")
    txtCust_nama.Focus()
ElseIf txtCust_email.Text = ""Then
    MsgBox("Email harus diisi.")
    txtCust_email.Focus()
ElseIf txtCust_alamat.Text = ""Then
    MsgBox("Alamat harus diisi.")
    txtCust_alamat.Focus()
ElseIf txtCust_kota.Text = ""Then
    MsgBox("Kota harus diisi.")
    txtCust_kota.Focus()
ElseIf txtCust_phone.Text = ""Then
    MsgBox("Telepon harus diisi.")
    txtCust_phone.Focus()
ElseIf txtCust_reg_on.Text = ""Then
    MsgBox("Tanggal harus diisi.")
    txtCust_reg_on.Focus()
Else
    Dim cb AsNew OleDb.OleDbCommandBuilder(da)
    If btnTambah.Text = "Tambah"Then
        custBaru = ds.Tables("Customer").NewRow
        custBaru("Cust_id") = txtCust_id.Text
        custBaru("Cust_nama") = txtCust_nama.Text
        custBaru("Cust_email") = txtCust_email.Text
        custBaru("Cust_alamat") = txtCust_alamat.Text
        custBaru("Cust_kota") = txtCust_kota.Text
        custBaru("Cust_phone") = txtCust_phone.Text
        custBaru("Cust_reg_on") = CDate(Date.Today)
        ds.Tables("Customer").Rows.Add(custBaru)
```

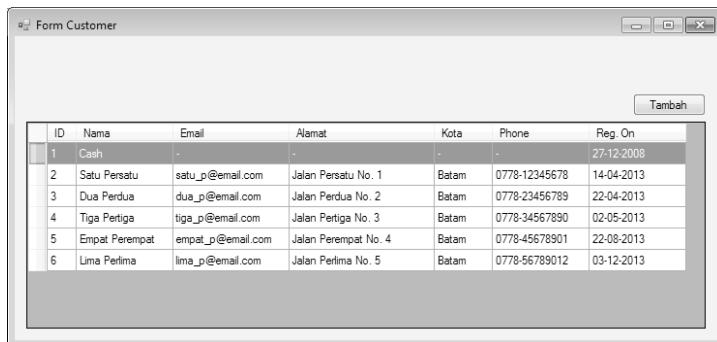
```

ElseIf btnTambah.Text = "Ubah"Then
    dsRow.Item("Cust_nama") = txtCust_nama.Text
    dsRow.Item("Cust_email") = txtCust_email.Text
    dsRow.Item("Cust_alamat") = txtCust_alamat.Text
    dsRow.Item("Cust_kota") = txtCust_kota.Text
    dsRow.Item("Cust_phone") = txtCust_phone.Text
    dsRow.Item("Cust_reg_on") = txtCust_reg_on.Text
EndIf
da.Update(ds, "Customer")
pnCustomer.Visible = False
setupDGV()
EndIf
EndSub

```

Di sini, kondisi jauh lebih banyak dari form yang lain, karena memang item data juga jauh lebih banyak. Namun, proses penyimpanan tetap menggunakan cara yang sama, jadi seharusnya tidak ada kesulitan bagi Anda untuk mengikutinya.

Jika Anda jalankan, maka tampilan awalnya akan langsung menampilkan semua Customer yang ada dalam database.



Gambar 9-14. Tampilan Form Customer

Dan pada saat user menekan tombol **Tambah**, form akan menampilkan panel **pnCustomer**, dengan isian data default **Cust_id** yang mengambil nilai selanjutnya dari nilai **Cust_id** yang lain, dan **Cust_reg_on**, yang mengambil nilai dari tanggal hari ini. Dan jika kemudian user tersebut mengisikan data seorang **Customer** baru, maka tampilan akan tampak sebagai berikut.

Form Customer

ID	7
Nama	Test
Email	test@email.com
Alamat	Jalan Test No. 7
Kota	Test
Telepon	0778-TEST-001
Registered on	30/5/2014

Tambah

	Kota	Phone	Reg. On
1	-	-	27-12-2008
2	Batam	0778-12345678	14-04-2013
3	Batam	0778-23456789	22-04-2013
4	Batam	0778-34567890	02-05-2013
5	Batam	0778-45678901	22-08-2013
6	Batam	0778-56789012	03-12-2013

Batal **Tambah**

Gambar 9-15. Proses penambahan data

Data baru tersebut akan langsung ditampilkan dalam **Data Grid**.

Form Customer

	ID	Nama	Email	Alamat	Kota	Phone	Reg. On
1	Cash	-	-	-	-	-	27-12-2008
2	Satu Persatu	satu_p@email.com	Jalan Persatu No. 1	Batam	0778-12345678	14-04-2013	
3	Dua Perdua	dua_p@email.com	Jalan Perdua No. 2	Batam	0778-23456789	22-04-2013	
4	Tiga Pertiga	tiga_p@email.com	Jalan Pertiga No. 3	Batam	0778-34567890	02-05-2013	
5	Empat Perempat	empat_p@email.com	Jalan Perempat No. 4	Batam	0778-45678901	22-08-2013	
6	Lima Perlima	lima_p@email.com	Jalan Perlima No. 5	Batam	0778-56789012	03-12-2013	
7	Test	test@email.com	Jalan Test No. 7	Test	0778-TEST-001	30-05-2014	

Gambar 9-16. Data Customer baru

Kemudian jika user melakukan klik-ganda pada data baru tersebut.

ID	7
Nama	Test 07
Email	test@email.com
Alamat	Jalan Test No. 7
Kota	Test
Telepon	0778-TEST-001
Registered on	30-05-2014

Batal **Ubah**

Gambar 9-17. Pengubahan data

Setelah user menekan tombol **Ubah**, maka form akan menyimpan perubahan tersebut, dan me-refresh tampilan dalam **Data Grid**.

ID	Nama	Email	Alamat	Kota	Phone	Reg. On
1	Cash	-	-	-	-	27-12-2008
2	Satu Persatu	satu_p@email.com	Jalan Persatu No. 1	Batam	0778-12345678	14-04-2013
3	Dua Perdua	dua_p@email.com	Jalan Perdua No. 2	Batam	0778-23456789	22-04-2013
4	Tiga Pertiga	tiga_p@email.com	Jalan Pertiga No. 3	Batam	0778-34567890	02-05-2013
5	Empat Perempat	empat_p@email.com	Jalan Perempat No. 4	Batam	0778-45678901	22-08-2013
6	Lima Perlima	lima_p@email.com	Jalan Perlima No. 5	Batam	0778-56789012	03-12-2013
7	Test 07	test@email.com	Jalan Test No. 7	Test	0778-TEST-001	30-05-2014

Gambar 9-18. Data baru setelah diubah

Jadi, demikianlah akhir dari form-form dasar. Selanjutnya, akan dibahas pembuatan form **Penjualan**.

###

BAB 10 | Form Jual

Form **Jual** ini adalah form utama dari aplikasi ini. Di sini akan terjadi transaksi penjualan, jadi ada beberapa unsur yang akan ditampilkan dan dikerjakan dalam form ini.

Salah satunya adalah proses penyusunan sebuah **Data Grid** melalui pemrograman. Selain itu, ada juga pengaturan kolom, mana kolom yang bisa dimasuki dan yang tidak. Ada juga kolom yang nilai di dalamnya mengambil dari hasil nilai kolom lain. Selain itu, dalam form ini, user memasukkan data baru langsung pada **Data Grid**, bukan melalui panel seperti pada form sebelumnya.

Marilah kita mulai dengan penyusunan **Data Grid** tersebut, melalui sub-routine **setupDGV()**.

Tambahkan sebuah Window Forms pada aplikasi Anda, dan beri nama form ini dengan nama **frmJual**. Kemudian, lakukan klik-ganda pada form untuk mendapatkan sub-routine **Form_Load**.

Dalam sub-routine tersebut, panggilah sub-routine **setupDGV()**. Selain itu, deklarasikanlah sebuah variable bernama **JualDGV** sebagai sebuah object **DataGridView**.

```
Private JualDGV AsNewDataGridView  
  
PrivateSub Form2_Load(sender AsObject, e AsEventArgs) _  
Handles MyBase.Load  
    SetupDGV()  
EndSub
```

Kemudian, mulailah untuk masuk dalam sub-routine **setupDGV()**. Langkah pertama yang dilakukan dalam sub-routine itu, tambahkanlah sebuah **DataGridview** object (dengan menggunakan nama variable yang sudah Anda deklarasikan) pada kumpulan Controls untuk form ini.

```
PrivateSub SetupDGV()
Me.Controls.Add(JualDGV)
```

Me.Controls adalah kumpulan kontrol yang terdapat pada object **Me**, dalam hal ini form **frmJual** itu sendiri. Setelah itu, Anda bisa mulai bekerja dengan object baru Anda, misalnya dengan mengatur jumlah kolom, atau mengatur format tampilannya.

```
JualDGV.ColumnCount = 8
With JualDGV.ColumnHeadersDefaultCellStyle
    .BackColor = Color.Navy
    .ForeColor = Color.White
    .Font = NewFont(JualDGV.Font, FontStyle.Bold)
EndWith
```

Di sini diatur supaya **DataGrid** mempersiapkan 7 buah kolom baru, kemudian diatur penggunaan warna cell dalam **DataGrid** tersebut.

Kemudian mulailah untuk mengatur masing-masing kolom pada **DataGrid** ini. **DataGrid** ini akan digunakan untuk menyimpan data item penjualan, jadi, selain nomor baris, harus ada kolom yang menampung kode barang, nama barang, kemudian jumlah penjualan, perhitungan unit barang, harga, dan terakhir sub-total. Ada 7 buah kolom, sesuai dengan yang sebelumnya sudah ditentukan di atas.

```
With JualDGV
    .Name = "JualDGV"
    .Location = NewPoint(8, 132)
    .Size = NewSize(653, 250)
    .AutoSizeRowsMode = _
    DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders
    .ColumnHeadersBorderStyle = _
        DataGridViewHeaderBorderStyle.Single
    .CellBorderStyle = DataGridViewCellBorderStyle.Single
    .GridColor = Color.Black
    .RowHeadersVisible = False

    .Columns(0).Width = 30
    .Columns(0).ReadOnly = True
    .Columns(0).SortMode = _
        DataGridViewColumnSortMode.NotSortable
    .Columns(0).DefaultCellStyle.Alignment = _
        DataGridViewContentAlignment.MiddleRight
    .Columns(0).DefaultCellStyle.BackColor = Color.LightGray
    .Columns(1).Name = "Kode"
    .Columns(1).Width = 80
    .Columns(1).SortMode = _
        DataGridViewColumnSortMode.NotSortable
    .Columns(2).Name = "Nama"
    .Columns(2).Width = 200
    .Columns(2).ReadOnly = True
    .Columns(2).SortMode = _
        DataGridViewColumnSortMode.NotSortable
    .Columns(2).DefaultCellStyle.BackColor = Color.LightGray
```

```

.Columns(3).Name = "Qty"
.Columns(3).Width = 40
.Columns(3).SortMode = _
DataGridViewColumnSortMode.NotSortable
.Columns(4).Name = "Unit"
.Columns(4).Width = 50
.Columns(4).ReadOnly = True
.Columns(4).SortMode = _
DataGridViewColumnSortMode.NotSortable
.Columns(4).DefaultCellStyle.BackColor = Color.LightGray
.Columns(5).Name = "Harga"
.Columns(5).Width = 100
.Columns(5).SortMode = _
DataGridViewColumnSortMode.NotSortable
.Columns(6).Name = "Sub-Total"
.Columns(6).Width = 150
.Columns(6).ReadOnly = True
.Columns(6).SortMode = _
DataGridViewColumnSortMode.NotSortable
.Columns(6).DefaultCellStyle.BackColor = Color.LightGray
.Columns(7).Name = "CurStock"
.Columns(7).Visible = False

.SelectionMode = DataGridViewSelectionMode.CellSelect
.MultiSelect = False
.AllowUserToResizeColumns = False
.AllowUserToResizeRows = False
.AllowUserToAddRows = False
.ColumnHeadersHeightSizeMode = _
DataGridViewColumnHeadersHeightSizeMode.DisableResizing
EndWith

```

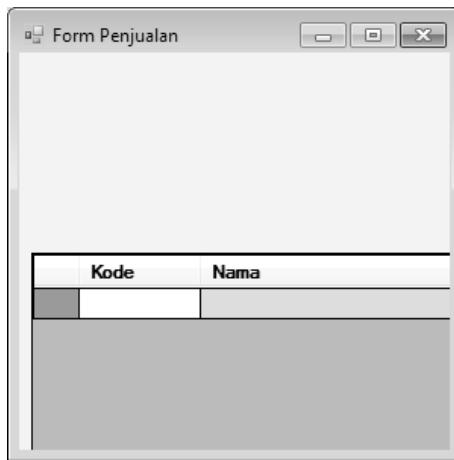
Setelah semua setting yang diperlukan dituliskan, akhiri dengan menambahkan sebuah baris data dalam **DataGrid** tersebut.

