BAB IX
MENGAKSES MULTIPLE TABEL

MYSQL mendukung beberapa metode yang dapat mengakses beberapa tabel sekaligus dengan sebuah satetement SQL. Cara pertama bis adengan menggunakan perintah join atau dapat menambah subquery pada pada statement utama SQL. Untuk lebih jelasnya kita akan melihat beberapa contoh dibawah ini.

IX.1 Joining tabel pada statement SELECT

Pada kasusu SELECT yang ini, kita akan mebuat link antara 2 tabel dengan fungsi join. Secara umum syntax nya yaitu :

```
<select statement>::=
SELECT
[<select option> [<select option>...]]
{* | <select list>}
[
FROM {<table reference> | <join definition>}
[WHERE <expression> [{<operator> <expression>}...]]
[GROUP BY <group by definition>]
[HAVING <expression> [{<operator> <expression>}...]]
[ORDER BY <order by definition>]
[LIMIT [<offset>,] <row count>]
]
<join definition>::=
{<table reference>, <table reference> [{, <table reference>}...]}
| {<table reference> [INNER | CROSS ] JOIN <table reference> [<join condition>]}
| {<table reference> STRAIGHT_JOIN <table reference>}
| {<table reference> LEFT [OUTER] JOIN <table reference> [<join condition>]}
| {<table reference> RIGHT [OUTER] JOIN <table reference> [<join condition>]}
| {<table reference> NATURAL [{LEFT | RIGHT} [OUTER]] JOIN <table reference>}
<table reference>::=
<table name> [[AS] <alias>]
```

[{USE | IGNORE | FORCE} INDEX <index name> [{, <index name>}...]]

<join condition>::=

ON <expression> [{<operator> <expression>}...]

| USING (<column> [{, <column>}...])

Sebagian syntax yang ada diatas mungkin sudah terdefinisi pada bab 7,namun tambahan clausa JOIN yang ada yaitu peluasan tentang clausa FORM. Dan clausa yang anda lihat sekarang yaitu

FROM {<table reference> | <join definition>}

dimana clausa yang aslinya

FROM <table reference> [{, <table reference>}...]

Adapun penjelasan syntax tentang clausa join yang ada sebelumnya yaitu

<join definition>::=

{<table reference>, <table reference> [{, <table reference>}...]}

| {<table reference> [INNER | CROSS] JOIN <table reference> [<join condition>]}

| {<table reference> STRAIGHT_JOIN <table reference>}

| {<table reference> LEFT [OUTER] JOIN <table reference> [<join condition>]}

| {<table reference> RIGHT [OUTER] JOIN <table reference> [<join condition>]}

| {<table reference> NATURAL [{LEFT | RIGHT} [OUTER]] JOIN <table reference>}

Untuk lebih lengkapnya,kita akan implementasikan pada contoh berikut ini:

```
CREATE TABLE Music
(
  MusicID  INT NOT NULL PRIMARY KEY,
  Judul  VARCHAR (50) NOT NULL,
  TahunTerbit  YEAR NOT NULL
)
ENGINE=INNODB;
```

Kemudian isi tabel tersebut :

```
INSERT INTO Music VALUES (11001," Tak Ada Yang Abadi",2005),
(22002," Diantara Kalian",2006),
(33003," Jangan Menyerah",2005),
(44004," Sang Mantan",2008),
(55005," Saat Kau Jauh",2009),
(66006," Okelah Klo Begitu",2007),
(77007," Menghapus Jejakmu",2007);
```

Lalu kita buat tabel kedua

```
CREATE TABLE Studio
(
  StudioID  INT NOT NULL PRIMARY KEY,
  NamaStudio  VARCHAR (50) NOT NULL,
  PemilikStudio  VARCHAR (50) NOT NULL
)
ENGINE=INNODB;
```

Kemudian isi tabel Studio

```
INSERT INTO Studio VALUES (0010," Sony Music"," Sony"),
(0011," Abadi  Music","Abdi"),
(0012," Sunny Music ","Mutia"),
(0013," Ceria Studio"," Risna "),
(0014," Persib Studio","Kadiman"),
(0015," Burning Studio ","Audy"),
(0016," Logitech Spectrum"," Taufik ");
```

