

# Mengenal *Datatypes* SQL Server 2005

**Rangga Praduwiratna**

ziglaret@yahoo.co.nz

<http://geeks.netindonesia.net/blogs/ziglaret>

## ***Lisensi Dokumen:***

Copyright © 2003-2007 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

## **Pendahuluan**

Pada artikel sebelumnya mengenai penggunaan index pada tabel database, saya sudah menjelaskan sedikit mengenai konsep pages dan extent yang digunakan oleh SQL Server. Pada artikel kali ini, saya ingin berbagi pengetahuan saya mengenai tipe-tipe data (*datatypes*) yang terdapat pada SQL Server 2005. Perlu diketahui bahwa SQL Server 2005 memerlukan .NET Framework 2.0 dalam proses instalasinya. Jadi jangan heran jika Anda menemukan beberapa tipe data baru disini yang memudahkan pengguna dan developer untuk menentukan tipe data terbaik untuk kolom (untuk berikutnya akan saya sebut *field*) pada tabel database Anda.

Menentukan tipe data yang tepat, juga merupakan teknik implementasi dan administrasi database yang baik. Dengan menerapkan tipe data yang efektif dan efisien, maka Anda dapat menghemat space harddisk dan lama waktu pemrosesan pencarian data dari SQL Server Anda.

## Tipe Data *Built-in* pada SQL Server 2005

Apa kegunaan menentukan tipe data pada kolom/*field* tabel database Anda? Tipe data akan membatasi jenis/tipe data yang dapat dimasukkan oleh *user*/pengguna ke dalam tabel database. Anda perlu merencanakan dan menentukan tipe data apa yang tepat untuk sebuah *field* tertentu. Jika diimplementasikan dengan tepat, maka hal ini akan banyak membantu Anda dan menghemat resource komputer server Anda. Saya akan membahas sedikit mengenai cara menghitung berapa space yang dibutuhkan untuk membuat 1 row pada tabel dengan berbagai *field* dengan tipe data yang berbeda. Hal ini kemudian akan berguna untuk menentukan berapa besar space yang dibutuhkan oleh database Anda.

Berikut adalah beberapa tipe data *built-in* yang terdapat pada SQL Server 2005 beserta space yang dibutuhkan pada harddisk saat mengimplementasikannya :

Bit. Tipe data bit hanya bisa menerima input angka 1 dan 0 sebagai nilai (atau bisa juga null, yang berarti tidak ada nilai). Tipe data ini sangat membantu jika Anda ingin menghasilkan output yes/no, true/false, dsb.

Int. Tipe data ini mungkin sudah banyak dikenal oleh Anda. Tipe data ini dapat menerima nilai mulai dari  $-2^{31}$  (-2,147,483,648) hingga  $2^{31}-1$  (2,147,483,647). Tipe data ini menghabiskan 4 *bytes* untuk menyimpan data pada harddisk.

Bigint. Tipe data ini mirip dengan int, hanya saja nilai yang diterima lebih besar daripada int. Tipe data ini dapat menerima nilai mulai dari  $-2^{63}$  (-9,223,372,036,854,775,808) hingga  $2^{63}-1$  (-9,223,372,036,854,775,807). Tipe data ini menghabiskan 8 *bytes* untuk menyimpan data pada harddisk.

Smallint. Tipe data ini juga mirip dengan int, hanya saja nilai yang diterima lebih kecil dari int. Tipe data ini dapat menerima nilai mulai dari  $-2^{15}$  (-32,768) hingga  $2^{15}-1$  (32767). Tipe data ini hanya membutuhkan 2 *bytes* untuk menyimpan data pada harddisk.

Tinyint. Tipe data ini menerima nilai yang lebih kecil dari smallint. Nilai yang bisa diterima mulai dari 0 hingga 255, dan hanya membutuhkan 1 *bytes* untuk menyimpan data pada harddisk.

