BAB VIII
MENGGUNAKAN FUNGSI DALAM STATEMENT SQL

Fungsi merupakan suatu ekspresi yang dapat dipanggil sesuai dengan kegunaannya. Dalam setiap function minimal terdiri dari satu argument. Isnya argumennya dapat berupa manipulasi data, perhitungan data, dan lainnya.

Perbandingan dan Konversi Data

MYSQL menyediakan 3 tipe fungsi dalam membandingkan dan konversi data. Ketiga tipe tersebut yaitu Comparison function, control flow function, dan cast function.

Fungsi Perbandingan

Perbandingan dalam fungsi ini memiliki kesamaan hasil jika kita menggunakan operator biasa. Namun fungsi perbandingan ini telah disediakan dalam MYSQL untuk memudahkan kita untuk mengakses sua tabel yang akan dibandingkan.

VIII.1 Fungsi GREATEST() dan LEAST()

Fungsi ini dapat membandingkan 2 atau lebih variabel nilai tergantung data yang akan dibandingkan. Syntax secara umumnya adalah sebagai berikut

```
GREATEST(<value1>, <value2> [{, <value>}...])
LEAST(<value1>, <value2> [{, <value>}...])
```

Dan berikut merupakan implementasi kedua fungsi tersebut

```
SELECT GREATEST( 2 . 4 . 8 . 9 . 3 . 6 .1 );
```

Hasil yang akan muncul pada statement diatas yaitu 9, karena fungsi diatas berfungsi untuk mencari nilai tertingggi dari data yang ada.

```
SELECT LEAST( 2 . 4 . 8 . 9 . 3 . 6  . 1 );
```

Hasil yang akan muncul pada statement diatas yaitu 1, karena fungsi diatas berfungsi untuk mencari nilai terendah dari data yang ada.

VIII.2 Fungsi COALESCE() dan ISNULL()

Fungsi COALESCE() mengembalikan nilai nilai dari argumen – argumen yang tidak NULL. Jika semua nilai NULL maka Nilai NULL yang akan dikembalikan. Syntax secara umum yaitu

COALESCE(<value> [{, <value>}...])

ISNULL(<expression>)

Dan berikut merupakan implementasi kedua fungsi tersebut

SELECT COALESCE(NULL, 2, NULL, 3);

Hasil yang akan muncul pada statement diatas yaitu 2, karena fungsi diatas berfungsi untuk mencari nilai selain nilai NULL yang terdefinisis pertama kali dari data yang ada.

SELECT ISNULL(1*NULL);

Hasil yang akan muncul pada statement diatas yaitu 1, karena nilai yang didalam kurung bernilai NULL(0) dan jika mendapat fungsi ISNULL maka nilai tersebut akan di negaikan / atu dibalikkan menjadi 1.

VIII.3 Fungsi INTERVAL() dan STRCMP()

Fungsi INTERVAL () berguna untuk membandingkan nilai integer yang pertama dengan nilai integer selanjutnya. Syntax secara umumnya

INTERVAL(<integer1>, <integer2> [{, <integer>}...])

Dan implementasinya

SELECT INTERVAL(6, -2, 0, 4, 7, 10, 12);

Hasilnya nilai 6 akan berpindah posisi pada urutan keempat sesua dengan tingkatannya (Ascending).

Fungsi STRCMP () berguna untuk membandingkan nilai string yang pertama dengan nilai string selanjutnya. Syntax secara umumnya

Dan implementasinya

SELECT STRCMP(“ tall,taller ”);

Hasilnya nlai yang akan keluar akan bernilai a-1

VIII.4 Control Flow Function

VIII.4.1 IF() Function

Fungsi if() membandingkan 3 ekspresi seperti yang ditunjukan syntax dibawah ini:

IF(<expression1>, <expression2>, <expression3)

Pejelsannya jika <expression1> sma dengan true ,mka nilai yang keluar adalah <expression2>, atau jika tidak nilai yang keluar yaitu <expression3>. Contohnya seperti dibawah ini:

```
SELECT IF(10>20, 'expression correct', 'expression incorrect');
```

Karena <expression1> bernilai false maka <expression3> yang akan muncul

VIII.4.2 Fungsi IFNULL() dan NULLIF()

Fungsi IFNULL() mengembalikan nilai berdasarkan spesifik ekspresi yang bernilai NULL. Dan syntaxnya secara umum yaitu

```
IFNULL(<expression1>, <expression2>)
```

Inilah implementasinya

```
SELECT IFNULL(5*NULL, 'expression salah');
```

Hasilnya karena <expression1> bernilai NULL maka nilai tersebut tidak akan ditampilkan. Berbeda dengan <expression2>, nilai tersebut karena berupa string maka akan ditampilkan.

Fungsi INULLIF() mengembalikan nilai NULL jika <expression1> sama dengan <expression2>, dengan kata lain akan menghasilkan <expression1>. Dan syntaxnya secara umum yaitu

```
NULLIF(<expression1>, <expression2>)
```

Inilah implementasinya

```
SELECT NULLIF(10*20, 20*10);
```

Karena <expression1> bernilai sama dengan <expression2>, maka nilai NULL yang akan ditampilkan.