Kemudian kita buat tabel selanjutnya

```
CREATE TABLE Album
(
  StudioID  INT NOT NULL,
  MusicID  INT NOT NULL,
  PRIMARY KEY (StudioID, MusicID),
```

```
  FOREIGN KEY (StudioID)  REFERENCES Studio (StudioID),
  FOREIGN KEY (MusicID)  REFERENCES Music (MusicID)
)
ENGINE=INNODB;
```

Lalu isi tabel Album

```
INSERT INTO Album VALUES (0010,33003), (0011,44004), (0012,77007),
(0013,11001), (0014,22002), (0015,55005), (0016,66006);
```

Jika dilihat dari tabel keseluruhan, bahwa relasi N ke N terjadi antar tabel Music dan Studio sehingga terbentuk table baru berupa tabel Album yang berisi primary key kedua tabel tersebut. Dan inilah fungsi JOIN yang ada pada fungsi MSQL.

IX.1.1 Membuat FULL JOIN

a) Basic JOIN

Secara umum syntax pada clausa ini yaitu

```
<table reference>, <table reference> [{, <table reference>}...]
```

Sehingga pemanggilan pada tabel pada SQL yang sudah kita buat tadi yaitu

```
SELECT Judul, TahunTerbit, StudioID
FROM Music, Studio
ORDER BY Judul;
```

Statement ini sekilas hampir serupa dengan statement SELECT yang dijelaskan pada bab sebelumnya, namun pada statement ini menggunakan 2 tabel yang memiliki relasi. Jika dilihat Judul dan TahunTerbit berada pada tabel Music da StudioID pad pada tabel Studio. Sehingga yang akan ditampilkan sebagai berikut

```
+--------------------+-------------+----------+
| Judul              | TahunTerbit | StudioID |
+--------------------+-------------+----------+
| Diantara Kalian    |        2006 |       16 |
| Diantara Kalian    |        2006 |       13 |
| Diantara Kalian    |        2006 |       10 |
```

Diantara Kalian	2006	15
Diantara Kalian	2006	12
Diantara Kalian	2006	14
Diantara Kalian	2006	11
Jangan Menyerah	2005	14
Jangan Menyerah	2005	11
Jangan Menyerah	2005	16
Jangan Menyerah	2005	13
Jangan Menyerah	2005	10
Jangan Menyerah	2005	15
Jangan Menyerah	2005	12
Menghapus Jejakmu	2007	11
Menghapus Jejakmu	2007	16
Menghapus Jejakmu	2007	13
Menghapus Jejakmu	2007	10
Menghapus Jejakmu	2007	15
Menghapus Jejakmu	2007	12
Menghapus Jejakmu	2007	14
Okelah Klo Begitu	2007	16
Okelah Klo Begitu	2007	13
Okelah Klo Begitu	2007	10
Okelah Klo Begitu	2007	15
Okelah Klo Begitu	2007	12
Okelah Klo Begitu	2007	14
Okelah Klo Begitu	2007	11
Saat Kau Jauh	2009	15
Saat Kau Jauh	2009	12
Saat Kau Jauh	2009	14
Saat Kau Jauh	2009	11
Saat Kau Jauh	2009	16
Saat Kau Jauh	2009	13
Saat Kau Jauh	2009	10
Sang Mantan	2008	14
Sang Mantan	2008	11

Sang Mantan	2008	16
Sang Mantan	2008	13
Sang Mantan	2008	10
Sang Mantan	2008	15
Sang Mantan	2008	12
Tak Ada Yang Abadi	2005	10
Tak Ada Yang Abadi	2005	15
Tak Ada Yang Abadi	2005	12
Tak Ada Yang Abadi	2005	14
Tak Ada Yang Abadi	2005	11
Tak Ada Yang Abadi	2005	16
Tak Ada Yang Abadi	2005	13

49 rows in set (0.00 sec)

Hasil ini sangat tidak efektif Karen setiap baris yang ada di studio akan di pasangkan pada setiap baris yang ada di Music atau dapat dikatakan sebagai Cartesian Product. Dan untuk meminimkan duplikasi maka syntaxnya

```
SELECT Judul, TahunTerbit, StudioID
FROM Music AS M, Album AS A
WHERE M.MusicID = A.MusicID
ORDER BY Judul;
```

Pada ssat menambahkan kondisi yang ada WHERE, maka hasil yang akan ditampilkan

Judul	TahunTerbit	StudioID
Diantara Kalian	2006	14
Jangan Menyerah	2005	10
Menghapus Jejakmu	2007	12
Okelah Klo Begitu	2007	16
Saat Kau Jauh	2009	15
Sang Mantan	2008	11

```
| Tak Ada Yang Abadi |       2005 |      13 |
+--------------------+------------+---------+
7 rows in set (0.02 sec)
```

