BAB VII
MENGGUNAKAN OPERATOR

Pada bab-bab sebelumnya, banyak ekspresi-ekspresi yang menggunakan yang mnggunakan fungsi operator. Clausa –clausa yang diabungkan dengan operator merupakan salah satu formula untuk menggunakan perintah dalam MYSQL. Di lain hal fungsi operator pada clausa juga dapat membatasi range dari kondisi yang diciptakan. Untuk lebih jelasnya kita akan membahas lebih dalam pada bab ini.

Di bawah ini merupakan operator-operator yang digunakan dalam SQL

1. **BINARY, COLLATE**
2. **NOT (logical negation), ! (logical negation)**
3. **- (unary minus), ~ (unary bit inversion)**
4. **^ (bitwise exclusive OR comparison)**
5. *** (multiplication), / (division), % (modulo)**
6. **- (subtraction), + (addition)**
7. **<< (bitwise shift left), >> (bitwise shift right)**
8. **& (bitwise AND)**
9. **| (bitwise OR)**
10. **All comparison operators except BETWEEN and NOT BETWEEN**
11. **BETWEEN, NOT BETWEEN**
12. **AND (logical addition), && (logical addition)**
13. **OR (logical OR comparison), | | (logical OR comparison), XOR (logical exclusive OR comparison)**

Penomoran ini juga dimaksudkan untuk mengetahui prioritas mna yang lebih didahulukan jika digunakan pada SQL.

Operator Arithmetic

Operator ini digunakan untuk menghitung 2 argument yang sama. Contohnya seperti dibawah ini

Operator	**Description**
+ (addition)	Adds the two arguments together.
- (subtraction)	Subtracts the second argument from the first argument.

- (unary)	Changes the sign of the argument.
* (multiplication)	Multiplies the two arguments together.
/ (division)	Divides the first argument by the second argument.
% (modulo)	Divides the first argument by the second argument and provides the remainder from that operation.

Kita akan melihat imlplementasinya secara langsung

```
CREATE TABLE TV
(
        TVID SMALLINT NOT NULL PRIMARY KEY,
        StokGudang  SMALLINT NOT NULL,
        Pemesanan SMALLINT NOT NULL,
        BnyakMerk SMALLINT NOT NULL
);
```

Kemudian kita akan mengisi tabelnya

```
INSERT INTO TV VALUES (201, 12 , 5 , 4), (202, 14 , 6 , 3), (203, 9 , 7 , 6);
```

Dan syntax dibawah ini merupakan kombinasi 2 argument yang dikombinasikan oleh operator arithmetic

```
INSERT INTO TV VALUES (204, 12 , StokGudang - 5 , 4);
```

Dan syntx berikut akan menampilkan data dalam tabel, yang dikombinasikan oleh operator arithmetic

```
SELECT TVID, StokGudang+Pemesanan-BnyakMerk AS Persediaan
FROM TV;
```

Dan Hasilnya:

```
+------+------------+
| TVID | Persediaan |
+------+------------+
```

201	13
202	17
203	10
204	15

+------+------------+

4 rows in set (0.00 sec)

Dan contoh berikut menggunakan lebih dari satu operator arithmetic

```
SELECT TVID, StokGudang+Pemesanan-BnyakMerk AS Persediaan,
StokGudang+Pemesanan*2-BnyakMerk AS PesanBnyak
FROM TV;
```

Dan Hasilnya:

+------+------------+------------+

| TVID | Persediaan | PesanBnyak |

+------+------------+------------+

201	13	18
202	17	23
203	10	17
204	15	22

+------+------------+------------+

4 rows in set (0.00 sec)

Operator arithmetic tidak hanya digunakan pada fungsi INSERT , namun dpat pula digunakan pada fungsi UPDATE

```
UPDATE TV
SET Pemesanan=Pemesanan/2;
```

Dan setelah proses tersebut dilakukan maka kita akan melihat tabel yang baru dengan syntx berikut

```
SELECT * FROM TV;
```

Dan hasilnya

```
+------+------------+-----------+-----------+
| TVID | StokGudang | Pemesanan | BnyakMerk |
+------+------------+-----------+-----------+
| 201 |       12 |       3 |       4 |
| 202 |       14 |       3 |       3 |
| 203 |        9 |       4 |       6 |
| 204 |       12 |       4 |       4 |
+------+------------+-----------+-----------+
4 rows in set (0.02 sec)
```