*Catatan kecil : mengapa saya mencantumkan space yang dibutuhkan untuk menyimpan data pada harddisk? Bayangkan jika Anda mempunyai 1 juta baris data dengan tipe data bigint pada tabel database Anda. Padahal nilai yang diterima dari user hanya berkisar 0 sampai 100 saja, berarti Anda membuang-buang  $(8*1.000.000)$  bytes -  $(1*1.000.000)$  bytes = 7.000.000 bytes atau sekitar 68 Kb] dari space harddisk Anda. Anda mungkin berpikir, ah ini kan kecil Cuma 68 Kb. Benar, tapi bayangkan jika ada beberapa field yang tidak efektif seperti field ini dan database server diakses oleh ratusan user pada waktu yang bersamaan, maka waktu pemrosesan untuk mencari data bakal menjadi lebih lama dari yang seharusnya.*

Decimal. Tipe data ini menerima nilai yang lebih presisi dibanding tipe data integer yang telah dibahas sebelumnya. Tipe data ini menggunakan 2 parameter untuk menentukan tingkat presisi nilai yang diterima; *precision* dan *scale*. *Precision* adalah jumlah digit yang bisa diterima oleh *field*, sedangkan *scale* adalah jumlah angka di belakang koma yang bisa diterima oleh *field*. Jadi, jika kita membuat parameter *precision* sebanyak 5 dan *scale* sebanyak 2 maka *field* kita bisa menerima nilai seperti ini : 123,45. Tipe data ini bisa menerima nilai mulai dari  $-10^{38}$  hingga  $10^{38}-1$ . Tipe data ini menghabiskan 5-17 *bytes* untuk menyimpan data pada harddisk, tergantung pada tingkat kepresisian nilai yang dimasukkan.

Numeric. Tipe data ini pada dasarnya sama dengan tipe data decimal. Jadi tipe data ini bisa disebut sinonim dari decimal.

Money. Tipe data ini dapat menerima nilai mulai dari  $-2^{63}$  (-9,223,372,036,854,775,808) hingga  $2^{63}-1$  (9,223,372,036,854,775,807). Tipe data ini menghabiskan 8 *bytes* untuk menyimpan data pada harddisk.

Smallmoney. Tipe data ini pada dasarnya sama dengan tipe data money, hanya saja nilai yang diterima lebih kecil, yaitu mulai dari -214,748.3648 hingga 214,748.3647. Tipe data ini menghabiskan 4 *bytes* untuk menyimpan data pada harddisk.

Float. Tipe data ini mirip dengan tipe data decimal, hanya saja parameter *scale* pada tipe data ini bisa menerima nilai yang tak terhingga, seperti pada nilai pi. Tipe data ini bisa menerima nilai mulai dari  $-1.79E + 308$  hingga  $1.79E +308$ . Jika Anda mendeskripsikan *field* dengan tipe data seperti ini : float(2), maka nilai output dari pi (misalnya) adalah 3,14. Angka 2 di dalam kurung menjelaskan berapa banyak angka yang harus ditampilkan dibelakang koma. Tipe data ini menghabiskan 4-8 *bytes* untuk menyimpan data pada harddisk.

Real. Tipe data ini mirip dengan tipe data float, hanya saja menerima nilai yang lebih kecil dibandingkan dengan float, yaitu mulai dari  $-3.40E +38$  hingga  $3.40E +38$ . Tipe data ini menghabiskan 4 *bytes* untuk menyimpan data pada harddisk.

Datetime. Tipe data ini dapat menerima nilai tanggal dan waktu mulai dari 1 Januari 1753 hingga 31 Desember 9999. Tipe data ini menghabiskan 8 *bytes* untuk menyimpan data pada harddisk.

Smalldatetime. Tipe data ini dapat menerima tanggal dan waktu mulai dari 1 Januari 1900 hingga 6 Juni 2079, dengan akurasi waktu yang digunakan adalah menit. Tipe data ini menghabiskan 4 *bytes* untuk menyimpan data pada harddisk.

Timestamp. Tipe data ini digunakan untuk mencatat record ketika data baru dimasukkan dan diupdate. Tipe data ini sangat berguna untuk mencari tahu perubahan yang terjadi pada database Anda.

Uniqueidentifier. Tipe data ini berfungsi untuk membuat nilai yang unik yang mungkin bisa tampil seperti ini 6F9619FF-8B86-D011-B42D-00C04FC964FF. Tipe data ini berguna jika Anda ingin membuat serial number atau id yang unik.