Dan pada kondisi WHERE yang sudah kita buat dapat dikombinasikan dengan operator lain seperti berikut

```
SELECT Judul, TahunTerbit, A.StudioID
FROM Music AS M, Album AS A
WHERE M.MusicID = A.MusicID AND TahunTerbit<2008
ORDER BY Judul;
```

Dan data yang akan ditampilkan

```
+--------------------+------------+---------+
| Judul              | TahunTerbit | StudioID |
+--------------------+------------+---------+
| Diantara Kalian    |       2006 |      14 |
| Jangan Menyerah    |       2005 |      10 |
| Menghapus Jejakmu  |       2007 |      12 |
| Okelah Klo Begitu  |       2007 |      16 |
| Tak Ada Yang Abadi |       2005 |      13 |
+--------------------+------------+---------+
5 rows in set (0.00 sec)
```

Dan jika lebih dari 2 tabel maka syntaxnya adalah

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M, Studio AS S, Album AS A
WHERE M.MusicID = A.MusicID  AND S.StudioID = A.StudioID AND
TahunTerbit<2006
ORDER BY Judul;
```

Data yang akan ditampilkan

```
+--------------------+------------+--------+
| Judul              | TahunTerbit | Studio |
```

```
+--------------------+-------------+--------+
| Jangan Menyerah    |       2005 |     10 |
| Tak Ada Yang Abadi |       2005 |     13 |
+--------------------+-------------+--------+
2 rows in set (0.02 sec)
```

IX.1.2 Membuat Inner Join dan Cross Join

Proses Inner join dan Cross join pada umumnya menghasilkan nilai yang sama dengan basic koin, namun dengan adanya dua proses tersebut dapat menyederhanakan syntax yang panjang pada proses basic join, sehingga penggunaan inner join dan Cross join dapat lebih praktis digunkan dibandingkan dengan basic join. Secara umum syntax inner dan cross join adalah sebagai berikut

```
<table reference> [INNER | CROSS ] JOIN <table reference> [<join condition>]
<join condition>::=
ON <expression> [{<operator> <expression>}...]
| USING (<column> [{, <column>}...])
```

Selanjutnya kita akan menggunakan JOIN, INNER JOIN dan CROSS JOIN.

```
SELECT Judul, StudioID FROM Music JOIN Studio;
SELECT Judul, StudioID FROM Music INNER JOIN Studio;
SELECT Judul, StudioID FROM Music CROSS JOIN Studio;
```

setiap syntax diatas meghasilkan nilai yang sama, namun dalam menetukan hasil yang diinginkan kita harus menggunakan kondisi lagi seperti WHERE. Namun dalam INNER JOIN dan CROSS JOIN yang digunkan adalah USING atau ON sebagai pengganti WHERE. Inilah penggunaannya

```
SELECT Judul, TahunTerbit, A.StudioID
FROM Music AS M JOIN Album AS A
ON M.MusicID = A.MusicID
ORDER BY Judul;
```

Atau dengan syntax berikut

```
SELECT Judul, TahunTerbit, A.StudioID
FROM Music AS M JOIN Album AS A
USING (MusicID)
ORDER BY Judul;
```

Dan kedua syntax tersebut akan menampilkan data yang sama

```
+--------------------+-------------+----------+
| Judul              | TahunTerbit | StudioID |
+--------------------+-------------+----------+
| Diantara Kalian    |        2006 |       14 |
| Jangan Menyerah    |        2005 |       10 |
| Menghapus Jejakmu  |        2007 |       12 |
| Okelah Klo Begitu  |        2007 |       16 |
| Saat Kau Jauh      |        2009 |       15 |
| Sang Mantan        |        2008 |       11 |
| Tak Ada Yang Abadi |        2005 |       13 |
+--------------------+-------------+----------+
```

7 rows in set (0.00 sec)

Dengan basic join, statement SELECT dapat juga dikombinasikan dengan clausa WHERE sebagai kondisi tambahan untuk lebih membatasi data yang akan diperoleh.