VII.1 Operator Pembanding

Operator pembanding digunakan untuk membandingkan 2 argumen atau lebih yang mendefinisikan nilai true, false, maupun NULL. Untuk lebih lanjutnya, kita akan lihat pada contoh berikut

```
SELECT TVID, StokGudang
FROM TV
WHERE BnyakMerk= 4;
```

Pada kondisi clausa WHERE mengandung satu operator yang mengandung persamaan. Dimana data yang akan ditampilkan yaitu data yang BnyakMerk = 4. MYSQL mendukung beberapa operator pembanding yaitu

Operator	Deskripsi
=	Menghitung nilai TRUE jika kedua argumen bernilai sama,kecuali kedua argumen bernilai NULL
<=>	Menghitung nilai TRUE jika kedua argumen bernilai sama,walaupun kedua argumen bernilai NULL
<>, !=	Menghitung nilai TRUE jika kedua argumen bernilai tidak sama.
<	Menghitung nilai TRUE jika argumen pertama bernilai lebih kecil dari argumen kedua

<=	Menghitung nilai TRUE jika argumen pertama bernilai lebih kecil atau sama dengan argumen kedua
>	Menghitung nilai TRUE jika argumen pertama bernilai lebih besar dari argumen kedua
>=	Menghitung nilai TRUE jika argumen pertama bernilai lebih besar atau sama dengan argumen kedua
IS NULL	Menghitung nilai TRUE jika kedua argumen sama dengan nilai NULL.
IS NOT NULL	Menghitung nilai TRUE jika kedua argumen tidak sama dengan nilai NULL.
BETWEEN	Menghitung nilai TRUE jika nilai dari argumen berada pada range yang telah ditentukan
NOT BETWEEN	Menghitung nilai TRUE jika nilai dari argumen tidak berada pada range yang telah ditentukan
IN	Menghitung nilai TRUE jika nilai dari argumen terdefinisi pada clausa IN
NOT IN	Menghitung nilai TRUE jika nilai dari argumen tidak terdefinisi pada clausa NOT IN
LIKE	Menghitung nilai TRUE jika nilai dari argumen terdefinisi pada clausa LIKE
NOT LIKE	Menghitung nilai TRUE jika nilai dari argumen tidak terdefinisi pada clausa NOT lIKE
REGEXP	Menghitung nilai TRUE jika nilai dari argumen terdefinisi pada clausa REGEXP
NOT REGEXP	Menghitung nilai TRUE jika nilai dari argumen tidak terdefinisi pada clausa NOT REGEXP

Untuk lebih lengkapnya kita akan melihat contoh berikut ini

```
CREATE TABLE Angkot
(
 AngkotID  SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 NamaSupir VARCHAR(50) NOT NULL,
 Jurusan VARCHAR(50) NOT NULL,
 JenisMobil ENUM ('Toyota ','Daihatsu') NOT NULL,
 JumlahMobil INT(2) NOT NULL,
 MaxPenumpang  INT(2) NOT NULL,
 TahunAktif YEAR,
```

```
 DataUpdate TIMESTAMP NOT NULL
);
```

Kemudian kita akan memasukkan datanya

```
INSERT INTO Angkot (NamaSupir, Jurusan, JenisMobil, JumlahMobil,
MaxPenumpang,TahunAktif )
VALUES ( “Adi”,”Cicaheum-Ciroyom”,’Toyota’,4,9,1990),
( “Ari”,”Dago-Kelapa”,’Toyota’,6,7,1995),
( “Andi”,”Cicaheum-Ciroyom”,’Daihatsu’,10,4,1996),
( “Budi”,”Dago-Kelapa”,’Daihatsu’,3,7,1989),
( “Cecep”,”Kelapa-Ledeng”,’Toyota’,9,8,1997),
( “Dika”,”Kalapa-Ledeng”,’Daihatsu’,5,6,1986),
( “Edi”,”Sadangserang-Caringin”,’Toyota’,4,8,1991),
( “Eka”,”Sadangserang-Caringin”,’Daihatsu’,5,9,1988);
```