VIII.4.3 Fungsi Case()

Fungsi ini lebih rumit dari fungsi sebelumnya. Fungsi ini menyediakan ekspresi yang lebih flexibel dalam megaplikasikan beberapa kondisi sekaligus. Fungsi ini memiliki dua format dalam pengaplikasiannnya

Yang pertama adalah

```
CASE WHEN <expression> THEN <result>
[{WHEN <expression> THEN <result>}...]
[ELSE <result>]
END
```

Dengan contohnya seperti berikut

```
SELECT CASE WHEN 15*4=60 THEN “60 benar”
WHEN 12*2=32 THEN “32 benar”
ELSE “Seharusnya 12*2=24”
END;
```

Nilai yang akan keluar yaitu “Seharusnya 12*2=24” karena kondisi CASE yang ada bernilai true, sedangkan yang kondisi “ 12*2=32 ” salah.

Syntax CASE yang kedua

```
CASE <expression>
WHEN <value> THEN <result>
[{WHEN <value> THEN <result>}...]
[ELSE <result>]
END
```

Dan implementasinya yaitu

```
SELECT CASE 15*1
WHEN 15 THEN “15 benar”
WHEN 30 THEN “30 benar”
WHEN 45 THEN “45 benar”
END;
```

Hasil yang akan keluar yaitu 15. Syntax yang kedua ini memiliki hasil yang sama dengan syntax sebelumnya. Hanya saja syntaxnya berbeda.

VIII.4.4 Fungsi CAST

Fungsi ini mengkonversi nilai ke nilai spesifik data yang baru. Inilah syntax secara umum

CAST(<expression> AS <type>)

Tipe-tipe yang disediakan pada syntax ini yaitu

- ❑ BINARY
- ❑ CHAR
- ❑ DATE
- ❑ DATETIME
- ❑ SIGNED [INTEGER]
- ❑ TIME
- ❑ UNSIGNED [INTEGER]

Untuk lebih memahaminya maka kita akan melihat pada contoh berikut

SELECT CAST(20091228 AS DATE);

Hasil yang akan ditampilkan adalah format data DATE yang menjadi 2009-12-28.

Fungsi CONVERT() juga mengubah tanggal yang hampir sama dengan fungsi CAST(), hanya saja formatnya sedikit berbeda

CONVERT(<expression>, <type>)

Seperti inilah cara penerapannya

SELECT CONVERT(20091228, DATE);

Hasil dari CONVERT() dan CAST() akan sama seperti dibawah ini

```
+-------------------------+
| CONVERT(20091228, DATE) |
+-------------------------+
| 2009-12-28              |
+-------------------------+
1 row in set (0.06 sec)
```

Untuk fungsi CONVERT() hanya boleh didefinisikan oleh nilai numerik. Adapun syntax lainnya yang dapat digunakan selain nilai numerik yaitu

CONVERT(<expression> USING <character set>)

Dan contoh dibawah ini mengubah bentuk string yang didefinisikan menjadi karakter latin2

SELECT CONVERT('cats and dogs' USING latin2);

VIII.5 Memanage Beberapa tipe data

VIII.5.1 Fungsi String

Fungsi string ini dapat memanipulasi dan mengextract nilai string. Inilah beberapa fungsinya.

VIII.5.1.1 Fungsi ASCII() dan ORD()

Fungsi ASCII dapat mengidentifikasi nilai numerik pada setiap karakter pada kalimat. Syntax nya yaitu

ASCII(<string>)

Implementasinya yaitu

SELECT ASCII("baju");

Maka nilai numerik yang dihasilkan pada kalimat baju yaitu;

```
+---------------+
| ASCII("baju") |
+---------------+
|          98 |
+---------------+
1 row in set (0.03 sec)
```

Fungsi ASCII hanya berfungsi untuk single –byte karakter . Untuk multy-byte karakter harus menggunakan fungsi ORD() yang syntaxnya adalah sebagai berikut

ORD(<string>)

Contohnya adalah sebagai berikut

SELECT ORD(37);

VIII.5.1.2 Fungsi CHAR_LENGTH(), CHARACTER_LENGTH(), and LENGTH()

Fungsi ini berguna untuk menghitung banyaknya karakter dalam suatu kalimat. Dan hasil yang akan ditampilkan pada ketiga fungsi tersebut adalah sama. Secara syntax

CHAR_LENGTH(<string>)

LENGTH(<string>)

Dan implementasinya

SELECT CHAR_LENGTH('jangkrik');

Banyaknya huruf yang ada yaitu 8 karakter.

SELECT LENGTH('jerapah');

Banyaknya huruf yang ada yaitu 7 karakter.

VIII.5.1.3 Fungsi CHARSET() dan COLLATION()

Kedua fungsi tersebut memiliki fungsi untuk mengetahui karakter set yang sedang digunakan. Secara syntax

CHARSET(<string>)

COLLATION(<string>)

Dan implementasinya

SELECT CHARSET('kucing');

Hasilnya:

```
+-------------------+
| CHARSET('kucing') |
+-------------------+
| latin1            |
+-------------------+
1 row in set (0.00 sec)
```

SELECT COLLATION('semut');

Hasilnya:

```
+--------------------+
| COLLATION('semut') |
+--------------------+
| latin1_swedish_ci  |
+--------------------+
1 row in set (0.00 sec)
```

VIII.5.1.4 Fungsi CONCAT() dan CONCAT_WS()

Kedua fungsi tersebut memiliki fungsi untuk mengabungkan beberapa kata menjadi satu kalimat. Secara syntax

CONCAT(<string1>, <string2> [{, <string>}...])

