Tutorial SQL
(Structured Query Language)

M. Choirul Amri
choirul@ilmukomputer.com
http://www.choirulamri.or.id

Structured Query Language (SQL) merupakan bahasa yang banyak digunakan dalam berbagai produk database. SQL dibangun di laboratorium IBM-San Jose California sekitar akhir tahun 70-an. Pertama kali dikembangkan sebagai bahasa di produk database DB2 yang sampai saat ini merupakan produk database andalan IBM. SQL sering di lafalkan dengan "sequel".

Saat ini organisasi standar America (ANSI) menetapkan standar bahasa SQL yaitu ANSI-92 standard. Masing-masing vendor database memiliki dialeknya sendiri sebagaian besar spesifikasinya mengacu pada standar ANSI tersebut dengan berbagai ekstensi tambahan. SQL Server menggunakan bahasa Transact-SQL dalam produknya, sedangkan Oracle menggunakan PL/SQL.

Dalam tutorial ini penulis menggunakan database NorthWind yang merupakan database sampel di SQL Server sebagai sarana latihan. Tools yang digunakan adalah Query Analyzer, yang dapat diakses dari menu Start > Program > Microsoft SQL Server > Query Analyzer. Anda juga dapat menggunakan produk database lain seperti MySQL atau Oracle dengan konsep yang sama.

Dasar SQL

Fungsi paling dasar dari SQL adalah untuk menampilkan data dari database. Data tersebut selanjutnya dapat difilter dan dimanipulasi sesuai kebutuhan aplikasi.

Perintah perintah dalam SQL terbagi dalam 2 kelompok besar :

- Data Manipulation Language
- Data Definition Language

Menampilkan Data dengan Statement SELECT

Syntax paling dasar untuk mengambil data dari database adalah sebagai berikut :

```
SELECT column
FROM table
```

Buka query Analyser dan pastikan anda telah terkoneksi dengan database Northwind. Tuliskan kode berikut untuk menampilkan data customer yang terdapat di tabel Customers.

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
```

Jalankan perintah tersebut dengan menekan tombol F5, maka akan tampil tiga kolom dari tabel Customers :

```
CustomerID CompanyName                              ContactName
---------- ---------------------------------------- ----------
ALFKI      Alfreds Futterkiste                       Maria
ANATR      Ana Trujillo Emparedados y helados        Ana
ANTON      Antonio Moreno Taquería                   Antonio
AROUT      Around the Horn                           Thomas
BERGS      Berglunds snabbköp                        Christina
BLAUS      Blauer See Delikatessen                   Hanna Moos
BLONP      Blondesddsl père et fils                  Frédérique
BOLID      Bólido Comidas preparadas                 Martín
BONAP      Bon app'                                  Laurence
……..
……..
```

Untuk menampilkan semua kolom dari suatu table, digunakan tanda asterik (*), daripada menyebutkan nama kolomnya satu per satu. Tiluskan statement berikut :

```
SELECT * FROM Customers
```

Maka akan ditampilkan seluruh kolom di table Customers yang berjumlah 11 kolom.

Meskipun cara ini sangat sederhana untuk menampilkan isi seluruh kolom dari suatu table, anda tetap dianjurkan untuk hanya mengambil data dari kolom yang anda perlukan saja. Pengambilan data yang tidak perlu mengakibatkan penurunan performa aplikasi.

Filter Data dengan WHERE

Perintah SELECT dan FROM diatas hanya membatasi jumlah kolom yang ditampilkan saja, sedangkan jumlah baris yang dihasilkan tidak dibatasi. Anda sering memerlukan hanya baris atau data yang memenuhi kriteria tertentu saja yang ditampilkan.

Klausa WHERE digunakan untuk menentukan kriteria RECORD yang ditampilkan. Syntax umumnya adalah sebagai berikut :

```
SELECT columns
FROM tables
WHERE Conditions
```

Perintah SQL diatas dapat dimodifikasi agar menampilkan data untuk customer dengan kode ALFKI saja. Kode SQL nya menjadi sebagai berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
WHERE CustomerID = 'ALFKI'
```

Setelah dirun maka tampil hasil query sebagai berikut :

```
CustomerID CompanyName                          ContactName
---------- ---------------------------------------- ----------
ALFKI      Alfreds Futterkiste                   Maria Anders

(1 row(s) affected)
```

Terlihat bahwa hanya dihasilkan satu record yang memenuhi kriteria. Dalam perintah tersebut kriteria yang digunakan adalah kolom CustomerID dengan nilai ALFKI.

Anda juga dapat membuat beberapa kriteria sekaligus dengan klausa WHERE. Logika yang digunakan bisa berupa OR (atau) serta AND (dan). Perhatikan contoh perintah berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
WHERE CustomerID = 'ALFKI' OR
       CustomerID = 'AROUT'
```

Maka dihasilkan dua buah record yang memenuhi kritreia tersebut :

```
CustomerID CompanyName                          ContactName
---------- ---------------------------------------- ----------
ALFKI      Alfreds Futterkiste                   Maria Anders
AROUT      Around the Horn                       Thomas Hardy
(2 row(s) affected)
```

Cara lain adalah menggunakan AND sehingga data yang ditampilkan hanya yang memenuhi kriteria yang disebutkan saja. Misalkan perintah berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
WHERE City  = 'London' AND
      ContactName = 'Thomas Hardy'
```