Data ini menunjukan operator (=) yang argumennya harus sama dengan yang didefinisikan

Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil
FROM Angkot
WHERE AngkotID=5;
```

Dan hasilnya yaitu

```
+-----------+---------------+------------+
| NamaSupir | Jurusan       | JenisMobil |
+-----------+---------------+------------+
| Cecep     | Kelapa-Ledeng | Toyota     |
+-----------+---------------+------------+
1 row in set (0.00 sec)
```

Data ini menunjukan operator (NULL) yang argumennya sama dengan nilai NULL. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil
```

```
FROM Angkot
WHERE AngkotID=NULL;
```

Dan hasilnya yaitu

Empty set (0.00 sec)

Data ini menunjukan operator NULL -Safe(< = >) yang argumennya sama dengan nilai NULL dan data akan ditampilkan. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil
FROM Angkot
WHERE Jurusan <=> NULL
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

Empty set (0.00 sec)

Hal ini menandakan bahwa tidaka da nilai NULL pada Jurusan.

Data ini menunjukan operator NULL yang argumennya sama dengan nilai NULL. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil
FROM Angkot
WHERE Jurusan IS NULL
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

Empty set (0.00 sec)

Hal ini menandakan bahwa tidaka da nilai NULL pada Jurusan.

Data ini menunjukan operator NOT NULL yang argumennya sama dengan nilai tidak NULL. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil
FROM Angkot
```

```
WHERE Jurusan IS NOT  NULL
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+----------+----------------------+------------+
| NamaSupir | Jurusan              | JenisMobil |
+----------+----------------------+------------+
| Adi      | Cicaheum-Ciroyom     | Toyota     |
| Andi     | Cicaheum-Ciroyom     | Daihatsu   |
| Ari      | Dago-Kelapa          | Toyota     |
| Budi     | Dago-Kelapa          | Daihatsu   |
| Cecep    | Kelapa-Ledeng        | Toyota     |
| Dika     | Kalapa-Ledeng        | Daihatsu   |
| Edi      | Sadangserang-Caringin | Toyota    |
| Eka      | Sadangserang-Caringin | Daihatsu  |
+----------+----------------------+------------+
8 rows in set (0.00 sec)
```

Data ini menunjukan operator lebih besar (>) yang argumen pertama lebih besar dari argumen kedua. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE JumlahMobil > 5
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+----------+-----------------+------------+-------------+
| NamaSupir | Jurusan         | JenisMobil | JumlahMobil |
+----------+-----------------+------------+-------------+
| Andi     | Cicaheum-Ciroyom | Daihatsu   |          10 |
| Ari      | Dago-Kelapa      | Toyota     |           6 |
| Cecep    | Kelapa-Ledeng    | Toyota     |           9 |
+----------+-----------------+------------+-------------+
```

3 rows in set (0.00 sec)

Data ini menunjukan operator lebih besar (<) yang argumen pertama lebih kecil dari argumen kedua. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE JumlahMobil < 5
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+-----------+----------------------+------------+-------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+-----------+----------------------+------------+-------------+
| Adi       | Cicaheum-Ciroyom     | Toyota     |           4 |
| Budi      | Dago-Kelapa          | Daihatsu   |           3 |
| Edi       | Sadangserang-Caringin | Toyota     |           4 |
+-----------+----------------------+------------+-------------+
```

3 rows in set (0.00 sec)

Data ini menunjukan operator IN yang argumennya didasarkan pada definisi yang dimaksudkan. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE Jurusan IN ("Sadangserang-Caringin","Kelapa-Ledeng")
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+-----------+----------------------+------------+-------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+-----------+----------------------+------------+-------------+
| Cecep     | Kelapa-Ledeng        | Toyota     |           9 |
| Edi       | Sadangserang-Caringin | Toyota     |           4 |
```

```
| Eka      | Sadangserang-Caringin | Daihatsu   |          5 |
+----------+-----------------------+------------+------------+
3 rows in set (0.00 sec)
```