CONCAT_WS(<separator>, <string1>, <string2> [{, <string>}...])

Dan implementasinya:

SELECT CONCAT('kakak', ' ', 'dan', ' ', 'adik');

Hasilnya:

```
+----------------------------------------+
| CONCAT('kakak', ' ', 'dan', ' ', 'adik') |
+----------------------------------------+
```

```
| kakak dan adik                        |
+---------------------------------------+
1 row in set (0.03 sec)
```

SELECT CONCAT_WS(‘ ‘, ‘paman’, ‘atau’, ‘bibi’);

Hasilnya:

```
+---------------------------------------+
| CONCAT_WS(' ', 'paman', 'atau', 'bibi') |
+---------------------------------------+
| paman atau bibi                       |
+---------------------------------------+
1 row in set (0.00 sec)
```

VIII.5.1.5 Fungsi INSTR() dan LOCATE()

Kedua fungsi tersebut memiliki fungsi untuk mengabungkan beberapa kata menjadi satu kalimat. Secara syntax

INSTR(<string>, <substring>)

LOCATE(<substring>, <string>)

LOCATE(<substring>, <string>, <position>)

Dan implementasinya:

SELECT INSTR(‘Ayah dan Ibu’, ‘ibu’);

Hasilnya:

```
+-----------------------------+
| INSTR('Ayah dan Ibu', 'ibu') |
+-----------------------------+
|                          10 |
+-----------------------------+
1 row in set (0.00 sec)
```

SELECT LOCATE(‘kakak’, ‘adik dan kakak’);

Hasilnya:

```
+----------------------------------+
| LOCATE('kakak', 'adik dan kakak') |
+----------------------------------+
|                               10 |
+----------------------------------+
1 row in set (0.00 sec)
```

SELECT LOCATE(‘paman’, ‘paman dan bibi serta paman lainnya’, 15);

Hasilnya:

```
+-----------------------------------------------------------+
| LOCATE('paman', 'paman dan bibi serta paman lainnya', 15) |
```

```
+-----------------------------------------------------------+
|                                                        22 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

VIII.5.1.6 Fungsi LCASE(), LOWER(), UCASE(), and UPPER()

Keempat fungsi tersebut memiliki fungsi untuk membesarkan atau mengecilkan kata-kata yang terdapat dalam MYSQL. Syntaxnya sebagai berikut

LOWER(<string>)

UPPER(<string>)

Dan implementasinya:

SELECT LOWER('LAMPU');

Hasilnya:

```
+----------------+
| LOWER('LAMPU') |
+----------------+
| lampu          |
+----------------+
1 row in set (0.00 sec)
```

SELECT UPPER('monitor');

Hasilnya:

```
+------------------+
| UPPER('monitor') |
+------------------+
| MONITOR          |
+------------------+
1 row in set (0.00 sec)
```

VIII.5.1.7 Fungsi LEFT() and RIGHT()

Fungsi ini memiliki kegunaan untuk mengetahui kata-kata sebanyak jumlah kata-kata yang ingin diketahui baik dimulai dari kanan maupun kiri. Maka syntaxnya

LEFT(<string>, <length>)

RIGHT(<string>, <length>)

Dan implementasinya:

SELECT LEFT('kingkong', 4);

Hasilnya:

```
+---------------------+
| LEFT('kingkong', 4) |
+---------------------+
| king                |
+---------------------+
1 row in set (0.00 sec)
```

SELECT RIGHT('pemandangan', 4);

Hasilnya:

```
+-------------------------+
| RIGHT('pemandangan', 4) |
+-------------------------+
| ngan                    |
+-------------------------+
1 row in set (0.00 sec)
```

VIII.5.1.8 Fungsi REPEAT() and REVERSE()

Fungsi **REPEAT()** memiliki kegunaan untuk mengulangi kata-kata sebanyak jumlah pengulangan yang diinginkan. Sedangkan fungsi **REVERSE()** berfungsi untuk membalikkan susunan kata-kata.Secara umum syntaxnya

REPEAT(<string>, <count>)

REVERSE(<string>)

Implementasinya

SELECT REPEAT('kaku', 3);

Hasilnya:

```
+--------------------+
| REPEAT('kaku', 3) |
+--------------------+
| kakukakukaku       |
+--------------------+
1 row in set (0.00 sec)
```

SELECT REVERSE('buku');

Hasilnya:

```
+------------------+
| REVERSE('buku') |
+------------------+
| ukub            |
+------------------+
1 row in set (0.00 sec)
```

VIII.5.1.9 Fungsi SUBSTRING()

Fungsi ini berfungsi untuk menentukan kata-kata yang ingin diketahui. Untuk lebih jelasnya maka kita akan melihat contohnya. Dan inilah syntax secara umumnya.

SUBSTRING(<string>FROM <position>)

SUBSTRING(<string>, <position>, <length>)

Implementasinya:

SELECT SUBSTRING('mobil polisi', 10);

Hasilnya:

```
+------------------------------+
| SUBSTRING('mobil polisi', 10) |
+------------------------------+
| isi                          |
+------------------------------+
1 row in set (0.00 sec)
```

SELECT SUBSTRING('mobil dan angkot dan kendaraaan lainnya', 10, 4);

Hasilnya:

```
+-----------------------------------------------------------+
| SUBSTRING('mobil dan angkot dan kendaraaan lainnya', 10, 4) |
+-----------------------------------------------------------+
| ang                                                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

VIII.5.2 Fungsi Numeric

Fungsi ini merupakan suatu fungsi yang menghitung penghitungn nilai numeric. Dibawah merupakan fungsi-fungsi numeric yang disediakan oleh MYSQL.

VIII.5.2.1 Fungsi CEIL(), CEILING(), and FLOOR()

CEIL() dan CEILING() memiliki kesamaan fungsi yang membulatkan ke atas nilai float. Sedangkan **FLOOR()** memiliki fungsi yang membulatkan ke bawah nilai float. Inilah syntax secara umumnya

CEILING(<number>)

FLOOR(<number>)

Implementasinya:

SELECT CEILING(5.78);

Hasilnya:

```
+---------------+
| CEILING(5.78) |
+---------------+
|             6 |
+---------------+
1 row in set (0.00 sec)
```

SELECT FLOOR(8.12);

Hasilnya:

```
+-------------+
| FLOOR(8.12) |
+-------------+
|           8 |
+-------------+
1 row in set (0.00 sec)
```

VIII.5.2.2 Fungsi COT()

Fungsi ini memiliki kegunaan untuk menghitung nilai cotangen. Secara umum syntaxnya sebagai berikut

COT(<number>)

Implementasinya:

SELECT COT(90);

Hasilnya:

```
+--------------------+
| COT(90)            |
+--------------------+
| -0.501202783380153 |
```

VIII.5.2.3 Fungsi MOD()

Fungsi ini memiliki kegunaan untuk mengetahui apakah hasil bagi kedua nilai memiliki nilai sisa bagi setelah proses pembagian dilakukan. Secara umum syntaxnya

MOD(number1>, <number2>)

Implementasinya:

SELECT MOD(30, 2);

Hasilnya:

```
+------------+
| MOD(30, 2) |
+------------+
|          0 |
+------------+
```

SELECT MOD(31,2);

Hasilnya:

```
+------------+
| MOD(31, 2) |
+------------+
|          1 |
+------------+
```

VIII.5.2.4 Fungsi PI()

Fungsi ini berguna untuk mengembalikan nilai PI. Secara umum syntaxnya

PI()

Implementasinya

SELECT PI();

Hasilnya:

```
+----------+
| pi()     |
+----------+
| 3.141593 |
+----------+
```

VIII.5.2.5 Fungsi POW() dan POWER()

Fungsi ini berguna untuk memangkatkan suatu nilai. Secara umum syntaxnya

POW(<number>, <power>)

Implementasinya

```
SELECT POW(7, 6);
```

Hasilnya:

```
+-----------+
| POW(7, 6) |
+-----------+
|    117649 |
+-----------+
1 row in set (0.00 sec)
```

VIII.5.2.6 Fungsi ROUND() and TRUNCATE()

Fungsi **ROUND()** bertujuan untuk membulatkan nilai ke atas sebanyak nilai di belakang koma. Sedangkan fungsi **TRUNCATE()** bertujuan untuk membulatkan nilai ke bawah sebanyak nilai di belakang koma. Secar umum syntaxnya

ROUND(<number> [, <decimal>])

TRUNCATE(<number>, <decimal>)

Implementasinya

```
SELECT ROUND(2.68965, 2);
```

Hasilnya:

```
+-------------------+
| ROUND(2.68965, 2) |
+-------------------+
|              2.69 |
+-------------------+
1 row in set (0.00 sec)
```

```
SELECT TRUNCATE(5.23678, 2);
```

Hasilnya:

```
+----------------------+
| TRUNCATE(5.23678, 2) |
+----------------------+
|                 5.23 |
+----------------------+
1 row in set (0.00 sec)
```

VIII.5.2.7 Fungsi SQRT()

Fungsi ini berguna untuk mengakarkan suatu nilai. Secara umum syntaxnya

SQRT(<number>)

Implementasinya

SELECT SQRT(81);

Hasilnya:

```
+----------+
| SQRT(81) |
+----------+
|        9 |
+----------+
1 row in set (0.00 sec)
```

VIII.5.3 Date/Time

VIII.5.3.1 Fungsi ADDDATE(), DATE_ADD(), SUBDATE(), DATE_SUB() dan EXTRACT()

Fungsi ADDDATE() dan DATE_ADD() memiliki kesamaan dalam hasil eksekusinya. Secara umum syntax yang dimiliki fungsi-fungsi tersebut yaitu

ADDDATE(<date>, INTERVAL <expression> <type>)

ADDDATE(<date>, <days>)

