Perintah Multitable (Outer Join dan Implisit Join)

A. Perintah SQL untuk Mengakses Multitable

Pada dasarnya, *SELECT* merupakan perintah DML (*Data Manipulation Language*) untuk menghasilkan *record set* dengan mengekstrak data dari *database*. Perintah *SELECT* sering dipakai dalam sebuah *database* RDBMS (*Relational Database Management System*) dan memiliki bentuk yang kompleks. Format perintah *SELECT* yang digunakan adalah sebagai berikut.

SELECT select_list [INTO new_table] FROM table_source [WHERE search_condition] [GROUP BY group_by_expression] [HAVING search_condition] [ORDER BY order_expression [ASC | DESC]].

Misalnya, kita memiliki tabel **dt_mobil** dengan

B. Perintah Select Bertingkat (*Multilevel Select Command*)

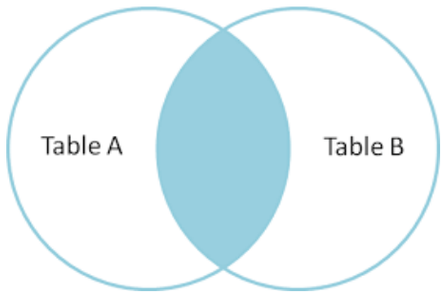
1. Perintah Join (*Join Command*)

Pada umumnya, mengambil data bisa dilakukan dengan cara mengakses beberapa table secara bersamaan. Salah satu metode yang digunakan untuk mengambil data dari beberapa tabel sekaligus dapat menggunakan perintah *join*. Perintah *join* (*join command*) pada SQL berfungsi menghubungkan tabel yang satu dengan tabel yang lain sehingga saling berhubungan atau berelasi satu sama lain. Hal ini bertujuan untuk menampilkan beberapa data dari tabel yang berbeda dengan menggunakan satu perintah. Perintah *join* diklasifikasikan menjadi sebagai berikut.

a. **Perintah *Inner Join***

Data yang akan ditampilkan pada *inner join* hanya data yang memiliki pasangan saja, sedangkan data pada tabel yang tidak memiliki sebuah kesamaan maka data tersebut tidak akan ditampilkan. *Inner Join* identik dengan sebuah perintah yang digunakan untuk menampilkan sebuah data atau *record* dengan menghubungkan dua tabel atau lebih dalam satu perintah. Tabel akan digabungkan dua arah menggunakan *inner join*, sehingga tidak ada data yang *NULL* di satu sisi.
Format penulisan yang digunakan adalah

```
SELECT column FROM TableP INNER JOIN TableQ ON P.columnName = Q.columnName;
```



Misalnya, menggabungkan tabel dt_mobil dan dt_servis dengan tujuan untuk menampilkan daftar nm_pelanggan yang pernah melakukan perbaikan (servis).

1) Penggabungan dengan *where*

Format penggabungan dengan WHERE menggunkan perintah:
SELECT tabelP.*, tabelQ.* FROM tabelP, tabelQ WHERE tabelP.PK=tabelQ.FK;
Misalnya, perintah SQL yang digunakan untuk menggabungkan tabel **dt_mobil** dan **dt_servis** sebagai berikut:

```
SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv,
dt_servis.tgl_servis FROM dt_mobil, dt_servis WHERE
dt_mobil.nopol=dt_servis.nopol;
```

Hasilnya sebagai berikut:

```
mysql> select * from dt_mobil;
+----------+--------------+---------------+
| nopol    | nm_pelanggan | merek         |
+----------+--------------+---------------+
| K 11 PWF | Oktyawati    | Toyota        |
| N 1 SA   | Maheswari    | Mercedez Benz |
| WA 2 N   | Diprayogi LS | Mitsubishi    |
| DW 1 WJT | Rangga Haris | Honda         |
+----------+--------------+---------------+
4 rows in set (0.00 sec)

mysql> select * from dt_servis;
+--------+----------+------------+----------+-------+-------+
| id_srv | nopol    | tgl_servis | model    | tipe  | warna |
+--------+----------+------------+----------+-------+-------+
|      1 | K 11 PWF | 2019-04-02 | FORTUNER | SUV   | GREY  |
|      2 | N 1 SA   | 2019-04-05 | AMG GTR  | SPORT | GREY  |
|      3 | N 1 SA   | 2019-04-10 | AMG GTR  | SPORT | GREY  |
|      4 | WA 2 N   | 2019-01-20 | CITY     | SEDAN | HITAM |
|      5 | DW 1 WJT | 2018-12-14 | FORTUNER | SUV   | GREY  |
+--------+----------+------------+----------+-------+-------+
5 rows in set (0.00 sec)

mysql> SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis
.tgl_servis FROM dt_mobil, dt_servis WHERE dt_mobil.nopol=dt_servis.nopol;
+----------+--------------+--------+------------+
| nopol    | nm_pelanggan | id_srv | tgl_servis |
+----------+--------------+--------+------------+
| K 11 PWF | Oktyawati    |      1 | 2019-04-02 |
| N 1 SA   | Maheswari    |      2 | 2019-04-05 |
| N 1 SA   | Maheswari    |      3 | 2019-04-10 |
| WA 2 N   | Diprayogi LS |      4 | 2019-01-20 |
| DW 1 WJT | Rangga Haris |      5 | 2018-12-14 |
+----------+--------------+--------+------------+
5 rows in set (0.37 sec)
```