Maka record yang ditampilkan harus memenuhi kedua kriteria di klausa WHERE. Hasil yang didapat adalah :

```
CustomerID CompanyName                          ContactName
---------- ---------------------------------------- ----------AROUT      Around the Horn
Thomas Hardy
(1 row(s) affected)
```

Apabila anda mengganti AND dengan OR, maka hasilnya akan berbeda, yaitu sebagai berikut :

```
CustomerID CompanyName                          ContactName
---------- ---------------------------------------- ----------
AROUT      Around the Horn                      Thomas Hardy
BSBEV      B's Beverages                        Victoria Ashworth
CONSH      Consolidated Holdings                Elizabeth Brown
```

```
EASTC      Eastern Connection               Ann Devon
NORTS      North/South                      Simon Crowther
SEVES      Seven Seas Imports               Hari Kumar
(6 row(s) affected)
```

Sampai di sini anda telah mempelajari bagaimana mengambil data dari database berdasarkan kriteria tertentu, serta perbedaan penggabungan kriteria yang menggunakan OR dan AND.

Sortir Data dengan ORDER BY

ORDER BY digunakan untuk mengurutkan hasil pencarian data. Secara default data yang ditampilkan disortir berdasarkan urutan masuknya data ke dalam tabel. Dengan menggunakan ORDER BY anda dapat mengurutkan berdasarkan kolom tertentu yang anda kehendaki.

Bila anda perhatikan perintah SQL diatas maka data yang dihasilkan telah diurutkan berdasarkan kolom CustomerID. Anda dapat merubahnya dengan mengurutkan berdasarkan kolom ContactName dengan perintah berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
ORDER BY ContactName
```

Sehingga hasilnya adalah :

```
CustomerID CompanyName                      ContactName
---------- ---------------------------------------- ----------
ROMEY      Romero y tomillo                 Alejandra Camino
MORGK      Morgenstern Gesundkost           Alexander Feuer
ANATR      Ana Trujillo Emparedados         Ana Trujillo
TRADH      Tradição Hipermercados           Anabela Domingues
GOURL      Gourmet Lanchonetes              André Fonseca
EASTC      Eastern Connection               Ann Devon
LAMAI      La maison d'Asie                 Annette Roulet
```

Terlihat bahwa data telah diurutkan berdasarkan ContactName secara ascending (dari a ke z). Anda dapat membalik urutan menjadi dari z ke a dengan merubah klausa ORDER BY menjadi seperti berikut :

```
ORDER BY ContactName desc
```

Secara default urutan yang digunakan adalah ascending.

Selain itu dapat pula digunakan beberapa kriteria pengurutan. Artinya pengurutan dilakukan berdasarkan kolom yang disebut pertama, setelah itu kolom kedua, dan selanjutnya. Perintah diatas dapat ditambahkan sehingga menjadi sebagai berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
ORDER BY ContactName, CompanyName
```

Maka data akan diurutkan mengikuti ContactName, dan selanjutnya berdasarkan CompanyName.

Perintah ORDER BY juga dapat digabungkan dengan WHERE misalnya sebagai berikut :

```
SELECT CustomerID, CompanyName, ContactName
FROM Customers
WHERE City  = 'London' OR
      ContactName = 'Thomas Hardy'
ORDER BY ContactName
```

Perhatikan baik-baik bahwa ORDER BY harus diletakkan setelah WHERE. Apabila urutan tersebut terbalik maka statement SQL tidak dapat dijalankan dan menghasilkan pesan error berikut :

```
Server: Msg 156, Level 15, State 1, Line 4
Incorrect syntax near the keyword 'WHERE'.
```

Cari yang Mirip dengan LIKE

Apabila WHERE memfilter data berdasarkan kriteria tertentu yang sudah pasti, maka LIKE digunakan untuk memberikan kriteria yang tidak memiliki kepastian.

Misalkan anda ingin mencari nama produk yang dimulai dengan huruf c maka digunakan perintah berikut :

```
SELECT ProductID, ProductName
from Products
WHERE ProductName LIKE 'c%'
```

Perhatikan tanda % setelah huruf c tersebut, yang dapat diartikan sebagai : semua yang dimulai dengan c. Keluaran perintah tersebut adalah :

```
ProductID   ProductName
----------- ------------------------
60          Camembert Pierrot
18          Carnarvon Tigers
1           Chai
2           Chang
39          Chartreuse verte
4           Chef Anton's Cajun Seasoning
5           Chef Anton's Gumbo Mix
48          Chocolade
38          Côte de Blaye
(9 row(s) affected)
```

Contoh lain adalah bila diinginkan mencari nama produk yang mengandung huruf v maka digunakan perintah berikut :

```
select ProductID, ProductName
from dbo.Products
WHERE ProductName LIKE '%v%'
```

Perhatikan bahwa tanda % diletakkan sebelum dan sesudah huruf v, yang berarti dicari segala sesuatu yang mengandung huruf v tersebut.

Tanda % tersebut biasa dikenal sebagai wildcard, yang berfungsi menentukan berbagai kriteria dalam operator LIKE. Daftar selengkapnya mengenai wildcard yang dapat digunakan di SQL Server dapat Anda baca di Books Online SQL Server.