Data ini menunjukan operator NOT BETWEEN yang argumennya didasarkan range pada definisi yang dimaksudkan. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE (JumlahMobil + MaxPenumpang) NOT BETWEEN 10 AND 15
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+----------+--------------+------------+------------+
| NamaSupir | Jurusan      | JenisMobil | JumlahMobil |
+----------+--------------+------------+------------+
| Cecep    | Kelapa-Ledeng | Toyota     |          9 |
+----------+--------------+------------+------------+
1 row in set (0.00 sec)
```

Data ini menunjukan operator LIKE yang argumennya didasarkan pada definisi yang mendekati nama yang dimaksudkan. Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE NamaSUpir LIKE “%A%”
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+----------+----------------------+------------+------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+----------+----------------------+------------+------------+
| Adi      | Cicaheum-Ciroyom     | Toyota     |          4 |
```

```
| Andi     | Cicaheum-Ciroyom     | Daihatsu   |         10 |
| Ari      | Dago-Kelapa          | Toyota     |          6 |
| Dika     | Kalapa-Ledeng        | Daihatsu   |          5 |
| Eka      | Sadangserang-Caringin | Daihatsu   |          5 |
+----------+----------------------+------------+------------+
5 rows in set (0.00 sec)
```

Data ini menunjukan operator REGEXP yang argumennya didasarkan pada definisi range huruf . Kemudian kita akan menjalankan perintah berikut:

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE Jurusan REGEXP "^[ad]"
ORDER BY NamaSupir;
```

Dan hasilnya yaitu

```
+----------+-------------+------------+-------------+
| NamaSupir | Jurusan     | JenisMobil | JumlahMobil |
+----------+-------------+------------+-------------+
| Ari      | Dago-Kelapa | Toyota     |          6 |
| Budi     | Dago-Kelapa | Daihatsu   |          3 |
+----------+-------------+------------+-------------+
2 rows in set (0.00 sec)
```

VII.2 OPERATOR LOGIKA

Operator logika berguna untuk mengetest validitas satu atau lebih ekspresi. Operator-operatornya yaitu

Operator	DESKRIPSI
AND	Menghitung nilai true jika kedua argumen bernilai true
OR	Menghitung nilai true jika salah satu argumen bernilai true
XOR	Menghitung nilai true jika salah satu argumen bernilai true
NOT	Menghitung nilai true jika kedua argumen bernilai false

Syntax dibawah ini menggunakan operator logika AND

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE Jurusan=”Kelapa-Ledeng” AND JenisMobil=”Toyota”
ORDER BY NamaSupir;
```

Dan Hasilnya

```
+----------+--------------+------------+-------------+
| NamaSupir | Jurusan      | JenisMobil | JumlahMobil |
+----------+--------------+------------+-------------+
| Cecep    | Kelapa-Ledeng | Toyota     |         9 |
+----------+--------------+------------+-------------+
1 row in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator logika OR

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE MaxPenumpang > 9 OR JenisMobil=”Toyota”
ORDER BY NamaSupir;
```

Dan Hasilnya

```
+----------+----------------------+------------+-------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+----------+----------------------+------------+-------------+
| Adi      | Cicaheum-Ciroyom     | Toyota     |         4 |
| Ari      | Dago-Kelapa          | Toyota     |         6 |
| Cecep    | Kelapa-Ledeng        | Toyota     |         9 |
| Edi      | Sadangserang-Caringin | Toyota     |         4 |
+----------+----------------------+------------+-------------+
4 rows in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator logika XOR

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE MaxPenumpang > 9 XOR JenisMobil=”Daihatsu”
ORDER BY NamaSupir;
```

Dan Hasilnya

```
+----------+----------------------+------------+-------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+----------+----------------------+------------+-------------+
| Andi     | Cicaheum-Ciroyom     | Daihatsu   |          10 |
| Budi     | Dago-Kelapa          | Daihatsu   |           3 |
| Dika     | Kalapa-Ledeng        | Daihatsu   |           5 |
| Eka      | Sadangserang-Caringin | Daihatsu   |           5 |
+----------+----------------------+------------+-------------+
4 rows in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator logika AND dan OR

```
SELECT NamaSupir, Jurusan, JenisMobil, JumlahMobil
FROM Angkot
WHERE MaxPenumpang > 5 AND (JenisMobil=”Daihatsu” OR Jurusan=”Dago-Kelapa”)
ORDER BY NamaSupir;
```

Dan Hasilnya