Char. Tipe data ini dapat digunakan untuk memasukkan data karakter non-Unicode dengan jumlah karakter yang fix. Tipe data ini bisa menerima hingga 8000 karakter, dan jumlah *bytes* yang dibutuhkan tergantung jumlah karakter yang dimasukkan. 1 karakter membutuhkan 1 *bytes*, sehingga jika Anda mendefinisikan seperti ini : *char*(5) maka *field* tersebut hanya bisa menerima karakter sebanyak 5 buah karakter dengan space yang dibutuhkan untuk menyimpan data pada harddisk sebanyak 5 *bytes*.

Varchar. Tipe data ini mirip dengan tipe data *char*, namun tipe data ini berguna bagi Anda yang tidak mengetahui secara pasti jumlah karakter yang akan dimasukkan oleh *user*. Tipe data ini juga bisa menerima nilai hingga 8000 karakter. Jadi, jika pada tipe data *char*, Anda mendefinisikan *char*(5), maka Anda akan selalu membutuhkan 5 *bytes* untuk menyimpan data pada harddisk, walaupun jumlah karakter yang dimasukkan hanya 1 hingga 4 karakter; maka pada tipe data ini, jumlah *bytes* yang dibutuhkan akan lebih fleksibel. Misalnya jika Anda mendefinisikan *varchar*(30) untuk sebuah *field*, maka *field* tersebut dapat menerima data hingga 30 karakter (30 *bytes*), namun jika Anda hanya memasukkan 1 karakter, maka jumlah *bytes* yang dibutuhkan hanya sebanyak 1 *bytes*.

Varchar(max). Tipe data ini juga mirip dengan varchar, hanya saja, nilai yang bisa diterima mencapai  $2^{31}-1$  (2,147,438,67) bytes data.

Nchar. Tipe data ini mirip dengan tipe data char, namun tipe data ini bisa menerima nilai atau data Unicode (berbeda dengan tipe data char yang hanya bisa menerima nilai karakter non-Unicode). Tipe data ini bisa menerima nilai hingga 4000 karakter. Tipe data ini menghabiskan 2-8000 bytes untuk menyimpan data pada harddisk. Mengapa dibutuhkan 2-8000 bytes? Karena tipe data ini mengkali 2 bytes untuk setiap karakternya. Jadi jika user hanya memasukkan 1 karakter, maka dibutuhkan 2 bytes untuk menyimpan data pada harddisk.

Nvarchar. Tipe data ini mirip dengan tipe data varchar, namun tipe data ini bisa menerima nilai atau data Unicode. Tipe data ini juga bisa menerima nilai hingga 4000 karakter.

Nvarchar(max). Tipe data ini mirip dengan tipe data varchar(max), namun tipe data ini bisa menerima nilai atau data Unicode. Tipe data ini bisa menerima karakter hingga  $2^{31}-1$  (2,147,483,67) bytes data.

Binary. Tipe data ini dapat menerima data binary dengan maksimum 8000 bytes data. Tipe data ini diinterpretasikan sebagai string dari bit misalnya (110011001011).

Varbinary. Tipe data ini mirip dengan varchar, hanya saja nilai yang bisa diterima hanya data binary. Tipe data ini berguna untuk menyimpan data binary yang tidak diketahui dengan pasti jumlah bytes datanya.

Xml. Tipe data ini berguna untuk menyimpan data dalam format XML Document. Tipe data ini dapat menyimpan data hingga 2Gb. Tipe data ini merupakan tipe data baru yang terdapat di SQL Server 2005.

SQL\_Variant. Tipe data ini merupakan tipe data baru di SQL Server 2005, saya pribadi belum mengetahui dengan pasti kapan saat yang tepat untuk mengimplementasikan tipe data ini. Tipe data ini disebutkan dapat digunakan untuk mengubah tipe data sesuai dengan apa yang dimasukkan oleh user. Mungkin, ilustrasinya seperti ini : jika user memasukkan angka ke dalam field dengan tipe data ini, maka SQL\_Variant akan menyesuaikan menjadi int atau tipe data lain yang lebih sesuai (seperti varchar), tapi jika kemudian user mengisi field tersebut dengan tipe data char, sql\_variant akan mengubahnya tipe data field untuk row tersebut menjadi char. Tipe data ini sebenarnya kurang disarankan untuk digunakan karena tidak adanya batasan yang jelas dalam penggunaannya, dan dapat menyebabkan collision data.

## **Tipe Data *User* (User Data Types/UDTs)**

Jika pada bab sebelumnya saya membahas mengenai tipe data *built-in* atau tipe data yang memang sudah disediakan oleh SQL Server, maka pada bab ini, saya akan sedikit membahas mengenai tipe data *user* (*user datatypes*).