Melakukan Perhitungan

Selain mengambil data dari database anda dapat juga melakukan berbagai perhitungan terhadap data tersebut. Berbagai fungsi yang dapat dilakukan adalah penjumlahan, perkalian, pembagian dan pengurangan. Simbol-simbol yang digunakan adalah sebagai berikut :

- * Perkalian
- / Pembagian
- + Penjumlahan

- \- Pengurangan

Contoh berikut menghitung harga setiap produk yang terjual dengan mengalikan Quantity dan UnitPrice.

```
SELECT ProductID, (UnitPrice * Quantity) as TotalHarga
FROM [Order Details]
```

Misalkan tiap produk dikenakan diskon sebesar 10% dan ingin menampilkan harga setelah diskon, maka perintahnya adalah sebagai berikut :

```
SELECT ProductID, (UnitPrice * Quantity) as TotalHarga,
(UnitPrice * Quantity) * 0.1 as Diskon,
(UnitPrice * Quantity) * (1 - 0.1) as HargaDiskon
FROM [Order Details]
```

Diskon 10% adalah sama dengan 0.1, sehingga angka tersebut digunakan dalam kode program.

Hasil perhitungan selengkapnya adalah :

```
ProductID   TotalHarga     Diskon       HargaDiskon
----------- -------------------- ----------------------
11          168.0000      16.80000      151.20000
42          98.0000       9.80000       88.20000
72          174.0000      17.40000      156.60000
14          167.4000      16.74000      150.66000
51          1696.0000     169.60000     1526.40000
41          77.0000       7.70000       69.30000
51          1484.0000     148.40000     1335.60000
```

Membuat Alias dengan AS

Dalam contoh perhitungan dengan SQL diatas banyak digunakan keyword AS unatuk memberikan nama kolom. Fungsi AS tersebut adalah memberikan alias terhadap hasil perhitungan sehingga lebih mudah dibaca.

Apabila suatu perhitungan tidak disertakan alias menggunakan AS maka kolom hasil perhitungan tersebut menjadi tidak dikenal. Perhatikan contoh berikut :

```
SELECT ProductID, (UnitPrice * Quantity)
FROM [Order Details]
```

Maka kolom hasil perkalian tidak memiliki nama sebagaimana hasil berikut :

```
ProductID
----------- ----------
11          168.0000
42          98.0000
72          174.0000
14          167.4000
51          1696.0000
41          77.0000
```

Menghitung Group Data dengan Fungsi Agregat

Selain mengambil data dengan kriteria tertentu, sering juga diperlukan berbagai perhitungan yang bersifat ringkasan. Fungsi agregat merupakan sekumpulan fungsi yang siap dipakai untuk mendapatkan hasil penjumlahan, penghitungan frekuansi, rata-rata, dan lain-lain. Penggunaan fungsi ini sering digabungkan dengan klausa GROUP BY yang akan diterangkan kemudian.

Fungsi SUM

Apabila anda ingin mendapatkan jumlah dari sekelompok data yang memiliki kriteria tertentu maka SUM adalah pilihan yang tepat. Fungsi ini menjumlahkan nilai kolom tertentu yang telah dikelompokkan menurut kriteria tertentu.

Misalnya anda ingin menghitung jumlah jumlah barang yang terjual untuk kategori produk tertentu yang terdapat di tabel Order Details pada database NorthWind.

```
USE NorthWind
SELECT SUM(Quantity) as QTY from [Order Details]
WHERE ProductID = 11
```

Maka dihasilkan output sebagai berikut :

```
QTY
-----------
706
(1 row(s) affected)
```

Perintah tersebut diawali dengan USE Northwind yang artinya anda menggunakan database NorthWind. Misalnya anda ingin menggunakan database lain bernama Pubs maka diketikkan perintah USE Pubs. Perintah ini merupakan alternatif pemilihan database dengan cara manual dari menu combo di Query Analyser.

Selanjutnya diambil data kolom Quantity yang telah dimasukkan sebagai argumen fungsi SUM, sehingga output yang diharapkan adalah total Quantity produk yang memiliki ProductID = 11. Pengunaan kriteria ProductID merupakan suatu keharusan karena fungsi SUM akan menjumlahkan data yang memiliki kriteria tertentu. Kriteria tersebut dalam contoh diatas adalah ProductID = 11.

Fungsi SUM hanya dapat digunakan untuk menjumlahkan kolom dengan type data Numeric.

Fungsi COUNT

Untuk menghitung frekuensi pemunculan suatu data digunakan fungsi COUNT. Sebagaimana fungsi SUM, maka COUNT hanya dapat digunakan apabila data tersebut telah ditentukan kriterianya dengan klausa WHERE.

Misalkan anda ingin menghitung jumlah Customer yang terdapat di negara France. Tabel yang digunakan adalah tabel Customers.

```
SELECT COUNT (CustomerID) as Jumlah from Customers
WHERE Country = 'France'
```

Fungsi AVG

Fungsi ini hanya dapat digunakan untuk tipe data numeric, sebagaimana fungsi SUM. AVG menghitung rata-rata sekumpulan data yang telah ditentukan kriterianya menggunakan WHERE.

```
USE NorthWind
SELECT AVG(Quantity) as Rataan from [Order Details]
WHERE ProductID = 11
```

Contoh diatas menghitung rata-rata jumlah produk yang terjual untuk barang dengan ProductID = 11. Apabila diinginkan menghitung rata-rata seluruh jumlah peroduk terjual maka perintahnya menjadi :

```
SELECT AVG(Quantity) as Rataan from [Order Details]
```