SUBDATE(<date>, INTERVAL <expression> <type>)

SUBDATE(<date>, <days>)

EXTRACT(<type> FROM <date>)

Untuk lebih jelasnya kita akan melihat contoh-contohnya

SELECT ADDDATE('2009-11-29 15:49:59', INTERVAL '10:20' HOUR_MINUTE);

Hasilnya:

SELECT ADDDATE('2009-12-20 22:45:23', 21);

Hasilnya:

SELECT SUBDATE('2004-10-31 23:59:59', INTERVAL '12:10' HOUR_MINUTE);

Hasilnya:

SELECT SUBDATE('2004-12-31 23:59:59', 31);

Hasilnya:

SELECT EXTRACT(YEAR_MONTH FROM '2004-12-31 23:59:59');

Hasilnya:

VIII.5.3.2 Fungsi CURDATE(), CURRENT_DATE(), CURTIME(), CURRENT_TIME(), CURRENT_TIMESTAMP(), dan NOW()

Fungsi ini berguna untuk melihat tanggal atau dan waktu sekarang, sesuai waktu yang ada pada komputer. Secara syntax yaitu

CURDATE()

CURTIME()

NOW()

Dan implementasinya:

SELECT CURDATE();

Hasilnya

```
+------------+
| CURDATE()  |
+------------+
| 2010-01-04 |
+------------+
1 row in set (0.00 sec)
```

SELECT CURTIME();

Hasilnya

```
+-----------+
| CURTIME() |
+-----------+
| 12:26:30  |
+-----------+
1 row in set (0.00 sec)
```

SELECT NOW();

Hasilnya

```
+---------------------+
| NOW()               |
+---------------------+
| 2010-01-04 12:26:30 |
+---------------------+
1 row in set (0.00 sec)
```

VIII.5.3.3 Fungsi DATE(), MONTH(), MONTHNAME(), and YEAR()

Fungsi ini berguna untuk melihat tanggal, bulan, nama bulan dan tahun, sesuai waktu yang didefinisikan user. Secara syntax yaitu

DATE(<date>)

MONTH(<date>)

MONTHNAME(<date>)

YEAR(<date>)

Dan implementasinya:

SELECT DATE('2010-04-31 23:59:59');

Hasilnya

```
+-----------------------------+
| DATE('2010-04-31 23:59:59') |
+-----------------------------+
| NULL                        |
+-----------------------------+
1 row in set, 1 warning (0.00 sec)
```

SELECT MONTH('2010-03-31 23:59:59');

Hasilnya

```
+-----------------------------+
| MONTH('2010-03-31 23:59:59') |
+-----------------------------+
|                           3 |
+-----------------------------+
1 row in set (0.03 sec)
```

SELECT MONTHNAME('2010-10-31 23:59:59');

Hasilnya

```
+---------------------------------+
| MONTHNAME('2010-10-31 23:59:59') |
+---------------------------------+
| October                         |
+---------------------------------+
1 row in set (0.03 sec)
```

SELECT YEAR('2010-08-31 23:59:59');

Hasilnya

```
+-----------------------------+
| YEAR('2010-08-31 23:59:59') |
+-----------------------------+
|                        2010 |
+-----------------------------+
1 row in set (0.03 sec)
```

VIII.5.3.4 Fungsi DATEDIFF() and TIMEDIFF()

Fungsi ini berguna untuk melihat selisih waktu baik dari segi tanggal maupun waktu sesuai waktu yang didefinisikan user. Secara syntax yaitu

DATEDIFF(<date>, <date>)

TIMEDIFF(<time>, <time>)

Dan implementasinya:

SELECT DATEDIFF('2010-12-31 23:59:59', '2008-12-31 23:59:59');

Hasilnya

```
+-----------------------------------------------------------+
| DATEDIFF('2010-12-31 23:59:59', '2008-12-31 23:59:59') |
+-----------------------------------------------------------+
|                                                       730 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

SELECT TIMEDIFF('2010-01-3 23:59:59', '2010-01-1 23:59:59');

Hasilnya

```
+-----------------------------------------------------------+
| TIMEDIFF('2010-01-3 23:59:59', '2010-01-1 23:59:59') |
+-----------------------------------------------------------+
| 48:00:00                                                  |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

VIII.5.3.5 Fungsi DAY(), DAYOFMONTH(), DAYNAME(), DAYOFWEEK(), and DAYOFYEAR()

Fungsi ini berguna untuk melihat hari, nama bulan, nama hari, ahri dalam seminggu dan hari dalam tahun sesuai waktu yang didefinisikan user. Secara syntax yaitu

DAY(<date>)

DAYNAME(<date>)

DAYOFWEEK(<date>)

DAYOFYEAR(<date>)

Dan implementasinya:

SELECT DAY('2010-12-31 23:59:59');

Hasilnya

```
+---------------------------+
| DAY('2010-12-31 23:59:59') |
+---------------------------+
|                        31 |
+---------------------------+
1 row in set (0.00 sec)
```

SELECT DAYNAME('2010-05-31 23:59:59');

Hasilnya

```
+-------------------------------+
| DAYNAME('2010-05-31 23:59:59') |
+-------------------------------+
| Monday                        |
+-------------------------------+
1 row in set (0.00 sec)
```