Tipe data *user* (*user datatypes*, untuk selanjutnya akan saya singkat menjadi UDTs) dapat dibagi menjadi 2, yaitu T-SQL UDTs dan CLR UDTs. T-SQL UDTs digunakan untuk membuat konsistensi tipe data pada tabel, sedangkan CLR UDTs digunakan untuk membuat tipe data baru yang tidak terdapat pada SQL Server.

T-SQL UDTs. Tipe data ini pada dasarnya berfungsi sebagai aliasing pada tipe data *built-in* yang sudah terdapat di SQL Server untuk membuat konsistensi pada definisi tabel database Anda. Begini ilustrasinya. Seandainya Anda memiliki beberapa tabel database yang saling berkaitan, yaitu : Tabel Customer, Tabel Pegawai, dan Tabel Produk, semua tabel tersebut memiliki satu *field* yang sama yaitu *field* Kota. Untuk mendefinisikan tipe data *varchar*(30) untuk *field* ini, kita bisa melakukan 2 cara :

1. Kita bisa mendefinisikan tiap *field* untuk bertipe data *varchar*(30), atau
2. Kita membuat tipe data baru, misalnya tipe data *nama\_kota*, yang mengambil sifat dan parameter dari tipe data *varchar*(30). Seperti yang telah saya sebutkan sebelumnya, bahwa T-SQL UDTs bisa disebut sebagai aliasing, tujuannya untuk memudahkan *user* dan developer saat mendefinisikan tipe data untuk *field-field* pada tabel database. Tipe data baru *nama\_kota* ini tentu saja akan memiliki sifat yang sama dengan tipe data *varchar*(30).

CLR UDT's. Sebelum Anda dapat menggunakan tipe data ini, Anda harus menghidupkan fitur CLR dengan menggunakan Surface Area Configuration utility. Apa kegunaan tipe data ini? Seperti diketahui SQL Server tidak berorientasi objek, hal ini tentu berbeda dengan VB.NET misalnya, dimana kita bisa membuat banyak objek (*class*, dan *references* lainnya). Nah, Anda dapat menggunakan objek yang telah dibuat sebelumnya dengan menggunakan CLR UDTs ini.

Demikian penjelasan singkat mengenai beberapa tipe data yang terdapat di SQL Server 2005. Masih ada beberapa tipe data *built-in* lainnya sebetulnya, tapi yang saya bahas di bab sebelumnya merupakan tipe data yang sering digunakan. Untuk mengetahui lebih jelas mengenai tipe data T-SQL dan CLR, Anda dapat melihat pada SQL Server Book Online (BOL) pada bab 'T-SQL *User Defined Types*' dan 'CLR *User Defined Types*'.

## Penutup

Mengetahui tipe data yang tepat untuk diimplementasikan, merupakan hal yang berguna bagi para developer maupun database administrator. Hal ini dapat dioptimalkan untuk menghemat resource komputer yang digunakan untuk menyimpan data dan mengakses data.

## Referensi

- Jorden, Joseph. 2007. SQL Server 2005 DBA Street Smarts. Indiana. Wiley Publishing
- Solid Quality Learning. 2006. SQL Server 2005 Implementation and Maintenance. USA. Microsoft Press
- SQL Server 2005 Book Online (BOL)

## Biografi Penulis



**Rangga Praduwiratna.** Rangga Praduwiratna merupakan mahasiswa jurusan Teknologi Informasi Universitas Kristen Maranatha dan Teknik Arsitektur Institut Teknologi Bandung. Ia aktif menulis beberapa artikel mengenai teknologi Microsoft di INDC (Indonesia .NET Developer Community) dan merupakan MCTS (Microsoft Certified Technology Specialist) untuk teknologi SQL Server 2005 – Implementation and Maintenance.

Ia juga sempat menjadi asisten dosen Pemrograman Dasar C di Universitas Kristen Maranatha, dan beberapa kali menjadi pengajar AutoCAD untuk mahasiswa Arsitektur ITB yang diadakan oleh Ikatan Mahasiswa Arsitektur-Gunadharma ITB. Ia memiliki ketertarikan pada teknologi .NET, bahasa pemrograman C#, SQL Server 2005, serta beberapa software desain grafis.