```
JualDGV.Rows.Add()
```

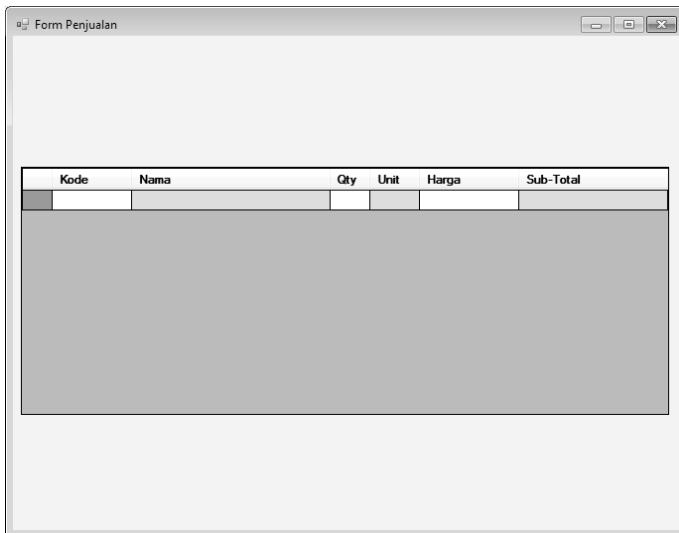
```
EndSub
```

Jalankan program Anda untuk melihat bagaimana tampilannya. Lihat Gambar 10-1.

Anda bisa lihat dalam tampilan awal tersebut, bahwa ukuran form kurang besar. Aturlah besar form supaya bisa menampung object **DataGrid** tersebut dengan sempurna.



Gambar 10-1. Tampilan awal form Penjualan



Gambar 10-2. Ukuran form yang sudah tepat

Ukuran form sudah tepat, dan Anda juga bisa melihat, bahwa dalam subroutine **setupDGV()** tadi, tidak hanya urutan kolom yang diatur, tetapi juga termasuk tampilan. Termasuk juga warna kolom yang berbeda, untuk membantu user membedakan posisi kolom.

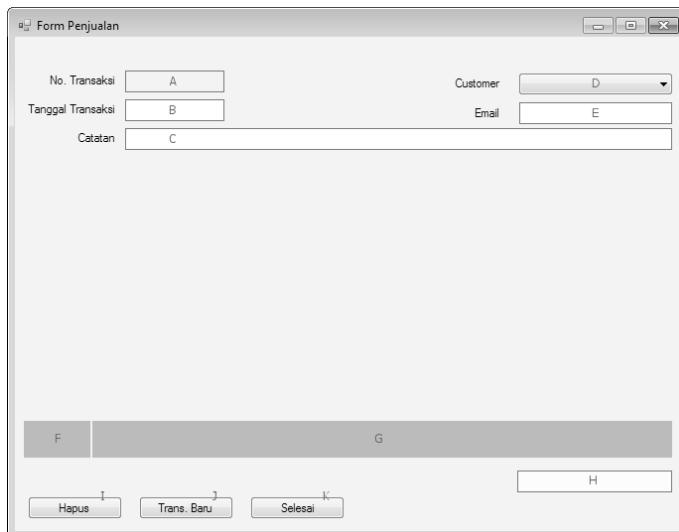
Perhatikan bahwa kolom terakhir, CurStock, tidak tampak karena memang berstatus Visible = False.

Langkah berikutnya adalah untuk menambahkan element-element pendukung pada form tersebut.

Pada form ini, terdapat beberapa **TextBox** biasa, dan sebuah **ComboBox**. Selain itu ada juga dua buah **TextBox**, di posisi agak ke bawah, yang akan menampilkan nama barang dalam font yang cukup besar, untuk menghindari kesalahan yang mungkin terjadi karena user terlalu terburu-buru.

Yang terakhir adalah adanya beberapa buah tombol di bagian bawah dari form ini.

Sebagai contoh, maupun sebagai pegangan, Anda dapat melihat dari gambar di bawah ini.



Gambar 10-3. Pegangan posisi dan item detail

Atur property dari kontrol-kontrol tersebut sebagai berikut:

- A: TextBox, Name = tJual_id, ReadOnly
- B: TextBox, Name = tJual_tgl
- C: TextBox, Name = tJual_note

```

D: ComboBox, Name = cbCust
E: TextBox, Name = tCust_email, ReadOnly
F: TextBox, Name = tBig_Qty
G: TextBox, Name = tBig_Name
H: TextBox, Name = tTotal, ReadOnly
I: Button, Name = btnHapus
J: Button, Name = btnBaru
K: Button, Name = btnSelesai

```

Selanjutnya, persiapkan Library dan variable-variable untuk koneksi database, dan penulisan sub-routine **LoadTables()**, untuk membuka table-table data yang diperlukan.

```

Imports System
Imports System.Windows
Imports System.Windows.Forms
Imports System.Data.OleDb

Public Class frmJual
Dim con As New OleDb.OleDbConnection
Dim dbProvider As String
Dim dbSource As String
Dim ds As New DataSet
Dim da As OleDb.OleDbDataAdapter

```

Dan sub-routine pembukaan table.

Di sini akan dibuat sebuah sub-routine yang bisa dipakai berulang untuk masing-masing table yang akan dibuka, yaitu table **Customer**, **Barang**, **Jual** dan **Jual_details**.

```

Private Sub LoadTables(tTable As String, tVar As DataSet)
Dim sql As String = ""
Select Case tTable
Case "Customer"
    sql = "SELECT * FROM Customer"
Case "Barang"
    sql = "SELECT Brg.Brg_id, Brg.Brg_nama, "& _
"Brg.Brg_unit, Brg.Brg_barcode, Brg.Brg_hpp, "& _
"Det.Brg_det_stock "& _
"FROM Barang Brg, Barang_details Det"& _
"WHERE Brg.Brg_id = Det.Brg_id"
Case "Jual"
    sql = "SELECT DISTINCT Jual_id FROM Jual "& _
"ORDER BY Jual_id"
Case "Jual_details"
    sql = "SELECT MAX(Jual_det_id) Jual_det_id "& _
"FROM Jual_details"
End Select

da = New OleDb.OleDbDataAdapter(sql, con)

```

```

        da.Fill(tVar, tTable)
EndSub

Perbaiki juga sub-routine untuk event Form_Load, dengan menambahkan proses koneksi dan pembukaan table, dengan memanfaatkan sub-routine LoadTables() ini.

PrivateSub frmJual_Load(sender AsObject, e AsEventArgs)
HandlesMe.Load
    dbProvider = "Provider=SQLOleDb;"
    dbSource = "Data Source=chibi\sqlexpress;Initial "&
"Catalog=dbToko;Integrated Security=SSPI"
    con.ConnectionString = dbProvider & dbSource
    con.Open()

    SetupDGV()
    LoadTables("Customer", ds)
    LoadTables("Barang", ds)
    LoadTables("Jual", ds)
    LoadTables("Jual_details", ds)
    setupLayout()
EndSub

```

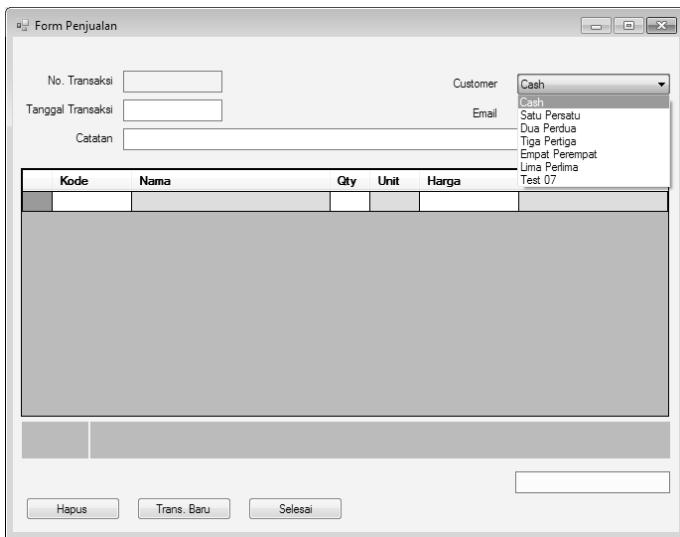
Setelah itu, buatlah sebuah sub-routine baru (yang dipanggil pada akhir event **Form_Load()**) untuk mengatur element lain dari form ini, yaitu **setupLayout()**.

```

PrivateSub setupLayout()
Dim i AsInteger
Dim dsRow AsDataRow
    table.Columns.Add("Cust_nama", GetType(String))
    table.Columns.Add("Cust_id", GetType(Integer))
For i = 0 To ds.Tables("Customer").Rows.Count - 1
    dsRow = ds.Tables("Customer").Rows(i)
    addrow(dsRow.Item("Cust_nama"), dsRow.Item("Cust_id"))
Next
    cbCust.DataSource = table
    cbCust.DisplayMember = "Cust_nama"
    cbCust.ValueMember = "Cust_id"
EndSub

```

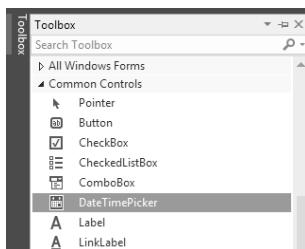
Jika kemudian program Anda jalankan, maka tampilan akan semakin mendekati akhirnya, dengan **ComboBoxCustomer** sudah aktif.



Gambar 10-4. Form dengan ComboBox Customer

Sekarang, bagaimana jika misalnya Anda mau menambahkan fitur baru, khususnya pada saat user hendak mengisikan data tanggal. Seperti yang Anda bisa pikirkan, format tanggal ada berbagai macam, misalnya **30-May-2014**, **30-05-2014**, **30/5/2014**, dan macam-macam lainnya.

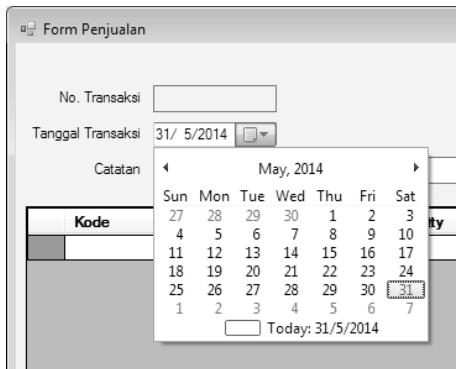
Di sini, Anda akan belajar mengenai object **DateTimePicker** yang sudah disediakan dalam Toolbox Anda.



Gambar 10-5. Object DateTimePicker

Tariklah object tersebut dari Toolbox ke form Anda, dan persis di atas **TextBox** tanggal Anda. Sekalian, buatlah supaya status **Visible** dari **TextBoxJual_tgl** itu menjadi **False**, sehingga yang akan terlihat nantinya hanyalah object **DateTimePicker** tersebut. Kemudian, ubahlah

property **Name** menjadi **dtJual_tgl**, dan property**Format** menjadi **Custom**.



Gambar 10-6. Tampilan object DateTimePicker

Selipkan baris program berikut pada bagian akhir dari sub-routine **setupLayout()**.