SELECT DAYOFWEEK('2010-08-31 23:59:59');

Hasilnya

```
+---------------------------------+
| DAYOFWEEK('2010-08-31 23:59:59') |
+---------------------------------+
|                               3 |
+---------------------------------+
1 row in set (0.00 sec)
```

SELECT DAYOFYEAR('2010-01-31 23:59:59');

Hasilnya

```
+---------------------------------+
| DAYOFYEAR('2010-01-31 23:59:59') |
+---------------------------------+
|                              31 |
+---------------------------------+
1 row in set (0.01 sec)
```

VIII.5.3.6 Fungsi SECOND(), MINUTE(), HOUR(), and TIME()

Fungsi ini berguna untuk melihat waktu baik detik, menit, jam dan time. sesuai waktu yang didefinisikan user. Secara syntax yaitu

SECOND(<time>)

MINUTE(<time>)

HOUR(<time>)

TIME(<time>)

Dan implementasinya:

SELECT SECOND('2008-12-31 23:59:30');

Hasilnya

```
+-------------------------------+
| SECOND('2008-12-31 23:59:30') |
+-------------------------------+
|                            30 |
+-------------------------------+
1 row in set (0.00 sec)
```

SELECT MINUTE('2009-12-31 23:34:59');

Hasilnya

```
+-------------------------------+
| MINUTE('2009-12-31 23:34:59') |
+-------------------------------+
|                            34 |
+-------------------------------+
1 row in set (0.00 sec)
```

SELECT HOUR('2002-10-31 11:59:59');

Hasilnya

```
+-----------------------------+
| HOUR('2002-10-31 11:59:59') |
+-----------------------------+
|                          11 |
+-----------------------------+
1 row in set (0.00 sec)
```

SELECT TIME('2005-12-31 23:59:59');

Hasilnya

```
+-----------------------------+
| TIME('2005-12-31 23:59:59') |
+-----------------------------+
| 23:59:59                    |
+-----------------------------+
1 row in set (0.00 sec)
```

VIII.5.4 Summarize Data

Kegunaan dari fungsi summarise ini yaitu dengan mengelompokkan data seperti gruup. Lebih jelasnya kita lihat contoh berikut:

```
CREATE TABLE Olahraga
(
        OLID SMALLINT NOT NULL PRIMARY KEY,
        Nama CHAR(25) NOT NULL,
        Tempat ENUM('Indoor', 'Outdoor') NOT NULL,
        Atlit TINYINT UNSIGNED NOT NULL
);
```

```
INSERT INTO Olahraga
VALUES (301, 'FISIK', 'Indoor', 32),
(302, 'MNTAL', 'Indoor', 35),
(303, 'PKRAN', 'Indoor', 28),
(304, 'FISIK', 'Outdoor', 19),
(305, 'FISIK', 'Indoor', 24),
(306, 'MNTAL', 'Outdoor', 21),
(307, 'PKRAN', 'Indoor', 29),
(308, 'PKRAN', 'Outdoor', 29),
(309, 'MNTAL', 'Indoor', 25),
(310, 'MNTAL', 'Outdoor', 30);
```

VIII.5.4.1 Fungsi AVG()

Fungsi ini berguna untuk menghitung rata-rata dari data yang ada. Secra syntax

```
AVG(<expression>)
```

Dan implementasinya:

```
SELECT Nama, Tempat, ROUND(AVG(Atlit)) AS Rata_rata
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+-----------+
| Nama  | Tempat  | Rata_rata |
```

```
+-------+---------+-----------+
| FISIK | Indoor  |       28 |
| FISIK | Outdoor |       19 |
| FISIK | NULL    |       25 |
| MNTAL | Indoor  |       30 |
| MNTAL | Outdoor |       26 |
| MNTAL | NULL    |       28 |
| PKRAN | Indoor  |       29 |
| PKRAN | Outdoor |       29 |
| PKRAN | NULL    |       29 |
| NULL  | NULL    |       27 |
+-------+---------+-----------+
10 rows in set (0.00 sec)
```

VIII.5.4.2 Fungsi SUM()

Fungsi ini berguna untuk menghitung jumlah dari data yang ada. Secra syntax

SUM(<expression>)

Dan implementasinya:

```
SELECT Nama, Tempat, SUM(Atlit) AS Total
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+-------+
| Nama  | Tempat  | Total |
+-------+---------+-------+
| FISIK | Indoor  |    56 |
| FISIK | Outdoor |    19 |
| FISIK | NULL    |    75 |
| MNTAL | Indoor  |    60 |
| MNTAL | Outdoor |    51 |
| MNTAL | NULL    |   111 |
| PKRAN | Indoor  |    57 |
| PKRAN | Outdoor |    29 |
| PKRAN | NULL    |    86 |
| NULL  | NULL    |   272 |
+-------+---------+-------+
10 rows in set (0.00 sec)
```

VIII.5.4.3 Fungsi MIN() and MAX()

Fungsi ini berguna untuk menghitung nilai maximum dan minimum dari data yang ada. Secara syntax

MAX(<expression>)

MIN(<expression>)

