PERINTAH JOIN

Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggunakan bentuk perintah **JOIN**.

Contoh Class Diagram Sistem Pembelian

1. Inner Join

Dengan inner join, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi. Sebagai contoh, kita akan menggabungkan tabel pelanggan dan pesan dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi). Misalkan isi tabel pelanggan dan pesan adalah sebagai berikut :

Tabel **pelanggan**

```
+--------------+-----------------+--------------------+
| id_pelanggan | nm_pelanggan    | email              |
+--------------+-----------------+--------------------+
| P0001        | Achmad Solichin | achmatim@gmail.com |
| P0002        | Budianto        | budi@luhur.com     |
| P0003        | Hasan           | hasan02@yahoo.com  |
| P0004        | Amin Riyadi     | aminudin@plasa.com |
+--------------+-----------------+--------------------+
```

Tabel **pesan**.

```
+----------+--------------+------------+
| id_pesan | id_pelanggan | tgl_pesan  |
+----------+--------------+------------+
|        1 | P0001        | 2008-02-02 |
|        2 | P0002        | 2008-02-05 |
|        3 | P0002        | 2008-02-10 |
|        4 | P0004        | 2008-01-20 |
|        5 | P0001        | 2007-12-14 |
+----------+--------------+------------+
```

1. Join dengan WHERE.

Penggabungan dengan klausa WHERE memiliki bentuk umum sebagai berikut:

```
SELECT tabel1.*, tabel2.* FROM tabel1, tabel2
WHERE tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan,
pelanggan.nm_pelanggan, pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan, pesan
WHERE pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya sebagai berikut:

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
+--------------+-----------------+----------+------------+
```

Hasil Penggabungan 2 Tabel dengan WHERE

Pada hasil perintah query di atas terlihat bahwa terdapat 5 (lima) transaksi yang dilakukan oleh 3 (tiga) orang pelanggan. Jika kita lihat kembali isi tabel pelanggan di atas, maka terdapat satu pelanggan yang tidak ditampilkan yaitu yang memiliki **id pelanggan P0003**. Pelanggan tersebut tidak ditampilkan karena belum pernah melakukan transaksi.

2. Join dengan klausa INNER JOIN.

Berikut ini bentuk umumnya:

```
SELECT tabel1.*, tabel2.*
FROM tabel1 INNER JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Dan berikut ini perintah SQL penggabungan tabel pelanggan dan pesan.

```
SELECT pelanggan.id_pelanggan,
pelanggan.nm_pelanggan, pesan.id_pesan,
pesan.tgl_pesan
FROM pelanggan INNER JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

3. Outer Join

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang **NULL** (kosong) di satu sisi. Sebagai contoh, kita akan menggabungkan tabel pelanggan dan pesan dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi).

Outer Join terbagi menjadi 2 (dua) yaitu **LEFT JOIN** dan **RIGHT JOIN**. Berikut ini bentuk umum dan contohnya:

a. LEFT JOIN.

Bentuk umum:

```
SELECT tabel1.*, tabel2.*
FROM tabel1 LEFT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Contoh perintah SQL:

```
SELECT pelanggan.id_pelanggan,
pelanggan.nm_pelanggan, pesan.id_pesan,
pesan.tgl_pesan
FROM pelanggan LEFT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya:

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0003        | Hasan           |     NULL | NULL       |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
+--------------+-----------------+----------+------------+
```

Hasil Perintah Left Join

Berbeda dengan hasil sebelumnya (inner join), penggunaan left join akan menampilkan juga data pelanggan dengan id P0003, walaupun pelanggan tersebut belum pernah bertransaksi. Dan pada kolom id_pesan dan tgl_pesan untuk pelanggan P0003 isinya NULL, artinya di tabel kanan (pesan) pelanggan tersebut tidak ada.

b. RIGHT JOIN

Bentuk umum:

```
SELECT tabel1.*, tabel2.*
FROM tabel1 RIGHT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Contoh perintah SQL:

```
SELECT pelanggan.id_pelanggan,
pelanggan.nm_pelanggan, pesan.id_pesan,
pesan.tgl_pesan
FROM pelanggan RIGHT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya:

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
+--------------+-----------------+----------+------------+
```

Hasil Perintah Right Join

Dengan right join, tabel yang menjadi acuan adalah tabel sebelah kanan (tabel pesan), jadi semua isi tabel pesan akan ditampilkan. Jika data pelanggan tidak ada di tabel pelanggan, maka isi tabel pesan tetap ditampilkan.

Menggabungkan 3 Tabel atau Lebih

Untuk menggabungkan tiga tabel atau lebih, pada dasarnya sama dengan penggabungan 2 (dua) tabel. Sebagai contoh misalnya kita akan menampilkan barang-barang yang dipesan beserta nama barang dan harganya untuk pemesanan dengan nomor 1, sedemikian sehingga hasilnya menjadi sebagai berikut:

id_pesan	id_produk	nm_produk	harga	jumlah
1	B0001	Buku Tulis	2700	2
1	B0003	Penggaris	3000	3
1	B0004	Pensil	2000	1

Contoh Hasil Penggabungan 3 Tabel

Bagaimana perintah SQL-nya?