Fungsi MIN dan MAX

Untuk mencari nilai maksimum dan minimum dari sekumpulan data anda dapat menggunakan fungsi MIN dan MAX. Misalkan anda ingin mencari jumlah barang yang paling banyak terjual maka dijalankan perintah berikut :

```
SELECT MAX(Quantity) as Maksimum from [Order Details]
```

Menggunakan Beberapa Fungsi Sekaligus

Fungsi-fungsi yang telah dijelaskan diatas juga dapat digunakan bersama-sama dalam satu perintah. Perhatikan baris kode berikut :

```
USE NorthWind
SELECT MAX(Quantity) as Maksimum,
     MIN(Quantity) as Minimum,
     AVG(Quantity) as Rataan
from [Order Details]
```

Jalankan perintah tersebut maka didapat hasil sebagai berikut :

```
Maksimum Minimum Rataan
-------- ------- -----------
130      1       23
(1 row(s) affected)
```

Klausa GROUP BY

Fungsi agregat yang telah dijelaskan sebelumnya hanya menampilkan satu baris hasil. Sering dibutuhkan untuk menampilkan rangkuman hasil perhitungan beberapa kelompok data dalam satu kali tampilan.

Misalnya anda ingin menghitung rata-rata dan jumlah produk yang terjual untuk setiap jenis produk. Untuk melakukan ini digunakan klausa GROUP BY yang berfungsi mengelompokkan data yang memiliki kriteria sama. Dengan demikian dapat dihasilkan suatu rangkuman hasil perhitungan untuk tiap kategori data. Tuliskan kode program berikut :

```
SELECT ProductID, SUM(Quantity)as Jumlah
from [Order Details]
GROUP BY ProductID
```

Perintah tersebut akan menghitung jumlah produk yang terjual untuk setiap ProductID dan mengelompokkan hasilnya berdasarkan ProductID tersebut. Hasil yang didapat sebagai berikut :

```
ProductID   Jumlah
----------- -----------
23          580
46          548
69          714
77          791
31          1397
15          122
62          1083
```

Terlihat jelas bahwa SQL Server telah mengelompokkan barang berdasarkan ProductID dan menghitung jumlah di kolom (QTY) untuk tiap ProductID tersebut.

Hal yang perlu mendapatkan perhatian adalah pada baris terakhir kode program tersebut yaitu pada bagian :

```
GROUP BY ProductID
```

Bagian ini berfungsi mengelompokkan barang berdasarkan ProductID nya. Sedangkan kolom ProductID sendiri telah disebutkan di dalam daftar SELECT. Ini merupakan aturan dasar apabila anda menggunakan GROUP BY dalam perhitungan dengan fungsi agregat. Kolom dalam daftar SELECT yang tidak dihitung dengan fungsi agregat harus dimasukkan dalam daftar GROUP BY. Pada contoh diatas kolom ProductID masuk dalam daftar SELCT tetapi tidak dihitung menggunakan fungsi SUM, sehingga harus dimasukkan dalam daftar GROUP BY.

Salah satu contoh pengembangan yang lebih kompleks dari perintah penggunaan GROUP BY ini adalah sebagai berikut :

```
SELECT ProductID, SUM(Quantity)as Jumlah,
     AVG(Quantity) as Rataan,
     SUM(Quantity*UnitPrice)as Nilai,
     AVG(Quantity*UnitPrice)as AVGNilai
from [Order Details]
GROUP BY ProductID
ORDER BY Nilai DESC
```

Kode tersebut menghitung jumlah, rata-rata, serta nilai uang tiap kategori produk. Nilai uang dari tiap produk didapat dengan mengalikan Quantity dengan UnitPrice. Selanjutnya data diurutkan berdasarkan nilai uangnya dari yang terbesar hingga terkecil. Hasilnya adalah sebagai berikut :

```
ProductID   Jumlah   Rataan    Nilai         AVGNilai
----------- ----------- ----------- -------------------
38          623         25    149984.2000     6249.3416
29          746         23    87736.4000      2741.7625
59          1496        27    76296.0000      1412.8888
60          1577        30    50286.0000      986.0000
62          1083        22    49827.9000      1038.0812
56          1263        25    45121.2000      902.4240
51          886         22    44742.6000      1147.2461
17          978         26    35482.2000      958.9783
18          539         19    31987.5000      1184.7222
28          640         19    26865.6000      814.1090
```

Perhatikan baik-baik bahwa perintah ORDER BY harus selalu diletakkan setelah GROUP BY.

Menghilangkan Penggandaan dengan DISTINCT

Penggunaan keyword DISTINCT hampir mirip dengan GROUP BY yaitu untuk melakukan grouping hasil pencarian. Tetapi DISTINCT hanya berfungsi menghilangkan duplikasi hasil dan tidak dapat digunakan untuk membantu melakukan berbagai perhitungan fungsi agregat.

Penerapan praktisnya misalnya anda ingin mencari negara mana saja yang yang terdapat di tabel customer. Apabila anda menggunakan perintah SELECT saja maka akan muncul banyak duplikasi karena banyak terdapat customer yang berasal dari negara yang sama. Dengan menggunakan DISTINCT maka duplikasi tersebut dapat dihilangkan.

```
SELECT DISTINCT Country
FROM Customers
```

Cobalah bandingkan hasilnya apabila anda menggunakan perintah berikut :

```
SELECT Country
FROM Customers
```

Membatasi GROUP BY dengan HAVING

Untuk membatasi hasil yang didapat dari suatu perintah yang menggunakan GROUP BY digunakan klausa HAVING. Fungsi klausa ini mirip dengan WHERE tetapi HAVING hanya dapat digunakan bersamaan dengan klausa GROUP BY dan diletakkan sesudah GROUP BY.