Dan implementasi dari MAX:

```
SELECT Nama, Tempat, MAX(Atlit) AS Maximum
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+---------+
| Nama  | Tempat  | Maximum |
+-------+---------+---------+
| FISIK | Indoor  |      32 |
| FISIK | Outdoor |      19 |
| FISIK | NULL    |      32 |
| MNTAL | Indoor  |      35 |
| MNTAL | Outdoor |      30 |
| MNTAL | NULL    |      35 |
| PKRAN | Indoor  |      29 |
| PKRAN | Outdoor |      29 |
| PKRAN | NULL    |      29 |
| NULL  | NULL    |      35 |
+-------+---------+---------+
10 rows in set (0.00 sec)
```

Dan implementasi dari MIN:

```
SELECT Nama, Tempat, MIN(Atlit) AS Minimum
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+---------+
| Nama  | Tempat  | Minimum |
+-------+---------+---------+
| FISIK | Indoor  |      24 |
| FISIK | Outdoor |      19 |
| FISIK | NULL    |      19 |
| MNTAL | Indoor  |      25 |
| MNTAL | Outdoor |      21 |
| MNTAL | NULL    |      21 |
| PKRAN | Indoor  |      28 |
| PKRAN | Outdoor |      29 |
| PKRAN | NULL    |      28 |
| NULL  | NULL    |      19 |
+-------+---------+---------+
10 rows in set (0.00 sec)
```

VIII.5.4.4 Fungsi COUNT()

Fungsi ini berguna untuk menghitung nilai agregat satu dengan nilai agregate lainnya. Secara syntax

```
COUNT([DISTINCT] {<expression> | *})
```

Dan implementasinya:

```
SELECT Nama, Tempat, COUNT(*) AS Jumlah_Atlit
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+--------------+
| Nama  | Tempat  | Jumlah_Atlit |
+-------+---------+--------------+
| FISIK | Indoor  |            2 |
| FISIK | Outdoor |            1 |
| FISIK | NULL    |            3 |
| MNTAL | Indoor  |            2 |
| MNTAL | Outdoor |            2 |
| MNTAL | NULL    |            4 |
| PKRAN | Indoor  |            2 |
| PKRAN | Outdoor |            1 |
| PKRAN | NULL    |            3 |
| NULL  | NULL    |           10 |
+-------+---------+--------------+
10 rows in set (0.05 sec)
```

VIII.5.4.5 Fungsi Bit

Fungsi ini berguna untuk menghitung nilai group dalam sebuah kolom menggunakan sebuah fungsi bit, kemudian bit tersebut dibandingkan dan nilainya pun akan kembali kepada fungsi tersebut. Secara syntax

```
BIT_AND(<expression>)
```

```
BIT_OR(<expression>)
```

```
BIT_XOR(<expression>)
```

Dan implementasi dari BitwiseAND:

```
SELECT Nama, Tempat, BIT_AND(Atlit) AS BitwiseAND
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+------------+
| Nama  | Tempat  | BitwiseAND |
+-------+---------+------------+
| FISIK | Indoor  |          0 |
| FISIK | Outdoor |         19 |
| FISIK | NULL    |          0 |
| MNTAL | Indoor  |          1 |
| MNTAL | Outdoor |         20 |
| MNTAL | NULL    |          0 |
| PKRAN | Indoor  |         28 |
| PKRAN | Outdoor |         29 |
| PKRAN | NULL    |         28 |
| NULL  | NULL    |          0 |
+-------+---------+------------+
10 rows in set (0.00 sec)
```

Dan implementasi dari BitwiseOR :

```
SELECT Nama, Tempat, BIT_OR(Atlit) AS BitwiseOR
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+-----------+
| Nama  | Tempat  | BitwiseOR |
+-------+---------+-----------+
| FISIK | Indoor  |        56 |
| FISIK | Outdoor |        19 |
| FISIK | NULL    |        59 |
| MNTAL | Indoor  |        59 |
| MNTAL | Outdoor |        31 |
| MNTAL | NULL    |        63 |
| PKRAN | Indoor  |        29 |
| PKRAN | Outdoor |        29 |
| PKRAN | NULL    |        29 |
| NULL  | NULL    |        63 |
+-------+---------+-----------+
10 rows in set (0.00 sec)
```

Dan implementasi dari BitwiseXOR :

```
SELECT Nama, Tempat, BIT_XOR(Atlit) AS BitwiseXOR
FROM Olahraga
GROUP BY Nama, Tempat WITH ROLLUP;
```

Hasilnya

```
+-------+---------+------------+
| Nama  | Tempat  | BitwiseXOR |
+-------+---------+------------+
| FISIK | Indoor  |         56 |
```

```
| FISIK | Outdoor |       19 |
| FISIK | NULL    |       43 |
| MNTAL | Indoor  |       58 |
| MNTAL | Outdoor |       11 |
| MNTAL | NULL    |       49 |
| PKRAN | Indoor  |        1 |
| PKRAN | Outdoor |       29 |
| PKRAN | NULL    |       28 |
| NULL  | NULL    |        6 |
+-------+---------+----------+
10 rows in set (0.00 sec)
```

Performance Sistem Operasi

Fungsi ENCODE() and DECODE()