```
dtJual_tgl.Format = DateTimePickerFormat.Custom  
dtJual_tgl.CustomFormat = "dd-MMM-yy"  
dtJual_tgl.Value = Today
```

Di sini, object**dtJual_tgl** tersebut dipaksa untuk menggunakan format **Custom**, yaitu “**dd-MMM-yy**”, yang akan menunjukkan nilai “**30-May-14**”.

Satu lagi yang belum terisi adalah **TextBox** No. Transaksi. Format dari **TextBox** ini adalah **10** karakter, dengan **4** digit pertama urutan angka, diikuti tanda titik, kemudian **2** angka bulan, tanda titik, dan **2** angka tahun.

Maka, tuliskan fungsi ini:

```
PrivateFunction getJual_id() AsString  
Dim txtID AsString = ""  
Dim txtTemp AsString  
Dim i AsInteger  
Dim tJual(ds.Tables("Jual").Rows.Count) AsInteger  
  
If ds.Tables("Jual").Rows.Count = 0 Then  
    txtID = "0001"  
Else  
    For i = 0 To ds.Tables("Jual").Rows.Count - 1  
        txtTemp = _  
        Microsoft.VisualBasic.Left(ds.Tables("Jual"). _  
        Rows(i).Item("Jual_id"), 4)
```

```

        tJual(i) = CInt(txtTemp)
Next
        txtID = (tJual.Max() + 1).ToString("0000")
EndIf
        txtID = txtID & "." & Format(DateTime.Today, "MM") &
        " ." & Format(DateTime.Today, "YY")

Return txtID
EndFunction

```

Cara kerja fungsi ini agak berbeda dengan fungsi **getXxx_id()** pada form-form sebelumnya, karena tipe data **Jual_id** bukanlah hanya sebagai numerik, melainkan sebuah **String** dengan **10** karakter. Misal datanya adalah '**0001.05.14**', di mana 4-digit pertama adalah nomor urut, namun dengan format **4** karakter, diikuti tanda titik, kemudian **2**-digit bulan (yaitu bulan **Mei**), tanda titik, dan **2**-digit tahun.

Pertama, fungsi akan memerlukan apakah dalam table **Jual** sudah ada data. Apabila masih kosong, maka fungsi menyimpan nomor awal langsung sebagai '**0001**' dalam sebuah variable **txtID**.

Namun, apabila sudah ada data dalam table **Jual**, maka fungsi akan melakukan looping pada table itu, dan menyimpan **4**-digit pertama dari nilai **Jual_id** yang lain (dengan menggunakan perintah **Left()**), dan mengubahnya menjadi tipe data **Integer**, dan menyimpannya dalam variable Array **tJual()**.

Sampai di sini sudah sama, nilai variable **txtID** diambil dari nilai terbesar ditambah dengan angka **1** dari variable Array tersebut, namun sebelumnya dikonversikan terlebih dahulu menjadi sebuah **String**, dengan format **4**-digit, dengan karakter '**0**' di depan.

```
txtID = (tJual.Max() + 1).ToString("0000")
```

Jadi, jika nilai terpesar adalah **2**, maka nilai **txtID** = '**0003**', dan jika nilai terbesar adalah **298**, maka nilai **txtID** menjadi '**0299**'.

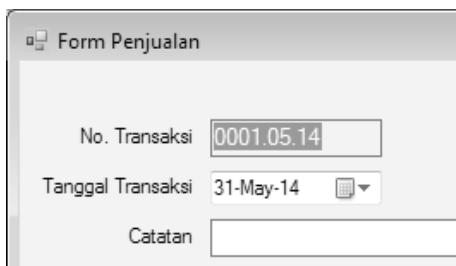
Setelah itu, tinggal digabung antara nilai **4**-digit tersebut dengan tanda titik dan format **String** dari **Bulan** dan **Tahun**.

```
txtID = txtID & "." & Format(DateTime.Today, "MM") &
        " ." & Format(DateTime.Today, "YY")
```

Dan kemudian tuliskan juga pemanggilnya di bagian bawah dari **setupLayout()**.

```
tJual_id.Text = getJual_id()
```

Maka, jika dijalankan, No. Transaksi akan sudah terisi.



Gambar 10-7. Nomor Transaksi otomatis

Langkah selanjutnya adalah mendeteksi pergantian nilai pada **ComboBox cbCust**. Jika ada pergantian nilai, dan nilai adalah valid, maka tuliskan data email dari customer terpilih (field **Cust_email**) pada **TextBox Cust_email**.

Di sini digunakan event **SelectedValueChanged** untuk item **cbCust** tersebut.

```
Private Sub cbCust_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cbCust.SelectedIndexChanged
    Dim CustRow() As DataRow
    If IsNumeric(cbCust.SelectedValue.ToString()) Then
        CustRow = ds.Tables("Customer").Select("Cust_id = " & _
            cbCust.SelectedValue.ToString())
    If CustRow IsNot Nothing Then
        tCust_email.Text = CustRow(0)("Cust_email")
    EndIf
    EndIf
EndSub
```

Prosesnya cukup sederhana, di mana fungsi akan memeriksa nilai yang didapat dari pergantian nilai **ComboBox**. Kemudian akan dilakukan pencarian data pada dataset, dan hasilnya dituliskan dalam **TextBox** yang sesuai.

Anda bisa lihat dalam hasilnya, bahwa data email akan dituliskan dalam **TextBox** yang sesuai.



Gambar 10-8. Data email dalam TextBox

Satu lagi, Anda perlu mendeteksi pergantian nilai pada item DateTimePicker. Jika berubah, maka simpan nilai yang baru pada variable **tJual_tgl** yang sudah dipersiapkan.

```
Private Sub dtJual_tgl_ValueChanged(sender As Object, e  
As EventArgs) Handles dtJual_tgl.ValueChanged  
If IsDate(dtJual_tgl.Value) Then  
    tJual_tgl.Text = dtJual_tgl.Value  
End If  
End Sub
```

Sekarang, Anda bisa mulai masuk dalam DataGrid**JualDGV**. Element ini adalah element yang paling rumit pada form ini, maka perhatikan dengan baik struktur dari unit ini.

Anda lihat, bahwa pada saat form dibuka, maka ada beberapa sub-routine yang otomatis dipanggil oleh event **Form_Load**, salah satunya adalah **setupDGV()**. Dan pada sub-routine **setupDGV()** tersebut, jika Anda lihat di dalamnya, ada statement untuk menambahkan satu baris data pada DataGrid.

Bagaimana jika Anda berikan suatu tindakan yang akan menambahkan urutan nomor baris secara otomatis? Jadi, pada pembukaan form, akan ada baris dengan nomor **1**. Jika nantinya user menambahkan satu baris lagi, baris baru akan memiliki nomor **2**. Begitu seterusnya.

Untuk itu, Anda gunakan event **RowsAdded** pada item **JualDGV**. Namun perlu Anda ingat bahwa DataGrid **JualDGV** tidak ada dalam design form Anda, karena element ini Anda buat secara program. Karena itu, element ini juga tidak ditampilkan dalam **Designer** Anda, karena itu seluruh header sub-routine harus Anda tuliskan sendiri.

```
Private Sub JualDGV_RowsAdded(sender As Object, e  
As DataGridViewRowsAddedEventArgs) Handles JualDGV.RowsAdded  
    JualDGV.Item(0, e.RowIndex).Value = CInt(e.RowIndex + 1)  
End Sub
```

Jadi, pada saat baris ditambahkan pada DataGrid tersebut, maka akan digunakan object **e.RowIndex** untuk menentukan cell pada kolom pertama, dan memberinya nilai awal.

Namun, jika Anda perhatikan pada tampilan program Anda.

```
Private Sub JualDGV_RowsAdded(sender As Object, _  
    e As DataGridViewRowsAddedEventArgs) _  
    Handles JualDGV.RowsAdded  
    JualDGV.Item(0, e.RowIndex).Value = CInt(e.RowIndex + 1)  
End Sub
```

Gambar 10-9. Pesan kesalahan

Jika Anda perhatikan pada pesan kesalahan tersebut, dan letakkan mouse Anda di sana, akan terlihat sedikit pesan yang lebih jelas.

```
Private Sub JualDGV_RowsAdded(sender As Object, _  
    e As DataGridViewRowsAddedEventArgs) _  
    Handles JualDGV.RowsAdded  
    JualDGV.Item(0, e.RowIndex).Value = Handles clause requires a WithEvents variable defined in the  
End Sub
```

Gambar 10-10. Penjelasan pesan kesalahan

Dijelaskan di sana bahwa clause **Handles** memerlukan adannya variable **WithEvents** didefinisikan. Hal ini disebabkan karena variabel **JualDGV** dibuat melalui program, padahal di depan pada saat deklarasi, Anda deklarasikan **JualDGV** secara biasa.

```
Private JualDGV AsNewDataGridView
```

Karena itu, ubahlah sedikit deklarasi variable tersebut dengan menambahkan **WithEvents** di sana.

```
Private WithEvents JualDGV AsNewDataGridView
```

Dengan penambahan itu, kesalahan tidak akan ditampilkan lagi, dan Anda bisa mencoba form Anda dijalankan.

Kode	Nama	Qty	Unit	Harga	Sub-Total
1					

Gambar 10-11. Nilai awal baris baru

Sekarang, perhatikan terlebih dahulu form Anda, dan bisa Anda lihat di bagian bawah ada tiga buah **TextBox**. Dua berukuran besar, dan satu berukuran biasa.

TextBox besar yang sebelah kiri akan menampung jumlah item yang hendak dibeli oleh customer, dan **TextBox** besar sebelah kanan akan menampung item yang akan dibeli, tergantung focus pada **DataGridView**. Sedangkan **TextBox** kecil di bawah akan menampung total nilai **Rupiah** dari semua item barang yang akan dibeli.

Untuk mengisikan nilai pada ketiga **TextBox** tersebut, Anda akan menuliskan sebuah sub-routine, bernama **refreshView()**.

```
PrivateSub refreshView(Big_Name AsString, curRow AsInteger)
Dim i AsInteger
Dim tot AsInteger = 0
Dim totqty AsInteger = 0

    JualDGV.Item(0, curRow).Value = CInt(curRow + 1)
    tBig_Name.Text = Trim(Big_Name)

    For i = 0 To JualDGV.RowCount - 1
        totqty = totqty + CInt(JualDGV.Item(3, i).Value)
        tot = tot + CInt(JualDGV.Item(6, i).Value)
        JualDGV.Item(0, i).Value = CInt(i + 1)
    Next
    tTotal.Text = FormatNumber(tot, 0)
    tBig_Qty.Text = totqty
EndSub
```

Sub-routine ini akan mengambil dua buah parameter, yaitu nama barang yang mau ditampilkan, dan sebuah variable tipe **DataRow**. Sub-routine ini akan dipanggil dari sub-routine event yang akan Anda buat selanjutnya, jadi untuk saat ini belum bisa dilakukan pengujian.

Selanjutnya, Anda akan memeriksa setiap masukan pada kolom yang bisa mendapat masukan dalam **JualDGV**. Ada tiga buah kolom yang bisa diedit, yaitu kolom **Kode**, kolom **Qty**, dan kolom **Harga**. User harus bisa memasukkan nilai Kode Barang agar form ini bisa dilakukan. Pada saat pengisian, apabila nilai yang dimasukkan adalah benar, maka secara otomatis nilai pada kolom **Qty** menjadi **1**, dan pada kolom **Harga** akan dituliskan nilai dari field **Brg_hpp** ditambah dengan margin **20%**.

Sebagai awal, deklarasikan sebuah variable **DataRow** untuk membantu proses di awal Class.

```
Dim kodeRow() AsDataRow
```

Kemudian, tuliskan sub-routine berikut:

```
PrivateSub JualDGV_CellValidating(sender AsObject, _
    e AsDataGridViewCellValidatingEventArgs) _
    Handles JualDGV.CellValidating
Dim goodRow AsBoolean = False

If e.FormattedValue IsNot "" Then
    SelectCase JualDGV.Columns(e.ColumnIndex).Name
    Case "Kode"
        kodeRow = ds.Tables("Barang").Select("Brg_id = '" &_
            e.FormattedValue & "'")
    If kodeRow.Count = 0 Then
        kodeRow = ds.Tables("Barang").Select("Brg_barcode = '" &
```

```

        & e.FormattedValue & ""))
If kodeRow.Count = 0 Then
    MsgBox("Kode barang salah.")
    e.Cancel = True
Else
    JualDGV.Item(e.ColumnIndex, e.RowIndex).Value = _
        Trim(kodeRow(0)("Brg_id"))
    goodRow = True
EndIf
Else
    goodRow = True
EndIf
Case "Qty"

Case "Harga"

EndSelect
EndIf
If goodRow Then
    JualDGV.Item(e.ColumnIndex + 1, e.RowIndex).Value = _
    Trim(kodeRow(0)("Brg_nama"))
    JualDGV.Item(e.ColumnIndex + 2, e.RowIndex).Value = _
    IIf(CInt(kodeRow(0)("Brg_det_stock")) - 1 > 0, 1, 0)
    JualDGV.Item(e.ColumnIndex + 3, e.RowIndex).Value = _
    kodeRow(0)("Brg_unit")
    JualDGV.Item(e.ColumnIndex + 4, e.RowIndex).Value = _
    kodeRow(0)("Brg_hpp") * 1.2
    JualDGV.Item(e.ColumnIndex + 5, e.RowIndex).Value = _
    CInt(JualDGV.Item(3, e.RowIndex).Value) * -
    CInt(JualDGV.Item(5, e.RowIndex).Value)
    JualDGV.Item(e.ColumnIndex + 6, e.RowIndex).Value = _
    CInt(kodeRow(0)("Brg_det_stock"))
    tBig_Name.Text = Trim(kodeRow(0)("Brg_nama"))
EndIf
IfNot (IsNothing(e)) Then
    refreshView(JualDGV.Item(2, e.RowIndex).Value, e.RowIndex)
EndIf
EndSub

```

Ada tiga kolom yang mau diperiksa, yaitu **Kode**, **Qty** dan **Harga**. Untuk itu, akan digunakan suatu block **Select Case**, dengan **Case** untuk masing-masing nilai kolom.

Namun, untuk memastikan bahwa data yang dimasukkan memberikan data item barang yang benar, di bagian awal Anda deklarasikan sebuah variable **Boolean** untuk membantu, yaitu variable **goodRow**, dengan nilai awal **False**.

Kondisi di awal memastikan bahwa nilai yang diberikan oleh form adalah nilai yang valid.