```
SELECT Judul, TahunTerbit, A.StudioID
FROM Music AS M JOIN Album AS A
ON M.MusicID = A.MusicID
WHERE TahunTerbit<2007
ORDER BY Judul;
```

Dan inilah data yang akan ditampilkan

```
+--------------------+-------------+----------+
| Judul              | TahunTerbit | StudioID |
+--------------------+-------------+----------+
```

```
| Diantara Kalian    |      2006 |      14 |
| Jangan Menyerah    |      2005 |      10 |
| Tak Ada Yang Abadi |      2005 |      13 |
+--------------------+-----------+---------+
```

3 rows in set (0.02 sec)

Pada syntax yang berikut akan diberikan contoh penggunaan INNER JOIN atau CROSS JOIN menggunakan lebih dari dua tabel

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M CROSS JOIN Album AS A ON M.MusicID = A.MusicID
CROSS JOIN Studio AS S ON S.StudioID = A.StudioID
WHERE TahunTerbit<2007
ORDER BY Judul;
```

Atau dengan sintax yang berikut

```
SELECT Judul, TahunTerbit, StudioID AS Studio
FROM Music JOIN Album USING (MusicID)
JOIN Studio USING (StudioID)
WHERE TahunTerbit<2007
ORDER BY Judul;
```

Dan inilah data yang sama akan ditampilkan oleh kedua syntax diatas

```
+--------------------+-----------+--------+
| Judul              | TahunTerbit | Studio |
+--------------------+-----------+--------+
| Diantara Kalian    |      2006 |     14 |
| Jangan Menyerah    |      2005 |     10 |
| Tak Ada Yang Abadi |      2005 |     13 |
+--------------------+-----------+--------+
```

3 rows in set (0.00 sec)

IX.1.3 Membuat STRAIGHT JOIN

Fungsi straight join hampir sama dengan basic join, namun kelebihannya yaitu data pada tabel akan dibaca mulai dari yang sebelah kiri hingga kekanan sehingga lebih optimal. Secara umum syntaxnya yaitu

```
<table reference> STRAIGHT_JOIN <table reference>
```

Berikut merupakan implementasinya

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M STRAIGHT_JOIN Studio AS S STRAIGHT_JOIN Album AS A
WHERE M.MusicID = A.MusicID AND S.StudioID = A.StudioID AND
TahunTerbit<2008
ORDER BY Judul;
```

Pada STRAIGHT JOIN pembuatan kondisi kembalai ke WHERE sehingga USING dan ON tidak digunakan. Adapun metode lain dalam menggunakan syntax ini

```
SELECT STRAIGHT_JOIN  Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M , Studio AS S , Album AS A
WHERE M.MusicID = A.MusicID AND S.StudioID = A.StudioID AND
TahunTerbit<2008
ORDER BY Judul;
```

Syntax yg diatas merupakan kombinasi antara basic join dan straight join.

IX.1.4 Membuat OUTER JOIN

Fungsi dari outer join yaitu untuk menampilkan seluruh data yang berelasi (minimal 2 tabel) dimana terdapat salah satu atau lebih data yang tidak memiliki pasangan di tabel lain. Metode yang digunakan ada dua yaitu LEFT JOIN dan RIGHT JOIN.

a) LEFT JOIN

Syntax untuk left join hampr sam dengan INNER atau CROSS join, namun sekarang kita harus memasukkan keywordnya. Inilah syntax secara umum.

<table reference> LEFT [OUTER] JOIN <table reference> [<join condition>]

<join condition>::=

ON <expression> [{<operator> <expression>}...]

| USING (<column> [{, <column>}...])

Inilah implementasinya

```
SELECT Judul, TahunTerbit, StudioID
FROM Music AS M LEFT JOIN Album AS A
ON M.MusicID = A.MusicID
ORDER BY Judul;
```

Atau dengan menggunakan syntax ini

```
SELECT Judul, TahunTerbit, StudioID
FROM Music AS M LEFT JOIN Album AS A
USING  (MusicID )
ORDER BY Judul;
```

Data yang akan ditampilkan oleh kedua syntax ini akan sama seperti dibawah ini

```
+--------------------+-------------+----------+
| Judul              | TahunTerbit | StudioID |
+--------------------+-------------+----------+
|  Diantara Kalian   |       2006 |       14 |
|  Jangan Menyerah   |       2005 |       10 |
|  Menghapus Jejakmu |       2007 |       12 |
|  Okelah Klo Begitu |       2007 |       16 |
|  Saat Kau Jauh     |       2009 |       15 |
|  Sang Mantan       |       2008 |       11 |
|  Tak Ada Yang Abadi |       2005 |       13 |
+--------------------+-------------+----------+
7 rows in set (0.00 sec)
```

Data ini memposisikan data tabel Music di sebelah kiri dan data tabel Studio di sebelah kanan.

Contoh berikut implementasi untuk dua tabel yang berelasi

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M LEFT JOIN Album AS A ON M.MusicID = A.MusicID
LEFT JOIN Studio AS S ON A.StudioID = S.StudioID
ORDER BY Judul;
```