Perintah berikut digunakan untuk mencari produk yang jumlah penjualannya melebihi 1000 buah di tabel Order Details.

```
SELECT ProductID, SUM(Quantity)as Jumlah,
FROM [Order Details]
GROUP BY ProductID
HAVING SUM(Quantity) > 1000
```

Apabila dijalankan maka hasilnya adalah sebagai berikut :

```
ProductID   Jumlah
----------- --------
31          1397
62          1083
21          1016
75          1155
60          1577
```

Terlihat jelas bahwa yang ditampilkan hanya yang memiliki jumlah diatas 1000 sebagaimana kriteria :

```
HAVING SUM(Quantity) > 1000
15
```

Mencari Data dengan BETWEEN

Keyword BETWEEN digunakan untuk mencari data yang memenuhi interval kriteria tertentu dalam suatu kolom. Biasanya digunakan untuk mencari data berdasarkan interval tanggal tertentu.

Penggunaan praktisnya misalnya anda ingin mencari penjualan yang terjadi antara tanggal 1 Januari 2003 sampai dengan 31 Januari 2003. Keyword BETWEEN digunakan bersamaan dengan interval tanggal tersebut. Tuliskan baris kode berikut :

```
USE NorthWind
SELECT OrderID, CustomerID, OrderDate
FROM Orders
WHERE OrderDate BETWEEN '07/01/1996' AND '07/31/1996'
```

Kode tersebut mencari data di tabel Orders yang memiliki OrderDate antara tanggal 1 Juli 1996 sampai dengan 31 Juli 1996. Setelah di run maka hasilnya adalah :

```
OrderID     CustomerID OrderDate
----------- ---------- -----------------
10248       VINET      1996-07-04 00:00:00.000
10249       TOMSP      1996-07-05 00:00:00.000
10250       HANAR      1996-07-08 00:00:00.000
10251       VICTE      1996-07-08 00:00:00.000
10252       SUPRD      1996-07-09 00:00:00.000
10253       HANAR      1996-07-10 00:00:00.000
```

Selain menggunakan kriteria tanggal anda juga dapat menggunakan keyword BETWEEN tersebut untuk tipe data lain misalnya string dan numeric. Misalnya anda ingin menampilkan data customer yang diawali huruf A sampai dengan D, maka digunakan kode program berikut :

```
SELECT * FROM dbo.Customers
WHERE CompanyName BETWEEN 'A' AND 'D'
```

Menggabungkan Tabel

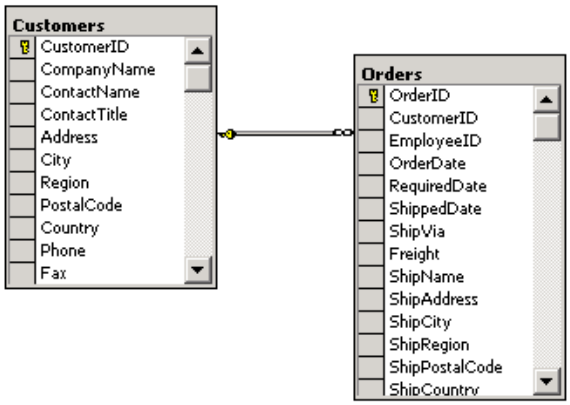
Anda telah mempelajari bagaimana menampilkan data dari database Northwind serta memfilternya dengan berbagai kriteria. Pengambilan data tersebut hanya dari satu tabel saja sehingga belum dapat menyajikan informasi lebih detil.

Perintah SQL dapat digunakan untuk menampilkan data dari 2 atau lebih tabel. Antara tabel tersebut harus memiliki penghubung yaitu Primary Key dan Foreign Key sebagaimana telah dijelaskan pada Bab 2 mengenai desain database. Dengan menggunakan kriteria penghubung tersebut anda dapat menampilkan data dari beberapa tabel secara konsisten.

Inner Join

Penggabungan tabel dalam perintah SQL menggunakan keyword JOIN. Jenis penggabungan tabel yang pertama adalah INNER JOIN. INNER JOIN hanya menampilkan data yang benar-benar terdapat di dalam tabel yang saling dihubungkan.

Misalnya anda menggabungkan tabel Customers dengan Orders, maka field penghubung yang digunakan adalah CustomerID. Perhatikan diagram berikut :



Pada gambar tersebut terlihat bahwa CustomerID merupakan Primary Key di tabel Customers dan menjadi Foreign Key di tabel Orders. Hubungan kedua tabel tersebut adalah One to Many, dimana satu customer dapat memiliki banyak order yang berulang.

Dalam teknik INNER JOIN maka hasil yang ditampilkan hanya record yang memiliki CustomerID sama di kedua tabel tersebut. Apabila terdapat customr yang CustomerID nya tidak ditemukan di tabel Orders maka data tersebut tidak ditampilkan.