```
If e.FormattedValue IsNot ""Then
```

Kemudian diperiksa posisi kolom saat itu:

```
SelectCase JualDGV.Columns(e.ColumnIndex).Name  
Case "Kode"
```

Di sini digunakan object property **e.ColumnIndex**. Jika kolom adalah benar Kode, maka proses dijalankan.

Proses cukup sederhana. Cari dari table **Barang**, apakah benar ada item barangnya, dari nilai **Kode** yang diinput. Jika benar, maka variable **goodRow** menjadi True, dan jika tidak ada data, maka ditampilkan sebuah **msgbox**, dan **goodRow** akan tetap menjadi False.

Begitu untuk block **Case** untuk **Kode**, selanjutnya apa yang dilakukan di luar block **Select Case**. Di sini dilakukan penanganan apabila variable **goodRow** bernilai **True**, yaitu apabila data yang dimasukkan memang valid.

Masing-masing kolom akan diberikan nilainya, termasuk kolom terakhir, **CurStock** yang tersembunyi.

Dan di bagian akhir, dibuat pemanggil sub-routine **refreshView()**.

Jalankan program Anda untuk melihat hasilnya.

Kode	Nama	Qty	Unit	Harga	Sub-Total
1 yn-br	T Yuan Biru	1	PCS	63000	63000

1	T Yuan Biru	63.000
---	-------------	--------

Gambar 10-12. Detail yang sudah hampir lengkap

Sekarang tambahkan kedua **Case** berikutnya, yaitu untuk **Qty** dan **Harga**.

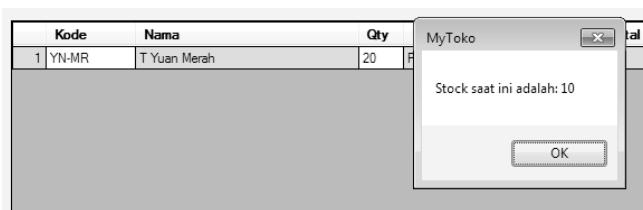
```
Case "Qty"  
Dim tempQty AsInteger  
    tempQty = CInt(kodeRow(0)("Brg_det_stock")) - _  
        CInt(e.FormattedValue)  
If tempQty < 0 Then  
    MsgBox("Stock saat ini adalah: "&_  
        kodeRow(0)("Brg_det_stock"))  
    e.Cancel = True  
EndIf
```

```

If JualDGV.Item(5, e.RowIndex).Value <> 0 Then
    JualDGV.Item(6, e.RowIndex).Value = -
    CInt(JualDGV.Item(3, e.RowIndex).Value) * -
    CInt(JualDGV.Item(5, e.RowIndex).Value)
EndIf
Case "Harga"
IfNot (IsNothing(JualDGV.Item(3, e.RowIndex).Value)) Then
    JualDGV.Item(6, e.RowIndex).Value = -
    CInt(JualDGV.Item(3, e.RowIndex).Value) * -
    CInt(JualDGV.Item(5, e.RowIndex).Value)
EndIf

```

Jika kolom yang baru diedit adalah kolom **Qty** maka dilakukan pengujian stock barang. Apabila hasil pengurangan nilai stock dengan Qty jual kurang dari **0**, yang berarti barang tidak cukup, maka akan ditampilkan sebuah pesan kesalahan yang memberitahukan stock saat itu, dan focus akan dikembalikan pada kolom tersebut.



Gambar 10-13. Pesan kesalahan pada Stock

Namun jika benar, maka nilai dari kolom **Qty** akan dikalikan dengan nilai dari kolom **Harga**, dan dituliskan pada kolom **Sub-Total**.

Jadi sekarang baris dalam **JualDGV** sudah bisa berfungsi dengan baik. Namun masih ada yang bisa Anda lakukan, yaitu memberikan format yang baik pada beberapa kolomnya, terutama kolom yang menampilkan data harga.

Anda bisa menggunakan event **CellFormatting** untuk itu. Tuliskan subroutine berikut:

```

PrivateSub JualDGV_CellFormatting(sender AsObject, _
    e AsDataGridViewCellFormattingEventArgs) _
    Handles JualDGV.CellFormatting
If e IsNot NothingThen
SelectCaseMe.JualDGV.Columns(e.ColumnIndex).Name
Case "Kode"
If e.Value IsNot NothingThen
    e.Value = UCase(e.Value)
EndIf
Case "Harga"
If e.Value IsNot NothingThen
    e.Value = FormatNumber(e.Value, 0)
EndIf

```

```

Case "Sub-Total"
If e.Value IsNot Nothing Then
    e.Value = FormatNumber(e.Value, 0)
EndIf
EndSelect
EndIf
EndSub

```

Jika kolom **Kode**, buat supaya nilai diberi format huruf kapital. Jika kolom **Harga** dan **Sub-Total**, berikan format numerik dengan **0** angka di belakang koma.

Bisa Anda lihat dalam hasilnya, sedikit berbeda dengan tampilan sebelumnya.

Kode	Nama	Qty	Unit	Harga	Sub-Total
1 YN-BR	T Yuan Biru	1	PCS	63,000	63,000

Gambar 10-14. Cell dengan format yang benar

Yang berikutnya adalah penanganan dua buah event pada item **JualDGV**, yaitu event **RowEnter** (yang akan dijalankan pada saat focus tiba pada suatu baris), dan event **PreviewKeyDown** (yang akan dijalankan pada saat suatu tombol keyboard ditekan). Event **PreviewKeyDown** akan digunakan untuk memeriksa penekanan tombol **Enter** dan **Tab**. Jika **Tab**, hanya akan memutar focus dalam satu baris. Namun, tombol Enter akan menyebabkan suatu baris baru dibuat, selama tidak ada baris kosong dalam item **DataGrid** tersebut.

```

Private Sub JualDGV_PreviewKeyDown(sender As Object, e
As PreviewKeyDownEventArgs) Handles JualDGV.PreviewKeyDown
If JualDGV.Rows.GetRowCount(False) > 0 Then
Select Case e.KeyCode
Case Keys.Enter
If JualDGV.Item(6, JualDGV.CurrentCell.RowIndex).Value _ 
    <> 0 And JualDGV.CurrentCell.RowIndex = _
    JualDGV.Rows.GetRowCount(False) - 1 Then
    JualDGV.Rows.Add()
    JualDGV.Item(1, _
        JualDGV.CurrentCell.RowIndex).Selected _ 
        = True
EndIf

```

```

CaseKeys.Tab
If JualDGV.CurrentCell.ColumnIndex = 6 Then
    JualDGV.Item(0, _
        JualDGV.CurrentCell.RowIndex).Selected =
        True
EndIf
EndSelect
EndIf
EndSub

PrivateSub JualDGV_RowEnter(sender AsObject, e
AsDataGridViewCellEventArgs) Handles JualDGV.RowEnter
IfNot (IsNothing(JualDGV.Item(2, e.RowIndex).Value)) Then
    refreshView(JualDGV.Item(2, e.RowIndex).Value, e.RowIndex)
EndIf
EndSub

```

RowEnter hanya akan melakukan refresh tampilan **DataGrid**, namun **PreviewKeyDown** akan melakukan proses yang lebih panjang. Event tersebut akan memeriksa keadaan form, dan akan menambahkan sebuah baris baru jika data sebelumnya adalah valid.

Untuk DataGrid, event-event yang dibuat sudah cukup. Sekarang akan membahas apa yang harus dilakukan pada saat user menekan tombol.

Pertama, tombol **btnHapus**. Jika user menekan tombol tersebut, semua data harus dihapus dari baris yang dipilih. Namun perlu diingat bahwa user bisa saja tidak sengaja menekan tombol tersebut, karena itu, ada baiknya diberikan pertanyaan terlebih dahulu pada user.

```

PrivateSub btnHapus_Click(sender AsObject, e AsEventArgs)
Handles btnHapus.Click
If JualDGV.RowCount > 1 And_
JualDGV.CurrentRow.Cells(1).Value IsNotNothingThen
IfMessageBox.Show(_
"Apakah baris item benar akan dihapus? "&_
Chr(13) &"#"&_
JualDGV.CurrentRow.Cells(0).Value _ 
& : "&_
UCase(JualDGV.CurrentRow.Cells(1).Value), _ 
"Hapus", MessageBoxButtons.YesNo, _ 
MessageBoxIcon.Warning) = DialogResult.Yes Then
    JualDGV.Rows.RemoveAt(_
JualDGV.CurrentRow.Cells(0).RowIndex)
If JualDGV.RowCount = 0 Then
    JualDGV.Rows.Add()
EndIf
    tTotal.Text = ""
    tBig_Qty.Text = ""
    tBig_Name.Text = ""
    MsgBox("Data dihapus.")
EndIf
EndIf
EndSub

```

Jika **DataGrid** berisi lebih dari **0** baris, maka baris yang terpilih bisa dihapus. Namun, jika baris hanya satu, dan baris tersebut dihapus, maka akan dibuat sebuah baris baru lagi.

Sekarang bagaimana kalau tombol **btnBaru** yang ditekan. Di sini, penekanan berarti akan membuat sebuah form baru. Apabila form sudah diisi, maka semua keadaan harus dikembalikan pada keadaan semula. Tentu saja, hal ini perlu ditanyakan juga pada user.

```
PrivateSub btnBaru_Click(sender AsObject, e AsEventArgs)
Handles btnBaru.Click
If MessageBox.Show("Apakah benar akan menghapus data?", "Refresh", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = DialogResult.Yes Then
    getJual_id()
    tJual_tgl.Text = ""
    tJual_note.Text = ""
    JualDGV.Rows.Clear()
    tTotal.Text = ""
    tBig_Qty.Text = ""
    tBig_Name.Text = ""
    cbCust.SelectedIndex = 0

    SetupDGV()
    setupLayout()
EndIf
EndSub
```

Di sini, kembalikan semua nilai pada nilai awal, termasuk tanggal maupun **ComboBox**.

Yang terakhir adalah penanganan tombol **btnSelesai**. Selesai di sini berarti bahwa pengisian data penjualan sudah selesai, dan semua sudah dicek dan semua valid. Karena itu, simpan data dalam table. Ada tiga table yang perlu dilakukan update, yaitu table **Jual**, table **Jual_details**, dan table **Barang_details**, untuk meng-update nilai stock setelah terjadi penjualan.

Dalam sub-routine ini, Anda akan menyusun statement **SQL** untuk melakukan update terhadap ketiga table tersebut.

Namun, harap Anda ingat juga, bagaimana apabila sebelum user mengganti parameter apa pun dalam form itu, user langsung menekan tombol **btnSelesai**? Anda juga harus memeriksa kondisi seperti itu.

Untuk itu, akan digunakan sebuah variable **Boolean**, yaitu variable **inEditMode**. Deklarasikan variable ini di atas, dan berikan nilai **False**.