Berikut data yang akan ditampilkan

```
+--------------------+-------------+--------+
| Judul              | TahunTerbit | Studio |
+--------------------+-------------+--------+
| Diantara Kalian    |        2006 |     14 |
| Jangan Menyerah    |        2005 |     10 |
| Menghapus Jejakmu  |        2007 |     12 |
| Okelah Klo Begitu  |        2007 |     16 |
| Saat Kau Jauh      |        2009 |     15 |
| Sang Mantan        |        2008 |     11 |
| Tak Ada Yang Abadi |        2005 |     13 |
+--------------------+-------------+--------+
7 rows in set (0.00 sec)
```

b) Membuat RIGHT JOIN

Fungsi RIGHT JOIN merupakan keterbalikan dari LEFT JOIN namu memiliki cara kerja yang sama dan sekarang data yang akan ditampilkan dimulai dari sebelah kanan ke kiri. Inilah syntax secar umumnya

```
<table reference> RIGHT [OUTER] JOIN <table reference> [<join condition>]
<join condition>::=
ON <expression> [{<operator> <expression>}...]
| USING (<column> [{, <column>}...])
```

Berikut implementasinya

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M RIGHT JOIN Album AS A ON M.MusicID = A.MusicID
RIGHT JOIN Studio AS S ON S.StudioID = A.StudioID
ORDER BY TahunTerbit;
```

Dan inilah data yang akan ditampilkan

```
+--------------------+-------------+--------+
| Judul              | TahunTerbit | Studio |
+--------------------+-------------+--------+
| Jangan Menyerah    |        2005 |     10 |
| Tak Ada Yang Abadi |        2005 |     13 |
| Diantara Kalian    |        2006 |     14 |
| Okelah Klo Begitu  |        2007 |     16 |
| Menghapus Jejakmu  |        2007 |     12 |
| Sang Mantan        |        2008 |     11 |
| Saat Kau Jauh      |        2009 |     15 |
+--------------------+-------------+--------+
7 rows in set (0.00 sec)
```

IX.1.5 Membuat NATURAL JOIN

NATURAL JOIN dapat menspesifikasikan letak JOIN (FUULL, RIGHT atau LEFT). Inilah syntax secara umumnya

```
<table reference> NATURAL [{LEFT | RIGHT} [OUTER]] JOIN <table reference>
```

Inilah implementasinya

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M NATURAL JOIN Album AS A
NATURAL JOIN Studio AS S
WHERE TAhunTerbit<2008
ORDER BY Judul;
```

Pada NATURAL JOIN tidak ada kondisi ON ataupun USING. Hal ini dikarenakan NATURAL JOIN secara otomatis menghubungkan kata kunci yang sama anatar tabel yang berelasi. Dan inilah data yang akan ditampilkan

```
+--------------------+-------------+--------+
| Judul              | TahunTerbit | Studio |
+--------------------+-------------+--------+
| Diantara Kalian    |        2006 |     14 |
| Jangan Menyerah    |        2005 |     10 |
| Menghapus Jejakmu  |        2007 |     12 |
| Okelah Klo Begitu  |        2007 |     16 |
| Tak Ada Yang Abadi |        2005 |     13 |
+--------------------+-------------+--------+
```

5 rows in set (0.00 sec)

Jika ingin menampilkan OUTER JOIN namun masih dengan NATURAL JOIN, maka kombinasinya syntaxnya

```
SELECT Judul, TahunTerbit, S.StudioID AS Studio
FROM Music AS M NATURAL LEFT JOIN Album AS A
NATURAL LEFT JOIN Studio AS S
WHERE TahunTerbit<2008
ORDER BY Judul;
```

Maka akan didapatkan data sebagai berikut

```
+--------------------+-------------+--------+
| Judul              | TahunTerbit | Studio |
+--------------------+-------------+--------+
| Diantara Kalian    |        2006 |     14 |
| Jangan Menyerah    |        2005 |     10 |
| Menghapus Jejakmu  |        2007 |     12 |
| Okelah Klo Begitu  |        2007 |     16 |
| Tak Ada Yang Abadi |        2005 |     13 |
+--------------------+-------------+--------+
```