```
+----------+----------------------+------------+-------------+
| NamaSupir | Jurusan              | JenisMobil | JumlahMobil |
+----------+----------------------+------------+-------------+
| Ari      | Dago-Kelapa          | Toyota     |           6 |
| Budi     | Dago-Kelapa          | Daihatsu   |           3 |
| Dika     | Kalapa-Ledeng        | Daihatsu   |           5 |
| Eka      | Sadangserang-Caringin | Daihatsu   |           5 |
```

```
+-----------+----------------------+------------+-------------+
```

4 rows in set (0.00 sec)

VII.3 OPERATOR BITWISE

Operator ini memiliki kegunaan yaitu untuk membandingkan dan memodifikasi nilai bit dengan nilai numeric yang tersimpan dalam database.

Operator – operator tersebut yaitu &, | , ^ , ~ , << , dan >>. Untuk lebih jlasnya akan kegunaan operator –operator tersebut, maka kita akan melihat contonhnya.

```
CREATE TABLE Sepatu
(
SepatuID SMALLINT NOT NULL PRIMARY KEY,
Ukuran TINYINT UNSIGNED NOT NULL
);
```

```
INSERT INTO Sepatu VALUES (201, 47), (202, 35), (203, 43);
```

Syntax dibawah ini menggunakan operator bitwise (|)

```
UPDATE Sepatu
SET Ukuran=Ukuran | 1
WHERE SepatuID=201;
```

Kemudian jalankan syntax berikut

```
SELECT * FROM Sepatu;
```

Dan hasil akhirnya

```
+----------+--------+
| SepatuID | Ukuran |
+----------+--------+
```

```
|      1 |    47 |
|      2 |    35 |
|      3 |    43 |
+----------+--------+
3 rows in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator bitwise (|)

```
UPDATE Sepatu
SET Ukuran=Ukuran << 1;
```

Dan hasilnya dengan menggunakan perintah SeLECT yaitu

```
+----------+--------+
| SepatuID | Ukuran |
+----------+--------+
|      1 |    94 |
|      2 |    70 |
|      3 |    86 |
+----------+--------+
6 rows in set (0.00 sec)
```

VII.4 Operator SORT

Operator ini memiliki kelebihan dalam membandingkan nilai secara kolom. Untuk lebih jelasnya kita akan melihat contoh berikut.

```
CREATE TABLE Hp
(
HpID SMALLINT NOT NULL PRIMARY KEY,
MerkHp VARCHAR(15) NOT NULL
);

INSERT INTO Hp
```

```
VALUES (101, 'Nokia'), (102, 'Siemens'), (103, 'SiEmens'), (104, 'SiemeNs'), (105, 'NoKia'),
(106, 'NokIa'), (107, 'NOkia'), (108, 'SiemenS'), (109, 'NokiA'), (110, 'nokia');

SELECT * FROM Hp
WHERE MerkHp='Nokia';
```

Dan hasilnya

```
+------+--------+
| HpID | MerkHp |
+------+--------+
|  101 | Nokia  |
|  105 | NoKia  |
|  106 | NokIa  |
|  107 | NOkia  |
|  109 | NokiA  |
|  110 | nokia  |
+------+--------+
6 rows in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator BINARY. Syntax ini akan menampilkan data sesuai dengan karakter baik upper maupun lower.

```
SELECT * FROM Hp
WHERE MerkHp = BINARY 'Siemens';
```

```
+------+---------+
| HpID | MerkHp  |
+------+---------+
|  102 | Siemens |
+------+---------+
1 row in set (0.00 sec)
```

Syntax dibawah ini menggunakan operator COLLATE. Syntax ini akan menampilkan data tanpa melihat karakter baik upper maupun lower.

```
SELECT * FROM Hp
WHERE MerkHp COLLATE latin1_german2_ci = 'Nokia';
```

```
+------+--------+
| HpID | MerkHp |
+------+--------+
|  101 | Nokia  |
|  105 | NoKia  |
|  106 | NokIa  |
|  107 | NOkia  |
|  109 | NokiA  |
|  110 | nokia  |
+------+--------+
6 rows in set (0.00 sec)
```