Coba ketikkan perintah berikut di Query Analyser :

```
SELECT CompanyName, OrderID, OrderDate
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

Dan perhatikan hasilnya sebagai berikut :

```
CompanyName                OrderID     OrderDate
---------------------------------------- ----------- ---------
Vins et alcools Chevalier  10248       1996-07-04 00:00:00.000
Toms Spezialitäten         10249       1996-07-05 00:00:00.000
Hanari Carnes              10250       1996-07-08 00:00:00.000
Victuailles en stock       10251       1996-07-08 00:00:00.000
Suprêmes délices           10252       1996-07-09 00:00:00.000
Hanari Carnes              10253       1996-07-10 00:00:00.000
……
…….
(830 row(s) affected)
```

Perintah SQL tersebut mengambil field CompanyName dari tabel Customers, sedangkan field OrderID dan OrderDate diambil dari tabel Orders. Perhatikan baik baik diagram pada gambar 3.2.

Kunci utama penggabungan dua tabel tersebut adalah keyword INNER JOIN dengan kriteria CustomerID. Perhatikan bahwa nama tabel juga harus dituliskan sebelum nama field CustomerID tersebut.

```
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

Antara nama tabel dengan nama field dibatasi dengan tanda titik (.) misalnya Orders.CustomerID. Tujuan pencantuman nama tabel tersebut adalah menghindari ambiguitas yang mengakibatkan error apabila perintah tersebut dieksekusi. Karena CustomerID tersebut terdapat di kedua tabel maka nama tabel harus dicantumkan agar dapat diidentifikasi secara unik field mana yang dimaksud.

Untuk memberikan gambaran lebih kompleks mengenai penerapan penggabungan tabel ini anda dapat memodifikasi contoh perintah yang terdapat penggunaan GROUP BY. Pada perintah tersebut anda menghitung Jumlah dan rataan produk yang terjual, tetapi hanya ditampilkan ProductID saja sedangkan nama produknya tidak terlihat karena terdapat di tabel lain.

Anda dapat menggabungan tabel Order Details tersebut dengan tabel Products yang menyimpan nama produk. Dengan demikian dapat ditampilkan baik ProductID maupun nama produknya. Tuliskan baris perintah berikut :

```
SELECT Products.ProductID, Products.ProductName,
     SUM([Order Details].Quantity) AS Jumlah,
     AVG([Order Details].Quantity) AS Rataan,
     SUM(Products.UnitPrice * [Order Details].Quantity)
     AS Nilai,
     AVG(Products.UnitPrice * [Order Details].Quantity)
     AS AVGNilai
FROM Products
INNER JOIN [Order Details]
ON Products.ProductID = [Order Details].ProductID
GROUP BY Products.ProductID, Products.ProductName
```

Setelah dijalankan maka akan tampak hasilnya :

```
ProductID ProductName Jumlah   Rataan  Nilai    AVGNilai
---------- ---------------------------------------- ---------
1          Chai        828      21    14904.0000 392.2105
2          Chang       1057     24    20083.0000 456.4318
3          Aniseed     328      27    3280.0000  273.3333
4          Chef Anton  453      22    9966.0000  498.3000
```

Pada perintah diatas anda dapat melihat bahwa semua nama field diawali dengan nama tabelnya. Hal ini menyebabkan penulisan kode program menjadi panjang, tetapi sisi positifnya adalah kita dapat langsung mengetahui posisi tabel dari tiap-tiap field yang ditampilkan.

Selain itu antara tabel Customers terdapat beberapa field yang namanya sama sehingga apabila tidak dituliskan nama tabelnya berakibat error pada perintah tersebut.

LEFT JOIN dan RIGHT JOIN

Berbeda dengan INNER JOIN yang hanya menampilkan irisan data dari tabel yang digabungkan maka LEFT JOIN akan menampilkan data dari tabel yang disebutkan terlebih dahulu.

LEFT JOIN

Misalkan pada diagram gambar 3.2 diatas digunakan perintah LEFT JOIN maka apabila tabel Customers disebutkan terlebih dahulu SQL Server akan menampilkan semua nama customer yang terdapat di tabel tersebut. Semua data customer ditampilkan walaupun CustomerID nya tidak terdapat di tabel Orders.

Penggunaan praktis perintah ini misalnya anda ingin mengetahui frekuensi order semua customer baik yang telah memiliki order maupun yang belum. Untuk kebutuhan tersebut maka ditampilkan semua customer berikut data frekuensi ordernya. Perhatikan contoh perintah berikut :

```
SELECT Customers.CustomerID, Customers.CompanyName,
     COUNT(Orders.OrderID) AS Frekuensi
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.CompanyName
ORDER BY COUNT(Orders.OrderID)
```

Karena menggunakan perintah LEFT JOIN maka semua data di tabel Customers ditampilkan seluruhnya walaupun frekuensi ordernya 0. Hasil perintah tersebut adalah :

```
CustomerID CompanyName                               Frekuensi
---------- ----------------------------------------- ----------
PARIS      Paris spécialités                         0
FISSA      FISSA Fabrica Inter. Salchichas S.A.      0
CENTC      Centro comercial Moctezuma                1
LAZYK      Lazy K Kountry Store                      2
GROSR      GROSELLA-Restaurante                      2
LAUGB      Laughing Bacchus Wine Cellars             3
……
……
(91 row(s) affected)
```

Anda dapat melihat terdapat 2 customer yang frekuensi 0 tetapi tetap ditampilkan. Di akhir tampilan hasil tersebut terlihat ada 91 baris yang ditampikan. Untuk melakukan pengecekan jalankan perintah berikut :

```
SELECT  COUNT(*) from dbo.Customers
```

maka hasilnya adalah :

```
-----------
91
(1 row(s) affected)
```

Artinya terdapat 91customer di tabel Customers, yaitu sama dengan jumlah baris di perintah sebelumnya. Dengan demikian memang benar bahwa LEFT JOIN telah menampilkan seluruh Customer yang berjumlah 91.