5 rows in set (0.00 sec)

IX.1.6 Joining Tabel pada statement UPDATE dan DELETE

Pada statement UPDATE fungsi basic join dapat juga digunakan. Hal ini dapat dilihat dari syntax yang dimiliki oleh statement UPDATE

<update statement>::=
UPDATE [LOW_PRIORITY] [IGNORE]
<single table update> | <joined table update>
<joined table update>::=
<join definition>
SET <column name>=<expression> [{, <column name>=<expression>}...]
[WHERE <where definition>]

Dan inilah implementasinya

```
UPDATE Album CROSS JOIN Studio USING (StudioID)
CROSS JOIN Music  USING (MusicID)
SET NamaStudio =" Angkasa Rec"
WHERE PemilikStudio = " Sony ";
```

Hasil dari fungsi diatas sama saja seperti proses UPDATE yang biasa, namun pada proses ini disisipkan fungsi join. Dan inilah data yang akan ditampilkan

```
+----------+--------------------+---------------+
| StudioID | NamaStudio         | PemilikStudio |
+----------+--------------------+---------------+
|       10 |  Angkasa Rec       |  Sony         |
|       11 |  Abadi  Music      | Abdi          |
|       12 |  Sunny Music       | Mutia         |
|       13 |  Ceria Studio      |  Risna        |
|       14 |  Persib Studio     | Kadiman       |
|       15 |  Burning Studio    | Audy          |
|       16 |  Logitech Spectrum |  Taufik       |
+----------+--------------------+---------------+
7 rows in set (0.00 sec)
```

Untuk fungsi delete sama seperti fungsi delete yang biasa, namun pada proses ini disispkan fungsi join. Inilah implementasinya

```
DELETE  A, S
FROM Studio AS S INNER JOIN Album AS A ON S.StudioID = A.StudioID
INNER JOIN Music AS M ON M.MusicID = A.MusicID
WHERE PemilikStudio= “Kadiman”;
```

IX.1.7 Membuat Subqueris pada statement SQL

Cara lain dalam mengakses multiple tabel pada statement SELECT, UPDATE dan DELETE yaitu dengan menambahkan subquery pada statement tersebut. Statement ini juga dapat digunakan pada clausa WHERE namun dengan menambahkan operator pembanding dalam clausa tersebut.

Menambahkan Subqueris pada statement SELECT

Cara yang sering digunakan dalam menambahkan subqueris padastatement SELECT yaitu dengan menggunakan clausa WHERE sebagai sarana dalam subqueris tersebut. Dan contohnya

```
SELECT NamaStudio
FROM Studio
WHERE StudioID=
        (
            SELECT A.StudioID
        FROM Album AS A, Music AS M
        WHERE A.MusicID=M.MusicID AND Judul=”Diantara Kalian”
        );
```

Penjelasan :

SELECT <nama tabel serta aliasnya>

FROM <nama tabel>

WHERE <kondisi kolom tabel>=

(

<Pembanding tambahan berupa subqueris>

);

Penggunaan subqueris sering digantikan oleh fungsi join. Contohnya sebagai berikut :

```
SELECT DISTINCT S.StudioID AS Studio
FROM Music AS M JOIN Album AS A ON M.MusicID = A.MusicID
JOIN Studio AS S ON S.StudioID = A.StudioID
WHERE Judul='Sang Mantan';
```

Dalam segi pemanfaatan penggunaan JOIn lebih baik daripada subqueris karena jika menggunakan subqueris an terdapat dua data yang sama, maka data tersebut tidak dapat ditampilkan, berbeda dengan JOIN.
Ini adalah cara untuk menggunakan subquery dalam cara yang lain

```
SELECT Judul,TahunTerbit
FROM Music
WHERE TahunTerbit<(SELECT MAX(Tahunterbit)-1 FROM Music)
ORDER BY Judul;
```

Penjelasan:
SELECT <nama kolom pada tabel>
FROM <nama tabel>
WHERE <kondisi yang menggunakan operator sebagai pembanding dengan subqueris>
ORDER BY <nama kolom pada tabel>;
Dan Hasilnya

```
+--------------------+-------------+
| Judul              | TahunTerbit |
+--------------------+-------------+
| Diantara Kalian    |        2006 |
| Jangan Menyerah    |        2005 |
| Menghapus Jejakmu  |        2007 |
| Okelah Klo Begitu  |        2007 |
```

```
| Tak Ada Yang Abadi |       2005 |
+--------------------+------------+
5 rows in set (0.00 sec)
```