```
Dim inEditMode AsBoolean = False
```

Bagaimana memberikan nilai **True** pada variable ini? Secara logika, segala perubahan yang terjadi pada form ini, mengganti tanggal, mengisi

Note, mengganti Customer, semua akan menyebabkan status berubah. Karena itu berikan pemberian status **True** pada lokasi-lokasi berikut.

Sub-Routine setupLayout()

```
PrivateSub setupLayout()
IfNot (inEditMode) Then
Dim i AsInteger
Dim dsRow AsDataRow
    table.Columns.Add("Cust_nama", GetType(String))
    table.Columns.Add("Cust_id", GetType(Integer))
For i = 0 To ds.Tables("Customer").Rows.Count - 1
    dsRow = ds.Tables("Customer").Rows(i)
    addrow(dsRow.Item("Cust_nama"), dsRow.Item("Cust_id"))
Next
EndIf
    cbCust.DataSource = table
    cbCust.DisplayMember = "Cust_nama"
    cbCust.ValueMember = "Cust_id"

dtJual_tgl.Format = DateTimePickerFormat.Custom
dtJual_tgl.CustomFormat = "dd-MMM-yy"
    dtJual_tgl.Value = Today

    tJual_id.Text = getJual_id()

    inEditMode = False
EndSub
```

Event JualDGV_CellValidating

```
If goodRow Then
    JualDGV.Item(x + 1, y).Value = ..
    JualDGV.Item(x + 2, y).Value = ..
    JualDGV.Item(x + 3, y).Value = ..
    JualDGV.Item(x + 4, y).Value = ..
    JualDGV.Item(x + 5, y).Value = ..
    JualDGV.Item(x + 6, y).Value = ..
    tBig_Name.Text = ..
inEditMode = True
EndIf
```

Event btnBaru_Click

```
PrivateSub btnBaru_Click(sender AsObject, e AsEventArgs)
Handles btnBaru.Click
If inEditMode Then
If MessageBox.Show("Apakah benar akan menghapus data?", "Refresh", MessageBoxButtons.YesNo, MessageBoxIcon.Question) =
DialogResult.Yes Then
    getJual_id()
    tJual_tgl.Text = ""
    tJual_note.Text = ""
    JualDGV.Rows.Clear()
```

```

    tTotal.Text = ""
    tBig_Qty.Text = ""
    tBig_Name.Text = ""
    cbCust.SelectedIndex = 0

    SetupDGV()
    setupLayout()
    inEditMode = False
EndIf
End If
EndSub

```

Kemudian, buatlah tiga buah sub-routine baru, masing-masing untuk menangani perubahan nilai pada item **tJual_id**, **tJual_note**, dan **tCust_email**.

Event TextChanged

```

PrivateSub tJual_note_TextChanged(sender AsObject, e
AsEventArgs) Handles tJual_note.TextChanged
    inEditMode = True
EndSub

PrivateSub tJual_id_TextChanged(sender AsObject, e
AsEventArgs) Handles tJual_id.TextChanged
    inEditMode = True
EndSub

PrivateSub tCust_email_TextChanged(sender AsObject, e
AsEventArgs) Handles tCust_email.TextChanged
If Trim(tCust_email.Text) <> "-" Then
    inEditMode = True
EndIf
EndSub

```

Sekarang Anda bisa masuk pada sub-routine utama yang akan menyimpan data ke dalam database. Namun, sebelumnya masih ada sebuah fungsi yang belum dituliskan, yaitu fungsi untuk memberikan nilai awal pada field Jual_det_id dari table Jual_details. Fungsi ini cukup serupa dengan fungsi getJual_id(), dan akan disebut getJual_det_id().

```

PrivateFunction getJual_det_id() AsInteger
Dim intID AsInteger = 0
Dim tJual_details AsInteger

If IsDBNull(ds.Tables("Jual_details").Rows(0). _
Item("Jual_det_id")) Then
    intID = 0
Else
    tJual_details = _
    CInt(ds.Tables("Jual_details").Rows(0).Item(0))
    intID = tJual_details
EndIf

```

```
Return intID  
EndFunction
```

Tuliskan terlebih dahulu sub-routine berikut secara lengkap.

```
PrivateSub btnSelesai_Click(sender AsObject, e AsEventArgs)  
Handles btnSelesai.Click  
Dim sql0, sql1 AsString  
Dim i AsInteger  
Dim intJual_det_id AsInteger = getJual_det_id()  
Dim objCmd AsOleDbCommand  
Dim curId AsInteger = 1  
  
If inEditMode Then  
If JualDGV.RowCount > 0 And_  
    JualDGV.Item(1, 0).Value IsNotNothingThen  
        sql0 = "INSERT INTO Jual (Jual_id, Jual_tgl, "& _  
            "Cust_id, Jual_note, User_id) VALUES "  
        sql1 = "("  
        sql1 = sql1 & "'"& tJual_id.Text &"', "  
        sql1 = sql1 & "'"&  
        (CDate(tJual_tgl.Text)).ToString("s") &"', "  
        sql1 = sql1 &CInt(cbCust.SelectedValue.ToString) & ", "  
        sql1 = sql1 & "'"& tJual_note.Text &"', "  
        sql1 = sql1 & curId & ")"  
  
        objCmd = NewOleDbCommand(sql0 & sql1, con)  
        objCmd.ExecuteNonQuery()  
  
        objCmd = Nothing  
  
        sql0 = "INSERT INTO Jual_details (Jual_det_id, "& _  
    "Jual_id, Jual_det_qty, Jual_det_harga, "& _  
    "Brg_id) VALUES "  
    For i = 0 To JualDGV.RowCount - 1  
    If JualDGV.Item(1, i).Value IsNotNothingThen  
        sql1 = "("  
        sql1 = sql1 & intJual_det_id + _  
        CInt(JualDGV.Item(0, i).Value) & ", "  
        sql1 = sql1 & "'"& tJual_id.Text &"', "  
        sql1 = sql1 &CInt(JualDGV.Item(3, i).Value) & ", "  
        sql1 = sql1 &CInt(JualDGV.Item(5, i).Value) & ", "  
        sql1 = sql1 & "'"&  
        UCase(JualDGV.Item(1, i).Value) & "' )"  
  
        objCmd = NewOleDbCommand(sql0 & sql1, con)  
        objCmd.ExecuteNonQuery()  
    EndIf  
    Next  
  
    objCmd = Nothing  
  
    sql0 = "UPDATE Barang_details SET Brg_det_stock = "  
    For i = 0 To JualDGV.RowCount - 1  
    If JualDGV.Item(1, i).Value IsNotNothingThen  
        sql1 = CInt(JualDGV.Item(7, i).Value) - _  
        CInt(JualDGV.Item(3, i).Value) & " WHERE "  
        sql1 = sql1 &"Brg_id = '"&_
```

```

UCase(JualDGV.Item(1, i).Value) & " ! "
    objCmd = NewOleDbCommand(sql0 & sql1, con)
    objCmd.ExecuteNonQuery()
EndIf
Next

objCmd = Nothing
MsgBox("Data telah disimpan .")
EndIf
EndIf
Me.Close()
EndSub

```

Di bagian awal, pada saat pendeklarasian tentukan sebuah variable sebagai kunci dari table **Jual_details**, dengan cara memanggil fungsinya.

```
Dim intJual_det_id AsInteger = getJual_det_id()
```

Kemudian dilakukan pemeriksaan kondisi **inEditMode**, dan setelah melewatinya, dilakukan pemeriksaan kondisi kedua untuk memastikan data baris adalah valid.

Setelah melewati pemeriksaan kedua ini, baru dimulai proses **SQL** yang pertama, yaitu untuk table **Jual**.

Persiapkan **SQL** statement yang mau dijalankan, kemudian setelah disimpan dalam variable **OleDbCommand**, maka kemudian dijalankan dengan menggunakan method**OleDbCommand.ExecuteNonQuery()**.

Table **Jual** akan menyimpan data header dari transaksi penjualan tersebut, termasuk di dalamnya, No ID Penjualan (**Jual_id**), Tanggal Penjualan (**Jual_tgl**), Customer ID (**Cust_id**), Catatan Penjualan (**Jual_note**), dan User ID (**User_id**).

Setelah berhasil dijalankan, variable **OleDbCommand** tersebut dihapus dari memory, dan kemudian proses dilanjutkan dengan table kedua, yaitu **Jual_details**.

Table ini akan menyimpan data detail dari transaksi penjualan tersebut, dan termasuk di dalamnya adalah No ID Detail Penjualan (**Jual_det_id**, nomor urut), No ID Penjualan sebagai kunci (**Jual_id**), Jumlah Item Penjualan (**Jual_det_qty**), Harga Item Penjualan (**Jual_det_harga**), dan ID Barang (**Brg_id**).

Begini seterusnya sampai dengan table ketiga, yaitu table **Barang_details**. Dalam table ini, akan diupdate status stock dari barang yang terjual.

Dalam akhir pembahasan form ini, perhatikan dua baris berikut. Satu baris pada bagian deklarasi variable:

```
Dim curId AsInteger = 1
```

Dan satu baris lagi pada statement penyimpan untuk table **Jual**:

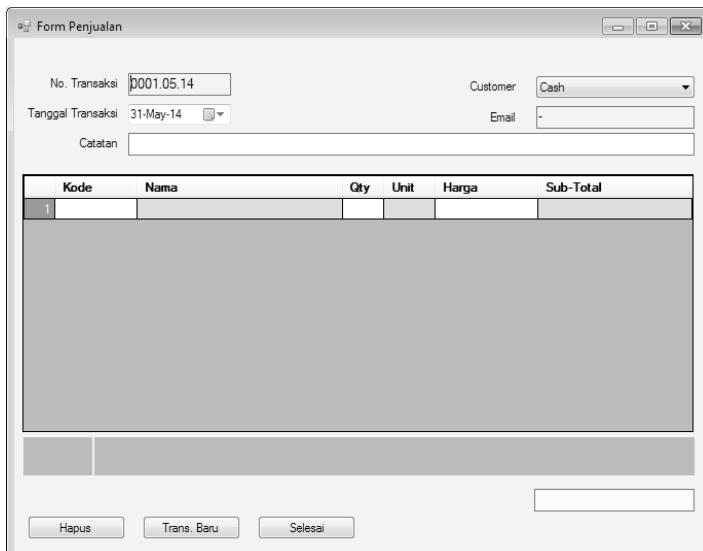
```
sql0 = "INSERT INTO Jual (Jual_id, Jual_tgl, " & _
       "Cust_id, Jual_note, User_id) VALUES " &_
       "(" &_
       sql1 & "" & tJual_id.Text & ", " &_
       sql1 & "" & -_
       (CDate(tJual_tgl.Text)).ToString("s") & ", " &_
       sql1 & CInt(cbCust.SelectedValue.ToString) & ", " &_
       sql1 & "" & tJual_note.Text & ", " &_
       sql1 & curId & ")"
```

Di sini ada sebuah variable, **curID**, yang mendapatkan nilai awal = **1**, dan dipakai dalam proses **INSERT** pada table **Jual**, dan menggantikan nilai untuk field **User_id**.

Variable ini digunakan untuk mencatat user yang menjalankan aplikasi ini. Pencatatan ini dilakukan melalui Form Login, yang akan dibahas setelah ini. Pencatatan **User_id** cukup penting dalam aplikasi ini, karena di sini bisa dicatat **SPG** yang berjaga, misalnya.

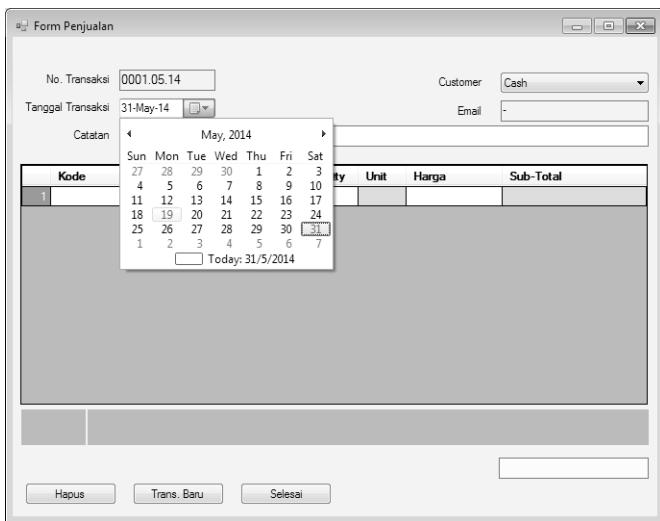
Dan karena saat ini form Login belum ada, maka untuk sementara nilai dari **curId** tersebut diatur sebagai konstan pada saat deklarasi.

Tampilan akhir dari form Penjualan ini adalah sebagai berikut:



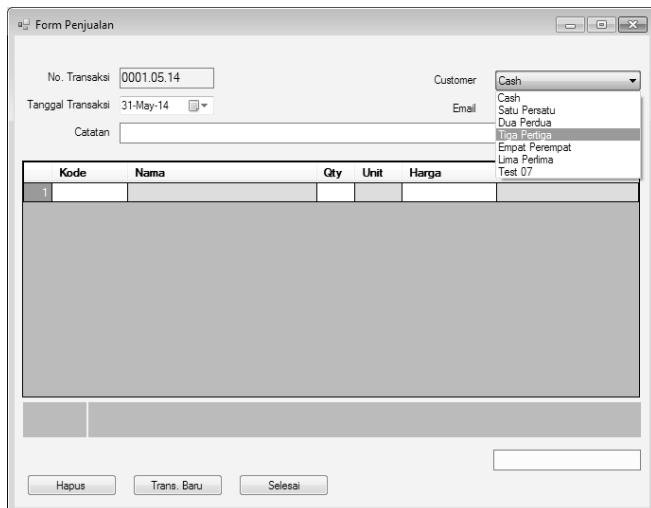
Gambar 10-16. Tampilan Form Jual

Memilih tanggal dari object **DateTimePicker**:



Gambar 10-17. Object DateTimePicker

Memilih dari ComboBox Customer:



Gambar 10-18. ComboBox Customer

Dilanjutkan dengan pengisian data barang, sebelum penekanan tombol **Enter** atau **Tab**:

The screenshot shows a Windows application window titled "Form Penjualan". The interface includes fields for transaction number ("No. Transaksi: 0001.05.14"), date ("Tanggal Transaksi: 31-May-14"), customer ("Customer: Tiga Pertiga"), and email ("Email: tiga_p@email.com"). A note field ("Catatan: Test Penjualan") is also present. Below these is a table with columns: Kode, Nama, Qty, Unit, Harga, and Sub-Total. A single row is visible with the code "1" and the name "yn-br". The quantity field contains "0". At the bottom are buttons for "Hapus", "Trans. Baru", and "Selesai".

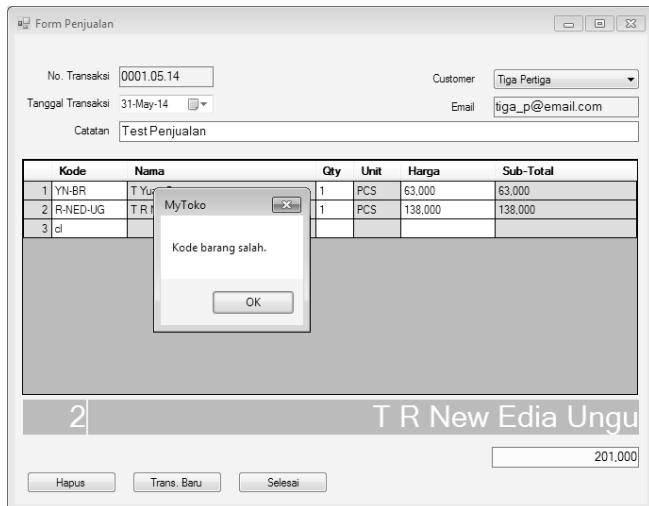
Gambar 10-19. Pengisian data barang

Setelah menekan tombol **Enter**, maka data barang, jika ada, akan ditampilkan dalam form tersebut.

The screenshot shows the same "Form Penjualan" window after an item has been entered. The table now displays a single row for "T Yuan Biru" with code "YN-BR", quantity "1", unit "PCS", price "63.000", and sub-total "63.000". The quantity field now contains "1". The note field still says "Test Penjualan". The bottom buttons remain the same: "Hapus", "Trans. Baru", and "Selesai".

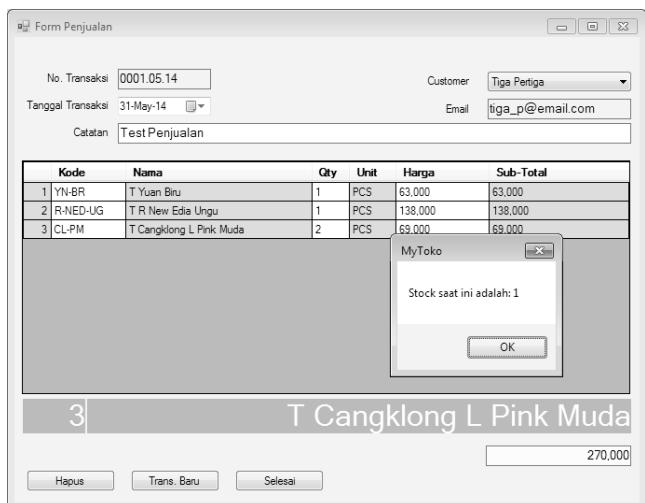
Gambar 10-20. Data barang secara lengkap

Namun, kalau data kode yang dimasukkan tidak benar:



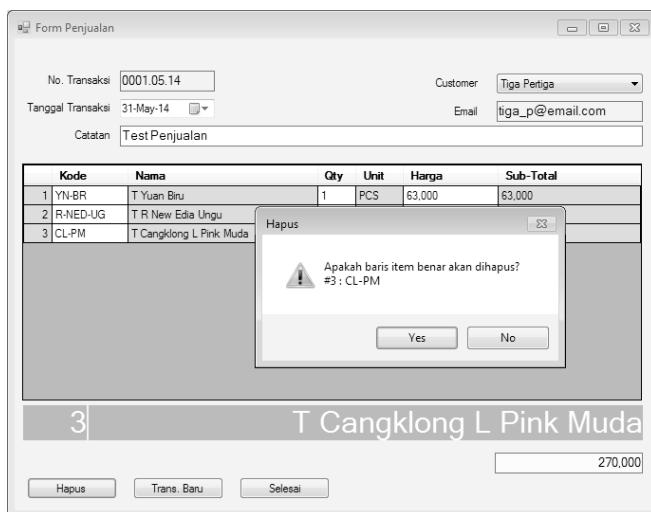
Gambar 10-21. Pesan kesalahan pada kolom Kode

Dan jika memasukkan data jumlah barang yang lebih besar dari status stock saat itu:



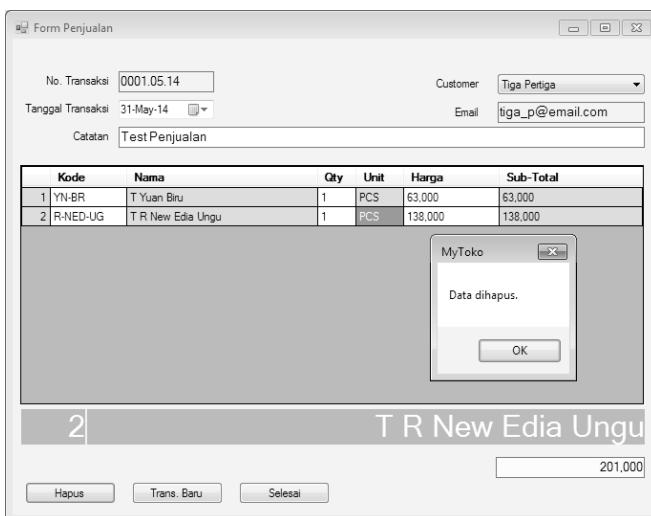
Gambar 10-22. Pesan kesalahan pada kolom Qty

Jika Anda menekan tombol Hapus pada baris data baru tersebut:



Gambar 10-23. Apakah baris item benar akan dihapus?

Dan dijawab dengan benar:



Gambar 10-24. Data baris dihapus

Dan setelah Anda menekan tombol Selesai:

Kode	Nama	Qty	Unit	Harga	Sub-Total	
1	YN-BR	T Yuan Biru	1	PCS	63,000	63,000
2	R-NED-UG	T R New Edia Ungu	1	PCS	138,000	138,000

Gambar 10-25. Data disimpan dalam tiga buah table

Demikianlah pembahasan untuk Form **Jual** ini. Setelah ini, Anda akan berlanjut mempelajari dua form terakhir dari aplikasi ini.

BAB 11

Form Login dan Form Utama

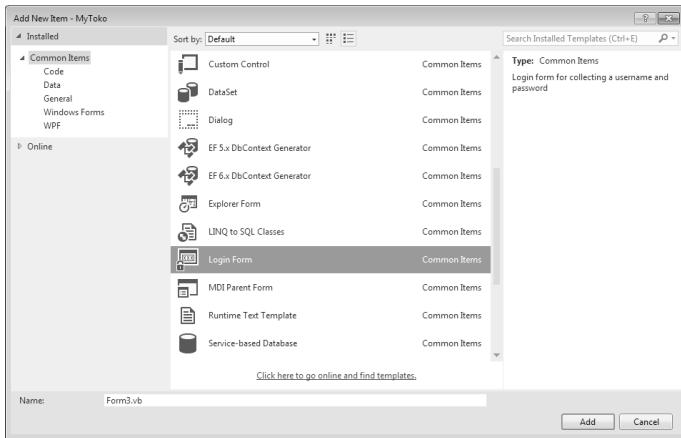
Form Login adalah form pembuka dalam aplikasi ini. Pada saat aplikasi mulai dijalankan, maka yang aktif terlebih dahulu adalah form ini, karena dari form ini, bisa diketahui siapa yang melakukan **Log-in**. Ada beberapa modul yang hanya bisa dilakukan oleh user yang memiliki privilege lebih tinggi daripada privilege yang dimiliki oleh seorang **SPG**.

Jika user sudah melakukan login, maka data login user tersebut akan dicatat (salah satunya dalam variable **curId**), dan akan terus dibawa ke bagian selanjutnya, yaitu dengan membuka Form **Utama**. Form **Utama** inilah yang akan menampung semua form yang lain, dan mengatur semuanya dari ini.

Dalam pembahasan ini, hanya diberikan dua macam privilege, dengan kode nilai **0** dan **1**. Kode **1** untuk privilege normal, di mana user tidak bisa melakukan proses **Import Data**, dan atau memanipulasi data dasar. Privilege dengan kode **0** akan bisa melakukan proses apa pun. Dalam pengembangannya, salah satu yang bisa dicoba adalah membagi-bagi kemampuan user untuk melakukan suatu tugas dengan batasan privilege semacam ini.

Form Login

Dalam pembuatan sebuah form Login, Anda bisa menggunakan template form Login yang sudah disediakan dalam **Visual Studio Express 2013**. Anda bisa memanggilnya dengan menambahkan sebuah **Windows Form** baru pada **Project** Anda, dan kemudian memilih template **Login Form** dari daftar.



Gambar 11-1. Template Login Form

Setelah terbuka, Anda bisa melihat format dasar dari form ini sudah berupa form Login yang siap digunakan.



Gambar 11-2. Tampilan template form Login

Sedangkan apabila Anda melihat pada tampilan programnya:

```
Public Class Form3
    ' TODO: Insert code to perform custom authentication using the provided username and password
    ' (See http://go.microsoft.com/fwlink/?LinkId=35539).
    ' The custom principal can then be attached to the current thread's principal as follows:
    ' My.User.CurrentPrincipal = CustomPrincipal
    ' where CustomPrincipal is the IPrincipal implementation used to perform authentication.
    ' Subsequently, My.User will return identity information encapsulated in the CustomPrincipal object
    ' such as the username, display name, etc.

    Private Sub OK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK.Click
        Me.Close()
    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel.Cl
        Me.Close()
    End Sub
End Class
```

Gambar 11-3. Tampilan program dari template form Login

Mari dimulai untuk mengerjakan form ini, pertama dengan mengatur tampilannya, dan memberikan nama yang sesuai untuk program Anda.

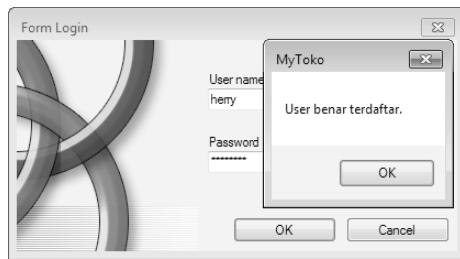
Hanya ada dua buah **TextBox** di sana, ubahlah namanya menjadi **txtUsername** dan **txtPassword**. Biarkan kedua tombol itu bernama **OK** dan **Cancel**.

Jadi, pada saat user menekan tombol **OK**, maka proses validasi akan berlangsung, yaitu dengan cara mengambil data User dari database, khususnya pada table **Users**.