RIGHT JOIN

Perintah RIGHT JOIN merupakan kebalikan dari LEFT JOIN, yaitu menampilkan semua isi tabel yang disebutkan kedua dalam perintah join. Dalam contoh di atas apabila LEFT JOIN diganti RIGHT JOIN maka semua isi tabel Orders akan ditampilkan semuanya.

Cobalah memodifikasi perintah diatas menjadi sebagai berikut :

```
SELECT Customers.CustomerID, Customers.CompanyName,
```

```
    COUNT(Orders.OrderID) AS Frekuensi
FROM Customers
RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.CompanyName
ORDER BY COUNT(Orders.OrderID)
```

Setelah anda jalankan perintah tersebut lihatlah perbedaaanya dengan perintah LEFT JOIN sebelumnya. Maka terlihat hanya dihasilkan 89 baris, berarti terdapat 2 customer yang CustomerID nya tidak terdapat di tabel Orders atau frekuensi ordernya = 0.

FULL JOIN

Jenis join terakhir adalah FULL JOIN yang menampilkan semua data dari dua tabel yang dihubungkan meskipun terdapat data yang tidak memiliki pasangan di tabel lainnya.

Misalnya kita mengambil data dari tabel Country dengan tabel City menggunakan FULL JOIN. Data dari kedua tabel akan ditampilkan semuanya baik untuk nama kota yang tidak memiliki data negara maupun sebaliknya. Perhatikan perintah berikut :

Manipulasi Data dengan SQL

Selain untuk mengambil informasi dari database anda juga dapat menggunakan perintah SQL untuk memanipulasi data. Proses tersebut meliputi menambah, menghapus, dan mengedit data.

Perintah manipulasi data sangat sering digunakan dalam aplikasi database dan bahakan dapat dikatakan menjadi inti sebuah aplikasi. Sebuah tabel dapat diisi dengan data, dihapus, maupun diedit datanya. Perintah-perintah tersebut dilaksanakan berdasarkan kriteria tertentu menggunakan keyword WHERE, BETWEEN maupun LIKE.

Statement INSERT

Untuk mengisikan data ke dalam suatu tabel digunakan perntah INSERT yang memiliki syntax umum sebagai berkut :

```
INSERT table (column list)
VALUES (value list)
```

Misalnya untuk mengisikan data customer baru dituliskan perintah berikut :

```
INSERT Customers (CustomerID, CompanyName, ContactName)
VALUES ('MJTR', 'Majuterus', 'Lisha')
```

Perintah tersebut mengisikan data di tabel Customers untuk tiga kolom yaitu CustomerID, CompanyName dan ContactName. Sedangkan kolom lain yang tidak diisi maka terisi dengan nilai default sesuai dengan desain tabelnya. Apabila desain tabelnya tidak mengijinkan nilai NULL maka anda harus mengisikan nilainya dalam perintah INSERT tersebut.

Apabila anda menampilkan data tabel Customers maka tampak data yang telah diisikan tersebut sebagai berikut :

```
CustomerID  CompanyName                    ContactName
----------------------------------------- --------------------
MAISD       Maison Dewey           Catherine Dewey                MEREP         Mère
Paillarde           Jean Fresnière
MJTR        Majuterus               Lisha
MORGK       Morgenstern Gesundkost Alexander Feuer
```

Apabila perintah INSERT digunakan untuk mengisikan seluruh kolom yang terdapat di suatu tabel maka nama kolom tidak perlu disebutkan secara eksplisit. Cukup disebutkan nilai data yang akan dimasukkan saja. Misalnya untuk mengisikan data ke tabel Shippers yang hanya terdiri dari tiga kolom dilancarkan perintah berikut :

```
INSERT Shippers
```

```
VALUES ('Megah Shipping', '021-55568953')
```

Anda mungkin bertanya mengapa VALUES yang diisikan hanya dua kolom, sedangan tabel Shippers terdiri dari 3 kolom. Jawabannya adalah karena kolom pertama yang bernama ShippersID telah disetting desainnya sebagai autonumber. Dengan demikian kolom tersebut akan terisi secara otomatis dengan angka berurut setiap terdapat data baru yang dimasukkan sehingga tidak perlu lagi diinsert secara eksplisit.

INSERT dari Tabel Lain

Pengisian tabel juga dapat menggunakan data yang diperoleh dari tabel lain. Caranya adalah dengan menggunakan perintah SELECT v berisi daftar data yang akan dimasukkan setelah perintah INSERT.

Misalnya anda ingin memasukkan data di tabel Suppliers ke dalam tabel Customers maka digunakan perintah berikut :

```
INSERT Customers (CustomerID, CompanyName, ContactName)
SELECT SupplierID, CompanyName, ContactName
FROM Suppliers
WHERE Country = 'USA'
```

Apabila anda belum melakukan perubahan apapun di tabel Suppliers maka akan terdapat 4 supplier yang diisikan ke tabel Customers.