IX.1.7.1 Cara kerja operator pada Subquery

a) Operator Any dan Some

Kedua operator ini memiliki kegunaan yang sama yang dapat membuat suatu ekspresi yang dapat membandingkan sebuah kolom dengan berbagai nilai yang dikembalikan oleh subquery. Contohnya seperti ini

```
SELECT Judul, TahunTerbit
FROM Music
WHERE TahunTerbit > ANY
(
    SELECT M.TahunTerbit
        FROM Music AS M JOIN Album AS A USING (MusicID)
        JOIN Studio AS S USING (StudioID)
        WHERE Judul='Menghapus Jejakmu'
)
ORDER BY TahunTerbit;
```

Operator ALL

Inilah contohnya :

```
SELECT Judul,TahunTerbit
FROM Music
WHERE TahunTerbit > ALL
(
    SELECT M.Tahunterbit
        FROM Music AS M JOIN Album AS A USING (MusicID)
        JOIN Studio AS S USING (StudioID)
        WHERE Judul='Menghapus Jejakmu'
)
ORDER BY Judul;
```

Hasilnya adalah

```
+--------------------+-------------+
| Judul              | TahunTerbit |
+--------------------+-------------+
| Diantara Kalian    |        2006 |
| Jangan Menyerah    |        2005 |
| Menghapus Jejakmu  |        2007 |
| Okelah Klo Begitu  |        2007 |
| Saat Kau Jauh      |        2009 |
| Sang Mantan        |        2008 |
| Tak Ada Yang Abadi |        2005 |
+--------------------+-------------+
7 rows in set (0.00 sec)
```

Operator IN dan NOT IN

Inilah contoh IN :

```
SELECT Judul,TahunTerbit
FROM Music
WHERE TahunTerbit  IN
(
    SELECT M.Tahunterbit
        FROM Music AS M JOIN Album AS A USING (MusicID)
        JOIN Studio AS S USING (StudioID)
        WHERE Judul='Menghapus Jejakmu'
)
ORDER BY Judul;
```

Inilah contohnya NOT IN:

```
SELECT Judul,TahunTerbit
FROM Music
WHERE TahunTerbit  NOT IN
(
    SELECT M.Tahunterbit
        FROM Music AS M JOIN Album AS A USING (MusicID)
```

```
        JOIN Studio AS S USING (StudioID)
        WHERE Judul='Menghapus Jejakmu'
)
ORDER BY Judul;
```

Hasilnya adalah

```
+--------------------+-------------+
| Judul              | TahunTerbit |
+--------------------+-------------+
| Diantara Kalian    |        2006 |
| Jangan Menyerah    |        2005 |
| Menghapus Jejakmu  |        2007 |
| Okelah Klo Begitu  |        2007 |
| Saat Kau Jauh      |        2009 |
| Sang Mantan        |        2008 |
| Tak Ada Yang Abadi |        2005 |
+--------------------+-------------+
7 rows in set (0.00 sec)
```

b) Operator EXISTS dan NOT EXISTS

Inilah contoh EXISTS :

```
SELECT MusicID, Judul
FROM Music AS M
WHERE EXISTS
(
    SELECT MusicID
        FROM Album AS A
        WHERE M.MusicID=A.MusicID
)
ORDER BY Judul;
```

Hasilnya adalah

```
+---------+---------------------+
| MusicID | Judul               |
```

```
+---------+--------------------+
|   33003 |  Jangan Menyerah   |
|   77007 |  Menghapus Jejakmu |
|   66006 |  Okelah Klo Begitu |
|   55005 |  Saat Kau Jauh     |
|   44004 |  Sang Mantan       |
|   11001 |  Tak Ada Yang Abadi |
+---------+--------------------+
6 rows in set (0.00 sec)
```

Inilah contoh NOT EXISTS :

```
SELECT MusicID, Judul
FROM Music AS M
WHERE NOT EXISTS
(

    SELECT MusicID
        FROM Album AS A
        WHERE M.MusicID=A.MusicID
)
ORDER BY Judul;
```