SELF JOIN

Membuat Self-Join

Salah satu alasan utama menggunakan alias tabel adalah untuk merujuk kepada tabel yang sama lebih dari satu kali pada statemen SELECT tunggal. Ada contoh untuk menunjukkan hal ini. Misalkan Anda ingin mengirim sebuah surat ke semua kontak pelanggan yang bekerja untuk perusahaan yang sama dimana Jim Jones bekerja. Query ini mengharuskan Anda pertama-tama mencari perusahaan tempat Jim Jones bekerja, dan selanjutnya pelanggan mana yang bekerja untuk perusahaan tersebut. Berikut ini satu cara untuk menyelesaikan masalah tersebut :

```
SELECT c1.KodeCus, c1.Nama, c1.Kontak FROM Customer c1,
Customer c2 WHERE c1.Nama = c2.Nama AND c2.Kontak='Jim Jones'
```

Dua tabel yang dibutuhkan pada query ini sebenarnya adalah tabel yang sama, dan karena itu tabel Customer muncul dua kali pada klausa FROM. Meskipun ini sah sama sekali, tetapi semua referensi ke tabel Customer akan ambigous karena DBMS tidak mengetahui tabel Customer yang Anda rujuk.
Untuk memecahkan masalah ini, digunakan alias-alias tabel. Kejadian pertama pada Customer punya alias c1, dan kejadian kedua mempunyai alias c2. Sekarang alias-alias itu dapat digunakan sebagai nama tabel. Statemen SELECT, misalnya, menggunakan prefiks c1 untuk menyatakan eksplisit nama lengkap dari field yang diinginkan. Jika ini tidak dilakukan, DBMS akan menghasilkan sebuah error karena ada dua field bernama KodeSup, Nama dan Kontak.
DBMS tidak dapat mengetahui yang mana yang Anda inginkan (bahkan sekalipun field itu hanya satu dan sama). Klausa WHERE pertama-tama menghubungkan tabel,

dan kemudian menyaring data dengan Kontak pada tabel kedua untuk mengembalikan hanya data yang diinginkan.

Natural Join

Kapanpun tabel digabungkan, sedikitnya ada satu field yang akan muncul pada lebih dari satu tabel (field yang digabungkan). Penggabungan standar (INNER JOIN yang Anda pelajari diatas) mengembalikan semua data, bahkan banyak kejadian pada field yang sama. NATURAL JOIN hanya mengurangi kejadian tersebut sehingga hanya satu pada setiap field yang dikembalikan.

Sebenarnya bukan NATURAL JOIN yang melakukan hal tersebut, melainkan Anda yang melakukan. NATURAL JOIN adalah join dimana dilakukan dengan menggunakan wildcard (SELECT *) untuk satu tabel dan sub-kumpulan eksplisit dari field untuk semua tabel lainnya. Berikut ini contohnya :

```
SELECT C.*, P.NoFaktur, P.Tanggal, BJ.KodeBrg, BJ.Jumlah,
BJ.Harga FROM Customer C, Penjualan P, BarangJual BJ WHERE
C.KodeCus = P.KodeCus   AND BJ.NoFaktur = P.NoFaktur   AND
KodeBrg = 'RGAN01'
```

Pada Contoh tersebut, wildcard digunakan hanya untuk tabel yang pertama. Seluruh field lainnya dengan jelas di daftar sehingga tidak ada field duplikat yang di dapat kembali.

Sebenarnya, setiap INNER JOIN yang Anda buah sejauh ini adalah NATURAL JOIN, dan Anda tidak pernah membutuhkan INNER JOIN yang bukan NATURAL JOIN.

Outer Join

Kebanyakan join menghubungkan record-record dalam satu tabel dengan record-record tabel lainnya. Tetapi kadang-kadang Anda perlu memasukkan record yang tidak mempunyai record-record yang berhubungan. Sebagai contoh, Anda mungkin menggunakan join untuk menyelesaikan tugas berikut :

- Menghitung berapa banyak pesanan yang dibuat oleh setiap pelanggan, termasuk pelanggan yang sudah melakukan pesanan.
- Mendaftar semua barang dengan banyaknya pesanan, termasuk barang yang tidak dipesan oleh siapapun.
- Menjumlah rata-rata ukuran penjualan, mencakup rekening pelanggan yang belum melakukan pesanan.

Pada setiap contoh tersebut, join memasukkan record-record tabel yang tidak mempunyai record berkaitan dalam tabel terkait. Tipe join ini disebut OUTER JOIN. Statemen SELECT berikut adalah INNER JOIN. Statemen ini mendapatkan kembali daftar semua pelanggan dan pesanan mereka :

```
SELECT Customer.KodeCus, Penjualan.NoFaktur FROM Customer
INNER JOIN Penjualan ON Customer.KodeCus = Penjualan.KodeCus
```

Untuk mendapatkan kembali daftar semua pelanggan, termasuk pelanggan yang tidak membuat pesanan, Anda dapat melakukan hal berikut :

```
SELECT Customer.KodeCus, Penjualan.NoFaktur FROM Customer LEFT
OUTER JOIN Penjualan ON Customer.KodeCus = Penjualan.KodeCus
```