```
PrivateSub OK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OK.Click
Dim dbProvider AsString = "Provider=SQLOleDb;"
Dim dbSource AsString =
"Data Source=chibi\sqlexpress;Initial "& _
"Catalog=dbToko;Integrated Security=SSPI"
Dim con AsNew OleDbConnection
    con.ConnectionString = dbProvider & dbSource
Dim cmd AsOleDbCommand = NewOleDbCommand( _
"SELECT * FROM Users WHERE User_nama = '"& _
txtUsername.Text & "' AND User_pwd = '"&_
txtPassword.Text & "'", con)
    con.Open()
Dim sdr AsOleDbDataReader = cmd.ExecuteReader()
If (sdr.Read() = True) Then
    MsgBox("User benar terdaftar.")
Else
    MessageBox.Show("Username atau password salah!")
EndIf
    con.Close()
    con = Nothing
    cmd = Nothing
EndSub
```

Anda bisa lihat di sini, bahwa proses validasi terhadap database terlihat sedikit berbeda dibandingkan pada form yang lain. Di sini, DataAdapter maupun DataSet tidak digunakan.

Pada dasarnya, form ini melakukan koneksi ke database hanya untuk mengambil data user, jika data sudah terambil, apakah hasilnya benar ataupun tidak, koneksi dibuang.



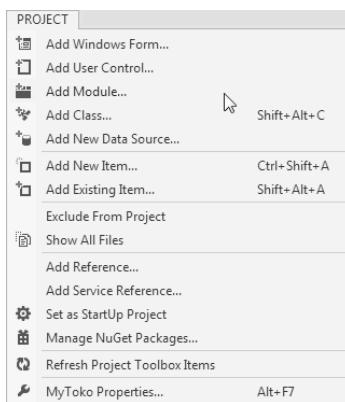
Gambar 11-4. Validasi berhasil

Masih ingatkah Anda pada variable **curId** yang dibahas pada bab sebelum ini? Di sana sedikit dijelaskan bahwa seharusnya nilai variable ini didapat dari form Login. Namun, bagaimana bisa sebuah variable yang dibuat pada form Login, namun akan digunakan pada form yang lain?

Jawabannya adalah dengan menjadikan variable tersebut bersifat **Public**, dan bukannya **Private**.

Untuk membuat variable supaya bersifat **Public**, Anda bisa melakukannya dengan meletakkannya pada sebuah object yang bisa diakses dari semua form. Misalnya pada sebuah **Module**.

Tambahkanlah sebuah module pada project Anda.



Gambar 11-5. Add Module

Dan pada module tersebut, tuliskan beberapa nama variable berikut.

```
ModuleModule1  
Public curUser AsString = ""
```

```
Public curId As Integer  
Public curPriv As Integer = 1  
Public frmUtama As frmUtama  
End Module
```

Dan kemudian letakkan pemberian nilai pada ketiga variable ini pada event **OK_Click**.

```
If (sdr.Read() = True) Then  
  
    curUser = Trim(sdr("User_nama"))  
    curId = sdr("User_id")  
    curPriv = sdr("User_priv")  
  
    MsgBox("User benar terdaftar.")  
Else  
    MessageBox.Show("Username atau password salah!")  
EndIf
```

Demikianlah pemrograman singkat untuk form Login ini. Ada beberapa baris perintah lagi yang perlu dimasukkan di dalamnya, yang berhubungan dengan Form **Utama**. Namun, karena Form **Utama** belum ada, maka ada baiknya sekarang Anda membuat Form **Utama** terlebih dahulu.

Form Utama

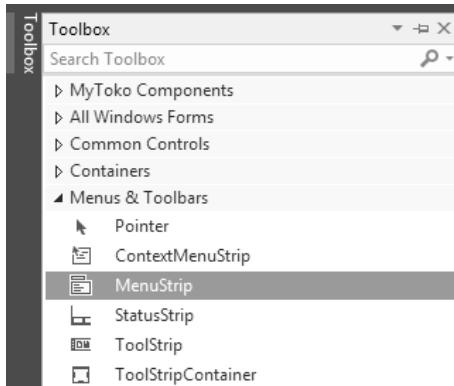
Tambahkanlah sebuah form terakhir, dan beri nama **frmUtama**.



Gambar 11-6. Form Utama

Dalam Form ini, akan digunakan sistem menu, dan bukan dengan sistem tombol, untuk memanggil modul-modul lainnya.

Tambahkan pada form Utama item ToolStrip yang diambil dari Toolbox, pada bagian Menus & Toolbars.



Gambar 11-7. Item ToolStrip dari Toolbox

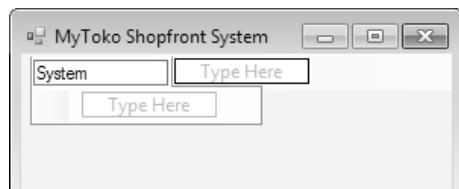
Setelah ditambahkan, keadaan form dan workspace Anda akan berubah menjadi semacam ini:



Gambar 11-8. *MenuStrip* pada workspace

Untuk menyusun menu Anda, perhatikan object di bagian atas form Anda, pada bagian yang tertulis “**Type Here**”.

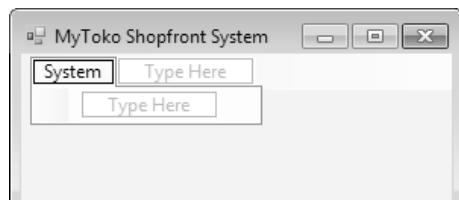
Letakkan cursor Anda di sana, dan tulislah **System**.



Gambar 11-9. Menambahkan menu item

Perhatikan bahwa pada saat Anda meletakkan cursor pada **TextBox** itu, maka tulisan “**Type Here**” ditampilkan lagi di sebelah kanan maupun bagian bawah dari **TextBox** yang pertama.

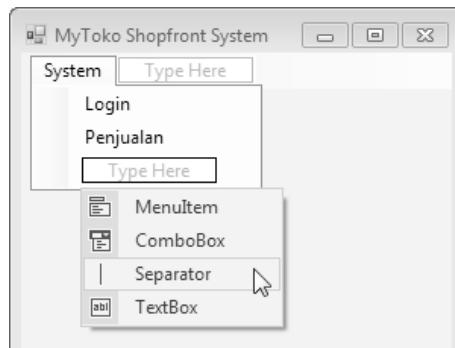
Begitu Anda tekan tombol **Enter**, maka tampilan dari menu Anda akan sedikit berubah.



Gambar 11-10. Tampilan menu

Tambahkan lagi menu item **Login** dan **Penjualan** di bawah menu **System**.

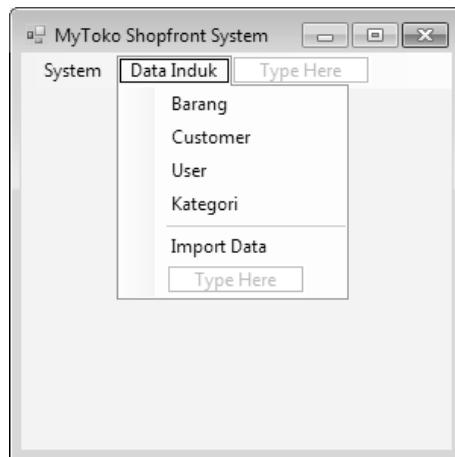
Di bawah **Penjualan**, tekanlah anak panah kecil pada **TextBox** kosong, dan pilihlah **Separator**.



Gambar 11-11. Separator

Setelah garis separator itu, untuk submenu terakhir, tuliskan **Keluar**.

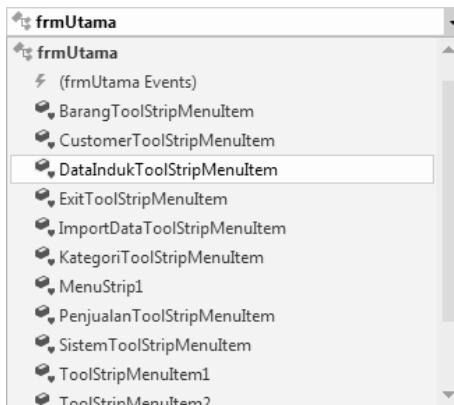
Lanjutkan untuk menyusun menu Anda, di samping **System**, tuliskan **Data Induk**, dan di bawahnya berturut dengan **Barang**, **Customer**, **User**, **Kategori**, sebuah separator, dan **Import Data**.



Gambar 11-12. Submenu Data Induk

Anda bisa menjalankan program Anda, dan Anda bisa melihat susunan menu yang Anda buat. Semua pilihan belum akan melakukan apa pun, karena memang belum Anda program untuk itu. Anda bisa keluar dengan menekan tombol silang berwarna merah pada sudut form Anda.

Pada tampilan program, Anda bisa melihat bagaimana **Visual Basic** akan memberikan nama dari menu dan submenu Anda secara default.

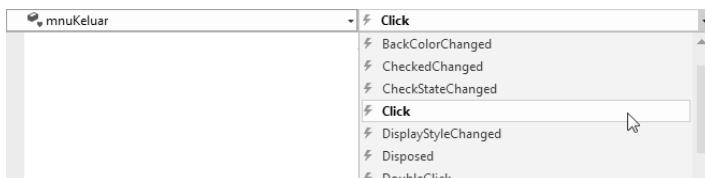


Gambar 11-13. Object menu pada form Anda

Nama default yang telah disediakan secara umum adalah nama yang panjang. Tentu saja Anda bisa mengubahnya menjadi nama yang lebih pendek, atau yang lebih mudah diingat.

Pilihlah submenu **Keluar** dari tampilan menu Anda. Property dari item menu tersebut akan ditampilkan di bagian window Property di sebelah kanan workspace Anda.

Carilah Property **Name**, dan gantilah menjadi, misalnya **mnuKeluar**. Kemudian, carilah event **Click**.



Gambar 11-14. Daftar event untuk object mnuKeluar

Buatlah sub-routine untuk itu.

```
PrivateSub mnuKeluar_Click(sender AsObject, e AsEventArgs)
Handles mnuKeluar.Click
Me.Close()
EndSub
```

Kemudian, buatlah sub-routine untuk semua anggota menu Anda.

```
PrivateSub mnuKeluar_Click(sender AsObject, e AsEventArgs)
Handles mnuKeluar.Click
Me.Close()
EndSub

PrivateSub mnuBarang_Click(sender AsObject, e AsEventArgs)
Handles mnuBarang.Click
If curPriv = 0 Then
frmBarang.Show()
EndIf
EndSub

PrivateSub mnuCustomer_Click(sender AsObject, e AsEventArgs)
Handles mnuCustomer.Click
If curPriv = 0 Then
frmCustomer.Show()
EndIf
EndSub

PrivateSub mnuKategori_Click(sender AsObject, e AsEventArgs)
Handles mnuKategori.Click
If curPriv = 0 Then
frmKategori.Show()
EndIf
EndSub

PrivateSub mnuLogin_Click(sender AsObject, e AsEventArgs)
Handles mnuLogin.Click
frmLogin.Show()
EndSub

PrivateSub mnuPenjualan_Click(sender AsObject, e AsEventArgs)
Handles mnuPenjualan.Click
frmJual.Show()
EndSub

PrivateSub mnuUser_Click(sender AsObject, e AsEventArgs)
Handles mnuUser.Click
If curPriv = 0 Then
frmUsers.Show()
EndIf
EndSub
```

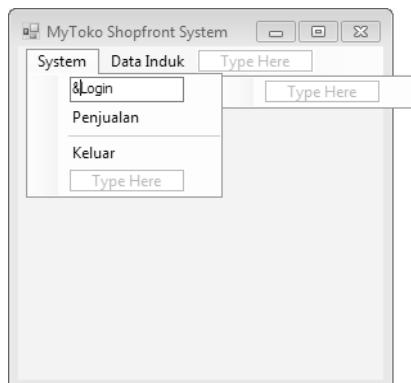
Tambahkan sebuah sub-routine yang akan menangani proses pembukaan form **Utama** ini.