Hasilnya adalah

```
+---------+-----------------+
| MusicID | Judul           |
+---------+-----------------+
|   22002 |  Diantara Kalian |
+---------+-----------------+
1 row in set (0.00 sec)
```

IX.1.8 Bekerja dengan Group Data

Terdapat fungsi tambahan bagi user untuk memprmudah perhitungan data yang berkelompok. Fungsi SUM() adalah solusinya. Untuk lebih jelasnya,kita lihat contoh berikut

```
CREATE TABLE Booking
(
        BookingID INT NOT NULL,
        MusicID INT NOT NULL,
        Jumlah INT NOT NULL,
        PRIMARY KEY (BookingID, MusicID),
        FOREIGN KEY (MusicID) REFERENCES Music (MusicID)
)
ENGINE=INNODB;

INSERT INTO Booking
VALUES (400, 11001, 1), (400, 22002, 1), (401, 33003, 2), (401, 44004, 1),
(401, 55005, 3), (402, 66006, 2), (402, 77007, 1),(403, 11001, 1), (401, 22002, 1),
(403, 33003, 2), (402, 44004, 1),
(405, 55005, 3), (403, 66006, 2), (405, 77007, 1);
```

Fungsi untuk menghitung total:

```
SELECT BookingID, SUM(Jumlah) AS Total
FROM Booking
GROUP BY BookingID
HAVING Total>(SELECT AVG(Jumlah) FROM Booking);
```

Hasilnya

```
+-----------+-------+
| BookingID | Total |
+-----------+-------+
|       400 |     2 |
|       401 |     7 |
|       402 |     4 |
|       403 |     5 |
|       405 |     4 |
+-----------+-------+
5 rows in set (0.06 sec)
```

IX.1.9 Menambah subqueris pada statement UPDATE

Sekarang kita akan melihat contoh statement UPDATE dengan Subqueries

```
UPDATE Music
SET Judul='Okelah Klo Begitu', TahunTerbit=2007
WHERE MusicID=
(
        SELECT A.MusicID
        FROM Studio AS S, Album AS A
        WHERE S.StudioID=A.StudioID AND PemilikStudio='Abdi'
);
```

Penjelasan :

Data yang telah di update akan berubah sesuai data yang telah dirubah.

IX.1.10 Menambah subqueris pada statement DELETE

Sekarang kita akan melihat contoh statement DELETE dengan Subqueries

```
DELETE A, M
FROM Album AS A, Music AS M
WHERE A.MusicID=M.MusicID AND A.StudioID=(SELECT StudioID FROM Studio
WHERE PemilikStudio="Risna");
```

IX.1.11 Membuat UNION yang Menggunakan JOIN pada statement SELECT

Penambahan waktu pasti akan terjadi ketika kita mengkombinasikan MULTIPLE tabel. Cara mudah dalam menggunakan operator UNION untuk koneksi pada statement, ditunjukkan oleh syntax berikut:

```
<select statement> UNION <select statement>
```

Dalam implementasinya

```
CREATE TABLE Studio2
(
```

```
        StudioID INT NOT NULL PRIMARY KEY,
        PemilikStudio VARCHAR(20)
);

INSERT INTO Studio2 VALUES
(0101,"Rani"),
(0102,"Mira"),
(0103," Arif "),
(0104,"Kartiwa"),
(0105,"Asti"),
(0106," Tori ");
```

Inilah syntax UNION

```
SELECT PemilikStudio FROM Studio
UNION
SELECT PemilikStudio FROM Studio2;
```

Dan hasilnya sebagai berikut

```
+---------------+
| PemilikStudio |
+---------------+
|  Sony         |
| Abdi          |
| Mutia         |
|  Risna        |
| Audy          |
|  Taufik       |
| Rani          |
| Mira          |
|  Arif         |
| Kartiwa       |
| Asti          |
|  Tori         |
+---------------+
```

12 rows in set (0.00 sec)

Atau dapat juga dikombinasikan dengan syntax berikut ini

```
(SELECT PemilikStudio FROM Studio)
UNION
(SELECT PemilikStudio FROM Studio2)
ORDER BY PemilikStudio;
```

Dan Hasilnya :

```
+---------------+
| PemilikStudio |
+---------------+
| Arif          |
| Risna         |
| Sony          |
| Taufik        |
| Tori          |
| Abdi          |
| Asti          |
| Audy          |
| Kartiwa       |
| Mira          |
| Mutia         |
| Rani          |
+---------------+
```

12 rows in set (0.05 sec)