Perintah tersebut menggunakan kriteria WHERE Country = 'USA', sehingga hanya supplier yang berada di negara USA saja yang dimasukkan ke tabel Customers.

Prinsip utama dalam penggunaan perintah tersebut adalah jumlah kolom yang akan diisi harus sama dengan kolom yang diambil dalam daftar SELECT. Dengan demikian apabila dua buah tabel memiliki jumlah kolom yang sama maka anda dapat menggunakan perintah tersebut untuk mengisikan seluruh kolom dengan perintah sederhana tanpa menyebutkan kolmnya satu per satu.

Misalkan tabel Customers dan Suppliers memiliki jumlah kolom sama dan anda ingin memasukkan semua supplier ke tabel Customers maka dapat digunakan perintah sederhana sebagai berikut :

```
INSERT Customers
SELECT * FROM Suppliers
```

Statement DELETE

Statement DELETE merupakan kebalikan perintah INSERT. Perintah ini menghapus data yang terdapat di suatu tabel. Data dihapus per record atau per baris berdasarkan kriteria tertentu.

Penentuan kriteria record mana yang akan dihapus bisa dilakukan dengan menggunakan klausa WHERE. Misalkan anda ingin menghapus data semua customer yang berada di negara France.

Syntax umum statement ini adalah sebagai berikut :

```
DELETE FROM table_name
WHERE Condition
```

Untuk menghapus data customer yang berasal dari Mexico di tabel Customers maka perintahnya adalah :

```
DELETE FROM Customers
where Country= 'Mexico'
```

Atau anda bisa juga menghapus data berdasarkan CustomerID yang merupakan primary key di tabel tersebut.

```
DELETE FROM Customers
where CustomerID= ‘ALFKI’
```

Selain menggunakan WHERE, dapat juga digabungkan dengan operator LIKE dan BETWEEN untuk membuat kriteria yang lebih fleksible. Misalnya sebagai berikut :

```
DELETE FROM Orders
```

```
WHERE OrderDate BETWEEN '07/01/1996' AND '07/31/1996'
```

Perintah tersebut akan menghapus data di tabel order yang memiliki OrderDate antara 1 Juli 1996 sampai dengan 31 Juli 1996.

Menghapus Seluruh Tabel

Apabila anda ingin mengosongkan tabel dan menghapus semua data yang ada di dalamnya maka digunakan perintah DELETE tanpa menggunakan kondisi WHERE.

Contoh berikut adalah perintah untuk mengosongkan isi tabel Products :

```
DELETE Products
```

Perintah tersebut hanya mengosongkan isi tabel saja tetapi tidak menghapus tabelnya.

Perlu diperhatikan apabila data dalam suatu tabel ternyata memiliki hubungan referential integrity dengan tabel lain maka penghapusan tersebut tidak dapat dilakukan. Mislanya apabila anda ingin menghapus data customer yang telah memiliki data order di tabel Orders, sedangakan antara kedua tabel tersebut memiliki hubungan referential integrity.

Untuk menghapus semua data di suatu tabel yang tidak memiliki hubungan referential integrity dengan tabel lain dapat digunakan perintah TRINCATE.

```
TRUNCATE TABLE Customers
```

Maka perintah tersebut akan menghapus semua data customer yang tidak memiliki hubungan dengan tabel lain.

Statement UPDATE

Apabila anda ingin mengedit atau merubah suatu data tanpa menghapusnya maka digunakan perintah UPDATE . Perintah ini juga menggunakan kondisi tertentu dengan klausa WHERE sebagaimana perintah DELETE.

Syntax umum statement UPDATE adalah sebagai berikut :

```
UPDATE table_name
SET Column1 = Value1, Column2 = Value2, …..
WHERE condition
```

Perintah ini melakukan perubahan pada kolom tertentu sebagaimana yang disebutkan dalam perintah SET. Perubahan dilakukan terhadap record yang memenuhi kriteria di klausa WHERE.

Misalkan anda ingin menaikkan harga produk sebesar 10% untuk semua barang yang memiliki CategoryID = 2, maka digunakan perintah berikut :

```
UPDATE Products
SET UnitPrice = UnitPrice * 1.1
WHERE CategoryID = 2
```

Harga barang terdapat di kolom UnitPrice sehingga kolom tersebut dikalikan dengan 1.1. Perintah tersebut akan menaikkan harga semua barang yang terdapat di tabel Products sesuai dengan kriteria CategoryID.

Selain menggunakan kriteria WHERE perintah ini juga dapat digabungkan dengan operator LIKE dan BETWEEN sebagaiman statement DELETE diatas.

Contoh berikut menggambarkan penggunaan beberapa kriteria untuk mengupdate suatu data :

```
UPDATE Customers
SET  Country = 'Indonesia',
     ContactName = 'Lisha'
```

```
WHERE City = 'Jakarta'
     OR City = 'Surabaya'
```

Perintah diatas berguna untuk merubah nama negara menjadi Indonesia dan nama ContactName menjadi Lisha apabila kolom City berisi Jakarta atau Surabaya.

----------0000000----------

*Untuk diskusi lebih jauh tentang SQL Server, silakan bergabung dengan milis sqlserver-indo@yahoogroups.com. Kirim email kosong ke sqlserver-indo-subscribe@yahoogroups.com. Atau daftar melalui web di: http://groups.yahoo.com/group/sqlserver-indo.