Berdasarkan hasil query diatas, terdapat lima transaksi servis yang dilakukan oleh tiga mobil **dt_mobil**. Namun, terdapat satu **mobil** yaitu **dt_mobil WA 2 N** tidak ditampilkan, karena belum pernah melakukan transaksi servis.

2) Penggabungan dengan *Inner Join*

Format penggabungan dengan WHERE menggunkan perintah:

SELECT tabelP.*, tabelQ.* FROM tabelP INNER JOIN tabelQ ON tabelP.PK=tabelQ.FK;

Misalnya, perintah SQL yang digunakan untuk menggabungkan tabel **dt_mobil** dan **dt_servis** sebagai berikut:

```
SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv,
dt_servis.tgl_servis FROM dt_mobil INNER JOIN dt_servis ON
dt_mobil.nopol=dt_servis.nopol;
```

Hasilnya adalah sama seperti penggabungan dengan *WHERE*. Perhatikan gambar berikut:

```
mysql> SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis.tgl_servis FROM
dt_mobil INNER JOIN dt_servis ON dt_mobil.nopol=dt_servis.nopol;
+----------+--------------+--------+------------+
| nopol    | nm_pelanggan | id_srv | tgl_servis |
+----------+--------------+--------+------------+
| K 11 PWF | Oktyawati    |      1 | 2019-04-02 |
| N 1 SA   | Maheswari    |      2 | 2019-04-05 |
| N 1 SA   | Maheswari    |      3 | 2019-04-10 |
| WA 2 N   | Diprayogi LS |      4 | 2019-01-20 |
| DW 1 WJT | Rangga Haris |      5 | 2018-12-14 |
+----------+--------------+--------+------------+
5 rows in set (0.00 sec)
```

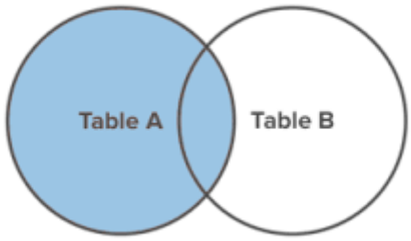
b. **Perintah *Outer Join***

Perintah jenis ini digunakan untuk tabel yang digabungkan dengan *outer join* menjadi satu arah, sehinggaterdapat data *NULL* (kosong) di satu sisi. Misalnya menggabungkan tabel **dt_mobil** dan **dt_servis** untuk menampilkan daftar nm_pelanggan yang melakukan **perbaikan (servis)**. Secara umum, *outer join* terbagi menjadi dua, yaitu sebagai berikut:

1) **Perintah *Left Join***

Perintah *left join* (*left outer join*) identik dengan suatu perintah yang digunakan untuk mengembalikan semua nilai dari tabel kiri ditambah dengan nilai dari tabel kanan yang sesuai (atau *null* jika tidak ada nilai sesuai). Pada dasarnya, fungsi dari *left join* hampir sama dengan *inner join*, hanya saja pada *right join* membuat sebuah parameter di sebelah kanan, maka pada *left join* justru

akan membuat sebuah parameter dari tabel sebelah kiri, dan jika terdapat data atau *record* yang kososng maupun tidak berelasi akan berisi *null* di sebelah kanan.