Seperti INNER JOIN, statemen SELECT ini menggunakan keyword OUTER JOIN untuk menentukan tipe join. Tetapi tidak seperti INNER JOIN, yang menghubungkan record pada kedua tabel, OUTER JOIN juga memasukkan record dengan yang tidak berhubungan. Dan seperti terlihat pada contoh sebelumnya, ketika membuat OUTER JOIN, Anda harus menentukan tabel dari mana Anda memasukkan semua record. Jika Anda menggunakan sintaks OUTER JOIN, Anda harus menggunakan keyword RIGHT atau LEFT. Contoh diatas menggunakan LEFT OUTER JOIN untuk memilih semua record dari tabel pada bagian kiri klausa FROM (tabel Customer). Untuk memilih semua record dari tabel pada bagian kanan, Anda dapat menggunakan RIGHT OUTER JOIN seperti terlihat pada contoh berikut :

```
SELECT Customer.KodeCus, Penjualan.NoFaktur FROM Customer
RIGHT OUTER JOIN Penjualan ON Customer.KodeCus =
Penjualan.KodeCus
```

memperhatikan bentuk OUTER JOIN yang digunakan, selalu ada dua bentuk dasar pada OUTER JOIN, yaitu LEFT OUTER JOIN dan RIGHT OUTER JOIN. Perbedaan di antara keduanya hanya urutan tabel yang akan dihubungkan. Dengan demikian, dua tipe OUTER JOIN dapat digunakan secara bergantian, dan keputusan tentang mana yang digunakan semata-mata tergantung pada kenyamanan.

- See more at: http://abdulmail.blogspot.com/2011/04/join-tabel-di-sql.html#sthash.fzTG1CEV.dpuf

Perintah SQL untuk menghapus data dari banyak tabel secara bersamaan

March 12, 2013 Rayyan MySQL, PHP, 5

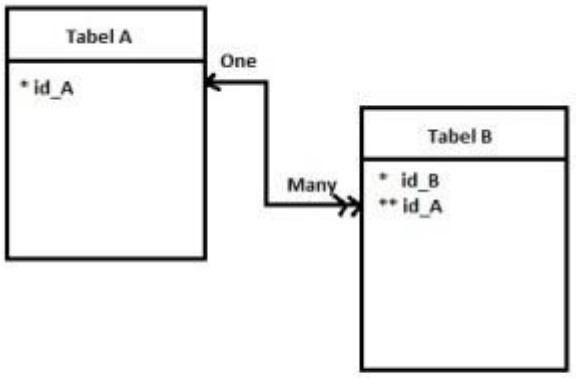
Untuk mempermudah dalam proses Penghapusan data pada tabel yang saling ber-relasi, dibutuhkan beberapa teknik untuk mempersingkat waktu proses dan mengurangi jumlah baris program yang Kita ketik. Pada perintah SQL hal itu sangat dimungkinkan, dengan syarat data yang akan dihapus merupakan data yang saling ber-relasi.

Beberapa pengalaman yang pernah saya jumpai dalam beberapa kasus, untuk melakukan proses hapus data saya harus melakukannya satu persatu pada tabel B, dimana setiap prosesnya akan melewati tabel A terlebih dahulu. Hal ini terjadi karena adanya relasi pada kedua tabel tersebut.

Jika teman-teman pernah mengalami kasus yang sama dengan yang saya alami, berikut cara untuk mengatasi kegundahan hati mu :

Pertama, pahami relasi yang ada pada ke dua tabel Anda, contoh pada tabel A memiliki Kunci Utama (*Primary Key*): id_A, dan pada tabel B memiliki Kunci Utama id_B dan sekaligus memiliki Kunci Tamu (*Secondary Key*): id_A.

Perhatikan Simulasi Gambar di bawah ini :



Relasi 2 Tabel

Gambar di samping memberikan Notasi seperti cerita Saya di atas tadi. Nah jika teman-teman memiliki jenis relasi seperti di samping, dan ingin menghapus salah satu data di dalam dua tabel tersebut, silahkan ikuti terus kelanjutan cerita sangkuriang ini..:

Perintah SQL yang dibutuhkan untuk menghapus data dari dua tabel untuk kasus tersebut seperti berikut :

Sintak SQL yang digunakan:

view source
print?

```
1 // Hapus 1 Data dengan Kondisi tertentu
2 DELETE FROM Alias_A, Alias_B USING Tabel_A AS Alias_A INNER JOIN
  Tabel_B AS Alias_B
3 WHERE Alias_A.field_A=Alias_B.field_B AND
  Alias_A.field_primary='isi_field'
4
5 // Hapus seluruh Data di dua tabel
6 DELETE FROM Alias_A, Alias_B USING Tabel_A AS Alias_A INNER JOIN
  Tabel_B AS Alias_B
7 WHERE Alias_A.field_A=Alias_B.field_B
```

Cukup Simpel ya, silahkan di Praktekkan dan perhatikan apa yang terjadi pada 2 tabel Anda, cara ini sudah banyak saya praktekkan di beberapa projek yang saya kerjakan.