```
PrivateSub frmUtama_Load(sender AsObject, e AsEventArgs)
Handles Me.Load
Me.Text = "My Toko - "& curUser
EndSub
```

Ada lagi yang bisa Anda lakukan untuk menu pada form Anda, yaitu memberikannya shortcut.

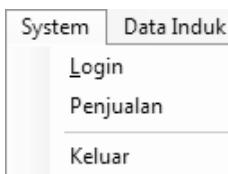
Misalnya, pada menu **Login**, Anda bisa menggarisbawahi huruf **L**, dan nantinya user dapat menggunakan shortcut untuk menjalankan modul Jual tanpa harus menggunakan mouse untuk membuka menu.

Caranya cukup mudah, tempatkan cursor Anda tepat sebelum huruf yang mau Anda garis-bawahi, dalam hal ini **L**, dan ketikkan karakter *ampersand* (**&**).



Gambar 11-15. Karakter untuk menambahkan shortcut pada menu

Jika Anda jalankan form Anda, maka tampilan menu **Login** akan memiliki garis bawah pada karakter **L**.



Gambar 11-16. Shortcut pada menu

Tambahkanlah shortcut untuk semua item menu yang ada:

J pada **Jual**

B pada **Barang**

C pada **Customer**

U pada **User**

K pada Kategori

I pada Import Data

Pilihlah item menu Keluar, dan perhatikan pada bagian Property, khususnya pada bagian ShortcutKeys.

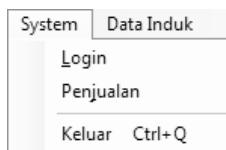
Perhatikan kedua item property **ShortcutKeys** dan **ShowShortcutKeys** di bawahnya. Pada saat ini, **ShortcutKeys** berisi nilai **None**, sedangkan **ShowShortcutKeys** berisi nilai **True**. Yang berarti bahwa untuk item menu **Exit** tersebut, saat ini tidak ada kombinasi shortcut padanya, tapi jika ada, maka kombinasi shortcut tersebut akan ikut ditampilkan.

Karena itu mari buat kombinasi shortcut **Ctrl+Q** untuk item menu tersebut.



Gambar 11-17. Pilihan kombinasi shortcut

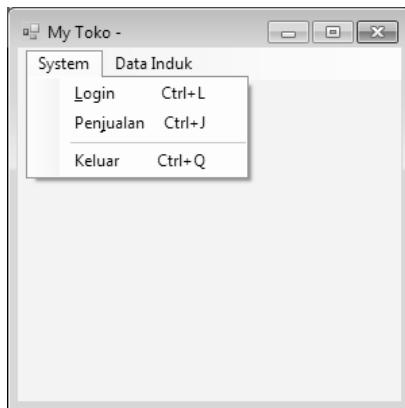
Jika dijalankan, maka menu akan tampak sebagai berikut.



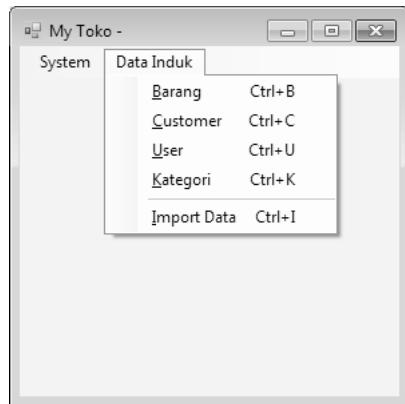
Gambar 11-18. Tampilan item menu Keluar

Jika pada form itu Anda tekan kombinasi **Ctrl+Q**, maka itu akan seperti Anda memilih keluar dari submenu.

Buatlah shortcut keys untuk semua item menu.



Gambar 11-19. Item menu System



Gambar 11-20. Item menu Data Induk

Ingatkah Anda tadi waktu membahas form Login, ada satu bagian yang ditinggalkan karena form **Utama** belum dibuat? Sekarang Anda bisa kembali lagi ke sana, dan sisipkan baris ini pada event **OK_Click**.

```
If (sdr.Read() = True) Then
    Dim fUtama As New frmUtama
        curUser = Trim(sdr("User_nama"))
        curId = sdr("User_id")
        curPriv = sdr("User_priv")
        fUtama.Show()
        fUtama.Refresh()
    Me.Hide()
Else
```

Dan kembali kepada form **Utama**, tambahkan sub-routine penanganan event **FormClosing**, untuk menutup semua window pada saat form **Utama** ditutup.

```
PrivateSub frmUtama_FormClosing(sender AsObject, e  
AsFormClosingEventArgs) HandlesMe.FormClosing  
frmLogin.Close()  
EndSub
```

Demikianlah pembahasan mengenai **Aplikasi Penjualan MyToko** ini. Kembalikan **Startup Form** pada **frmLogin**, dan Anda sudah siap untuk menggunakan aplikasi ini untuk kegunaan operasional toko Anda.

Terus kembangkan aplikasi ini untuk disesuaikan dengan kebutuhan Anda sendiri.

Sebuah tipe data adalah suatu atribut yang menentukan tipe dari data yang bisa dipegang oleh suatu object: misalnya data huruf, data mata uang, data hari dan waktu, dan seterusnya.

Adalah penting untuk menentukan tipe data yang akan digunakan dalam aplikasi Anda, untuk memastikan besar memory yang mungkin akan terpakai.

Dalam **SQL Server**, tipe data dikelompokkan berdasarkan beberapa kategori:

Exact Numerics

Data berupa angka yang pasti yang menggunakan data numeric.

Tipe Data	Range	Storage
Bigint	-2^63 (-9,223,372,036,854,775,808) sampai 2^63-1 (9,223,372,036,854,775,807)	8 bytes
Int	-2^31 (-2,147,483,648) sampai 2^31-1 (2,147,483,647)	4 bytes
Smallint	-2^15 (-32,768) sampai 2^15-1 (32,767)	2 bytes
tinyint	0 sampai 255	1 byte
decimal	1-9 angka di depan dan di belakang titik	5 bytes
	10-19 angka di depan dan di belakang titik	9 bytes
	20-28 angka di depan dan di belakang titik	13 bytes
	29-38 angka di depan dan di belakang titik	17 bytes
bit	0, 1 atau NULL	1 bit

money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
smallmoney	-214,748.3648 to 214,748.3647	4 bytes

Approximate Numerics

Tipe data angka non-presisi yang digunakan dengan data numerik floating point, yaitu suatu nilai yang ditunjukkan dengan .. $\times \text{base}^n$.

<i>Nilai n</i>	<i>Presisi</i>	<i>Storage</i>
1 – 24	7 digit	4 bytes
25-53	15 digit	8 bytes

<i>Tipe Data</i>	<i>Range</i>	<i>Storage</i>
float	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Tergantung nilai n
real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	8 bytes

Date dan Time

<i>Tipe Data</i>	<i>Range</i>	<i>Storage</i>
date	01-01-01 sampai 9999-12-31 1-Januari-1 sampai 31-Desember-9999	3 bytes
time	00:00:00.0000000 sampai 23:59:59.9999999	5 bytes
datetime	Date range: 1-Januari-1753 sampai 31-Desember-9999 Time range: 00:00:00 sampai 23:59:59.997	8 bytes
datetime2	Date range: 1-Januari-1 sampai 31-Desember-9999 Time range: 00:00:00 sampai 23:59:59.9999999	6-8 bytes

datetimeoffset	Date range: 1-Januari-1 sampai 31-Desember-9999 Time range: 00:00:00 sampai 23:59:59.9999999	10 bytes
smalldatetime	Date range: 1-Januari-1 sampai 6-Januari-2079 Time range: 00:00:00 sampai 23:59:59	4 bytes

Character Strings

Tipe Data	Format	Storage
char	char [(n)], dimana n bisa bernilai 1 sampai 8000	n bytes
varchar	varchar [(n max)], dimana n bisa bernilai 1 sampai 8000	n bytes sampai $2^{31}-1$ bytes
nchar	nchar [(n)], dimana n bisa bernilai 1 sampai 4000	$2 \times n$ bytes
nvarchar	nvarchar [(n max)], dimana n bisa bernilai 1 sampai 4000	$2 \times n$ bytes Sampai 2GB

Tipe Data	Keterangan	Storage
ntext	Panjang max string $2^{30}-1$ (1,073,741,823) bytes	(2 x panjang string) bytes
text	Panjang max string $2^{31}-1$ (2,147,483,647) bytes	1 sampai 2,147,483,647 bytes
image	Data biner	0 sampai $2^{31}-1$ (2,147,483,647) bytes

Binary Strings

Tipe Data	Range	Storage
binary	binary [(n)], dimana n bisa bernilai 1 sampai 8000	n bytes
varbinary	varbinary [(n max)], dimana n bisa bernilai 0 sampai 8000	0 sampai $2^{31}-1$ bytes

Tentang Penulis

Herry Raditya Wibowo, adalah alumnus Teknik Informatika UKDW tahun 1995 dan merupakan pengamat serta praktisi di bidang teknologi informasi. Pernah bekerja di beberapa perusahaan baik lokal maupun multinasional. Saat ini mengelola Mitra Data Komputindo, sebuah perusahaan IT yang sedang berkembang yang melayani pengembangan software, konsultasi, dan sistem IT pada umumnya. Tinggal di Batam dan masih akan terus menulis buku-buku lainnya.

Jubilee Enterprise, telah dipercaya oleh penerbit dan pembaca buku tanah air dalam satu dasawarsa ini. Hingga sekarang, tulisan-tulisannya yang diterbitkan dalam bentuk buku telah mencapai hampir 400 judul, sebagian besar bertema teknologi informasi dan kemudian disusul dengan tema psikologi & parenting, manajemen, fotografi, anak-anak, dan tema umum lainnya. Salah satu imprint Jubilee Enterprise adalah Jubilee Authors Companion yang merupakan sahabat bagi para penulis-penulis profesional yang ingin tetap berkarya.

Anda dapat memperoleh informasi lebih lanjut tentang Jubilee Enterprise lewat situs: www.thinkjubilee.com.

Catatan:

- Untuk melakukan pemesanan buku, hubungi Layanan Langsung PT Elex Media Komputindo:

Gramedia Direct

Jl. Palmerah Barat No. 33, Jakarta 10270

Telemarketing/CS: 021-53650110/111 ext: 3901/3902

Email: endang@gramediapublishers.com

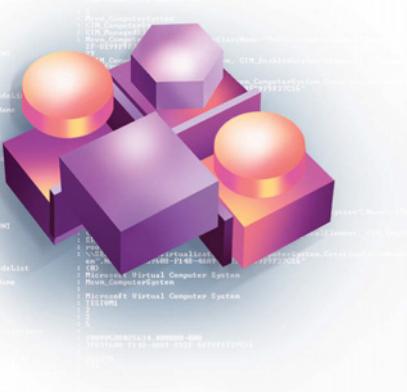
Visual Basic Database

Apabila Anda ingin mempelajari pemrograman database menggunakan Visual Basic, buku ini bisa menjadi pijakan pertama untuk belajar. Anda hanya membutuhkan software berikut untuk belajar dan mempraktikkan buku ini:

- Microsoft Visual Studio 2013 Express Edition (gratis)
- Microsoft Server 2014 Express (gratis)

Setelah itu, Anda akan belajar pemrograman database menggunakan Visual Basic. Agar pembahasannya aplikatif, Anda akan belajar langsung membuat aplikasi toko multifungsi yang memiliki fungsi login, pencatatan barang-barang baru, penjualan, pencatatan customer, pengaturan kategori, dan lain sebagainya.

Materi disajikan cukup komprehensif dan mudah dimengerti untuk kalangan pemula. Dapat dijadikan pegangan andal untuk membuat aplikasi serius yang melibatkan database. Jika Anda butuh script-script yang ada di dalam buku ini, Anda bisa mengunduhnya secara gratis pula.



PT ELEX MEDIA KOMPUTINDO
Kompas Gramedia Building
Jl. Palmerah Barat 29-37, Jakarta 10270
Telp. (021) 53650110-53650111, Ext 3214
Webpage: <http://www.elexmedia.co.id>

Kelompok
Pemrograman
Keterampilan
<input checked="" type="checkbox"/> Tingkat Pemula
<input checked="" type="checkbox"/> Tingkat Menengah
<input type="checkbox"/> Tingkat Mahir
Jenis Buku
<input checked="" type="checkbox"/> Referensi
<input checked="" type="checkbox"/> Tutorial
<input type="checkbox"/> Latihan

gramedia

ISBN 978-602-02-4584-3



9 786020 245843

121141677