Format penggabungan dengan *left join* menggunkan perintah berikut:
SELECT tabelP.*, tabelQ.* FROM tabelP LEFT JOIN tabelQ ON tabelP.PK=tabelQ.FK;
Adapun format penulisan *query* yang digunakan adalah sebagai berikut:

```
SELECT * FROM tabelP LEFT JOIN tabelQ ON tabelP.PK = tabelQ.FK
SELECT * FROM POST LEFT JOIN category on post.category_id= category.id_category;
```

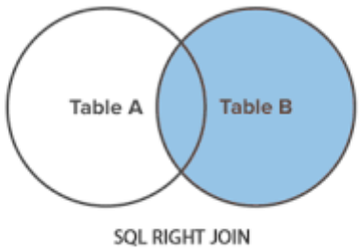
Misalnya, perintah SQL menggabungkan tabel **dt_mobil** dan **dt_servis** menggunakan *left join* adalah **SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis.tgl_servis FROM dt_mobil LEFT JOIN dt_servis ON dt_mobil.nopol=dt_servis.nopol;**

```
mysql> SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis.tgl_servis
 FROM dt_mobil LEFT JOIN dt_servis ON dt_mobil.nopol=dt_servis.nopol;
+----------+--------------+--------+------------+
| nopol    | nm_pelanggan | id_srv | tgl_servis |
+----------+--------------+--------+------------+
| K 11 PWF | Oktyawati    |      1 | 2019-04-02 |
| N 1 SA   | Maheswari    |      2 | 2019-04-05 |
| N 1 SA   | Maheswari    |      3 | 2019-04-10 |
| WA 2 N   | Diprayogi LS |      4 | 2019-01-20 |
| DW 1 WJT | Rangga Haris |      5 | 2018-12-14 |
| A 1 FA   | Oong         |   NULL | NULL       |
+----------+--------------+--------+------------+
6 rows in set (0.00 sec)
```

Penggunaan *left join* akan menampilkan data **dt_mobil** dengan id **A 1 FA**, walaupun **dt_mobil** tersebut belum pernah melakukan transaksi servis. Pada kolom **id_srv** dan **tgl_servis** untuk **nm_pelanggan A 1 FA** isinya *NULL* yang berarti pada tabel **dt_servis** untuk **nm_pelanggan** tersebut benar-benar tidak ada.

2) **Perintah *Right Join***

Teknik ini merupakan kebalikan dari *left join* meskipun fungsi dari *right join* hampir sama dengan *inner join*, hanya posisi *right join* akan membuat sebuah parameter pada sebelah kanan jika data pada tabel terdapat data atau *record* yang kosong (tidak berelasi), maka secara otomatis akan berisi *NULL*.



Format penggabungan dengan *right join* menggunkan perintah berikut:

```
SELECT * FROM tabelP RIGHT JOIN tabelQ ON tabelP.PK = tabelQ.FK
SELECT * FROM POST RIGHT JOIN category on post.category_id= category.id_category;
```

Misalnya, perintah SQL menggabungkan tabel **dt_mobil** dan **dt_servis** menggunakan *right join*, perintah yang digunakan adalah
SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis.tgl_servis FROM dt_mobil RIGHT JOIN dt_servis ON dt_mobil.nopol=dt_servis.nopol;
Berdasarkan format di atas, tabel yang menjadi acuan pada *right join* adalah tabel sebelah kanan (tabel **dt_servis**), sehingga semua isi tabel **dt_servis** akan ditampilkan. Meskipun data

nm_pelanggan tidak ada di tabel **dt_mobil**, tetapi isi tabel **dt_servis** tetap ditampilkan. Hasilnya adalah sebagai berikut.

```
mysql> SELECT dt_mobil.nopol, dt_mobil.nm_pelanggan, dt_servis.id_srv, dt_servis.tgl_servis
 FROM dt_mobil RIGHT JOIN dt_servis ON dt_mobil.nopol=dt_servis.nopol;
+----------+--------------+--------+------------+
| nopol    | nm_pelanggan | id_srv | tgl_servis |
+----------+--------------+--------+------------+
| K 11 PWF | Oktyawati    |      1 | 2019-04-02 |
| N 1 SA   | Maheswari    |      2 | 2019-04-05 |
| N 1 SA   | Maheswari    |      3 | 2019-04-10 |
| WA 2 N   | Diprayogi LS |      4 | 2019-01-20 |
| DW 1 WJT | Rangga Haris |      5 | 2018-12-14 |
+----------+--------------+--------+------------+
5 rows in set (0.00 sec)
```

c. Perintah *Full Join*

Perintah *full join* (sering disebut *full outer join*) akan mengembalikan seluruh baris dari kedua tabel yang kenai mode *ON* termasuk data-data yang bernilai *NULL*. Dalam hal ini, *full join* akan menghasilkan baris data jika ada data yang sama pada salah satu *table*. Format penggabungan *full join* menggunakan perintah berikut.

SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2 column_name.