Fungsi ini berguna untuk memberi keamanan pada data yaitu dengan merubah sejumlah data menjadi kombinasi char dan angka, dan ketika data dibutuhkan kembali maka data tersebut dapat dikembalikan seperti semula. Secara syntax

ENCODE(<string>, <key>)

DECODE(<encrypted string>, <key>)

Sebelum masuk implementasi, kita akan membuat tabelnya

```
CREATE TABLE User
(
        UserID SMALLINT NOT NULL PRIMARY KEY,
        Password VARCHAR(20) NOT NULL
);
```

Dan implementasi dari ENCODE:

```
INSERT INTO User
VALUES (10, ENCODE('devy', 'key10'));
```

```
SELECT * FROM User;
```

Hasilnya :

```
+--------+----------+
| UserID | Password |
+--------+----------+
|     10 | ‡0fd     |
+--------+----------+
1 row in set (0.00 sec)
```

Dan implementasi dari DECODE:

```
SELECT UserID, DECODE(Password, 'key10') AS KataKunci
FROM User;
```

Dan Hasilnya:

```
+--------+-----------+
| UserID | KataKunci |
+--------+-----------+
|     10 | devy      |
+--------+-----------+
1 row in set (0.00 sec)
```

Fungsi PASSWORD(), MD5(), SHA(), and SHA1()

Fungsi ini berguna untuk memberikan pilihan tipe pengamanan data yang memiliki fungsi yang sama dengan Encode(), namun sekali di enkripsi tidak akan bisa dibalikan seperti encode(). Secara syntax

PASSWORD(<string>)

MD5(<string>)

SHA(<string>)

Implementasinya:

SELECT PASSWORD(“Januari”);

Hasilnya

```
+-------------------------------------------+
| PASSWORD("Januari")                       |
+-------------------------------------------+
| *62A3A619FE123A995CE92DD4C03E25AFEE1F7AC2 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

SELECT MD5(“Januari”);

Hasilnya

```
+----------------------------------+
| MD5("Januari")                   |
+----------------------------------+
| 1831bebb539949a55a71435e32368a9f |
+----------------------------------+
1 row in set (0.00 sec)
```

SELECT SHA(“Januari”);

Hasilnya

```
+-----------------------------------------+
| SHA("Januari")                          |
+-----------------------------------------+
| 7a0a8c2a659348882d3653aa24212bedfd9e4e70 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

System-Related

Fungsi CURRENT_USER(), SESSION_USER(), SYSTEM_USER(),and USER()
Semua fungsi ini memberikan informasi yang sama yaitu tentang user pengguna dan host selama session terjadi. Secara syntax

USER()

Dan implementasinya

SELECT USER();

Hasilnya

```
+----------------+
| USER()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

Fungsi CONNECTION_ID(), DATABASE(), and VERSION()

Fungsi ini memberikan informasi tentang koneksi yang ada, database, dan versinya. Secara syntax

CONNECTION_ID()

DATABASE()

VERSION()

Dan implementasinya

SELECT CONNECTION_ID();

Hasilnya

```
+-----------------+
| CONNECTION_ID() |
+-----------------+
|               1 |
+-----------------+
1 row in set (0.00 sec)
```

SELECT DATABASE();

Hasilnya

```
+------------+
| DATABASE() |
+------------+
|tugas       |
+------------+
1 row in set (0.00 sec)
```

SELECT VERSION();

Hasilnya

```
+---------------------+
| VERSION()           |
+---------------------+
| 5.1.36-community-log |
+---------------------+
1 row in set (0.00 sec)
```

Fungsi INET_ATON() and INET_NTOA()

Fungsi INET_ATON() berguna untuk merubah IP Address menjadi nilai numeric. Sedangkan INET_NTOA()berguna untuk merubah nilai numeric menjadi IP Address Secara syntax

INET_ATON(<network address>)

INET_NTOA(<network integer>)

Implementasinya

SELECT INET_ATON('127.0.0.1');

Hasilnya

```
+------------------------+
| INET_ATON('127.0.0.1') |
+------------------------+
|             2130706433 |
+------------------------+
1 row in set (0.00 sec)
```

SELECT INET_NTOA(2130706433);

Hasilnya

```
+-----------------------+
| INET_NTOA(2130706433) |
+-----------------------+
| 127.0.0.1             |
+-----------------------+
1 row in set (0.00 sec)
```

Query and Insert

Fungsi FOUND_ROWS()

Fungsi ini memiliki kegunaan yang sama seperti fungsi LIMIT pada clausa SELECT. Secara syntax

FOUND_ROWS()

Implementasinya

SELECT FOUND_ROWS();

Hasilnya

```
+--------------+
| FOUND_ROWS() |
+--------------+
```

```
|           1 |
+--------------+
1 row in set (0.00 sec)
```

Fungsi LAST_INSERT_ID()

Fungsi ini memiliki kegunaan untuk mencatat data yang terakhir pada sebuah data yang terdapat pada kolom AUTO_INCREMENT. Secara syntax

LAST_INSERT_ID()

Implementasinya

SELECT LAST_INSERT_ID();

Hasilnya

```
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                0 |
+------------------+
1 row in set (0.00 sec)
```