PERTEMUAN 9
PENGGUNAAN FUNGSI

MATERI

a. Fungsi perbandingan, alir kontrol, dan *cast* untuk membandingkan dan mengkonversi data
b. Fungsi string, numerik, dan waktu untuk memanipulasi, menghitung, mengkonversi, menyaring, dan menggabungkan data
c. Fungsi agregasi yang dapat digunakan dalam pernyataan SELECT untuk meringkas data yang telah dikelompokkan oleh klausa GROUP BY
d. Fungsi enkripsi, berhubungan dengan sistem, *query* serta *insert* untuk melakukan operasi sistem

MEMBANDINGKAN DAN MENGKONVERSI DATA

FUNGSI PERBANDINGAN

Fungsi perbandingan sama dengan operator perbandingan pada pertemuan sebelumnya. Fungsi-fungsi ini mengijinkan Anda untuk membandingkan nilai-nilai yang berbeda dan, dari perbandingan tersebut, mengembalikan satu dari nilai atau mengembalikan nilai kondisi, true, false, atau NULL. Jika salah satu atau kedua argumen dalam perbandingan adalah NULL, maka NULL yang dikembalikan.

1. Fungsi GREATEST() dan LEAST()
   Dalam fungsi ini mengijinkan Anda untuk membandingkan dua atau lebih nilai dan mengembalikan nilai yang paling besar atau paling kecil, tergantung dari fungsi yang Anda gunakan. Nilai yang dispesifikasikan dapat berupa numerik, string, atau tanggal/waktu dan dibandingkan berdasarkan pada kumpulan karakter saat ini.
   Sintaksnya adalah sebagai berikut:

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

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

   Contoh penggunaan:

```
SELECT GREATEST(32, 21, 0, -9, 23);
```

   Akan menghasilkan nilai 32.

```
SELECT LEAST(32, 21, 0, -9, 23);
```

   Akan menghasilkan nilai -9.

   Contoh dalam pernyataan SELECT. Asumsi Anda telah memiliki tabel bernama BUKU dengan kolom-kolom yang bersesuaian.

```
SELECT * FROM Buku
WHERE ID_Buku=GREATEST(4, 34, 2, 12, 23)
ORDER BY judul;
```

   Dalam aplikasi yang sebenarnya Anda dapat menggantikan nilai-nilai dalam fungsi dengan variabel-variabel, sehingga Anda tidak perlu mengetahui nilai sebenarnya dari nilai-nilai yang dibandingkan.

2. Fungsi COALESCE() dan ISNULL()
   Fungsi COALESCE() mengembalikan nilai pertama dari sederetan argumen yang tidak NULL. Jika semua nilai adalah NULL, maka NULL yang dikembalikan. Sintaksnya adalah sebagai berikut:

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

Untuk fungsi ini, minimal Anda menspesifikasikan satu nilai, meskipun fungsi akan lebih berguna jika banyak nilai yang Anda sediakan. Contoh:

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

Dalam kasus ini, nilai 2 yang dikembalikan karena dia merupakan nilai pertama yang tidak NULL.

Fungsi ISNULL() juka berhubungan dengan nilai null, meskipun output bukan satu dari nilai yang dispesifikasikan. Fungsi ini akan mengembalikan nilai 1 jika ekspresi yang dievaluasi bernilai NULL; sebaliknya, fungsi akan mengembalikan nilai 0. Sintaksnya adalah sebagai berikut:

```
ISNULL (<ekspresi>)
```

Contoh:

```
SELECT ISNULL(1*NULL);
```

Ekspresi dalam pernyataan ini adalah 1*NULL. Sebagaimana telah disebutkan, ekspresi akan dinilai NULL jika salah satu argumen adalah NULL. Dikarenakan ekspresi mengevaluasi NULL, fungsi ISNULL() mengembalikan nilai 1, bukan 0.

3. Fungsi INTERVAL() dan STRCMP()
   Fungsi INTERVAL() membandingkan integer pertama yang terdapat dalam argumen terhadap integer berikutnya. Sintaksnya adalah sebagai berikut:

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

Mulai dari <integer2>, nilai harus tertulis dalam urutan *ascending* (urut naik). Jika <integer1> lebih kecil dari <integer2>, nilai 0 akan dikembalikan. Jika <integer1> lebih kecil dari <integer3>, nilai 1 akan dikembalikan. Jika <integer1> lebih kecil dari <integer4>, nilai 2 akan dikembalikan, dan seterusnya. Contoh:

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

Dalam kasus ini, <integer1> lebih besar dari <integer2>,<integer3>, dan <integer4>, namun lebih kecil dari <integer5>, jadi nilai yang dikembalikan adalah 3, yang menunjukkan posisi dari <integer5>. Dengan kata lain, 7 adalah nilai pertama dalam daftar yang lebih besar dari 6.

Fungsi STRCMP() berbeda dari INTERVAL() dalam hal pembandingan nilai string. Sintaksnya adalah sebagai berikut:

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

Dalam sintaks di atas, fungsi ini membandingkan tepat dua nilai. Fungsi akan mengembalikan nilai 0 jika <ekspresi1> sama dengan <ekspresi2> dan mengembalikan -1 jika <ekspresi1> lebih kecil dari <ekspresi2>. Jika <ekspresi1> lebih besar dari <ekspresi2>, atau jika NULL yang dikembalikan oleh perbandingan, fungsi akan mengembalikan nilai 1. Contoh:

```
SELECT STRCMP('big', 'bigger');
```

Nilai-nilai yang dibandingkan berdasarkan kumpulan karakter saat ini. Karena big lebih kecil dari bigger (alfabet pertama), maka pernyataan tersebut mengembalikan nilai -1.

FUNGSI ALIR KONTROL

Tipe-tipe fungsi yang akan Anda lihat adalah mengembalikan sebuah hasil dari pembandingan kondisi. Nilai yang dikembalikan ditentukan oleh kondisi mana yang bernilai *true*.

1. Fungsi IF()
   Sintaksnya adalah sebagai berikut:

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

Jika <ekspresi1> bernilai true, maka fungsi mengembalikan <ekspresi2>; sebaliknya, fungsi mengembalikan <ekspresi3>. Contoh:

```
SELECT IF(10>20, 'ekspresi benar', 'ekspresi salah');
```

Dalam pernyataan ini, nilai-nilai literal digunakan untuk ekspresi kedua dan ketiga. Karena ekspresi pertama (10>20) bernilai salah, ekspresi ketiga yang dikembalikan. Fungsi ini berguna jika Anda ingin mengembalikan respon spesifik pada nilai yang dikembalikan oleh fungsi IF().

2. Fungsi IFNULL() dan NULLIF()
   Fungsi IFNULL() mengembalikan nilai berdasarkan ekspresi spesifik mana yang bernilai NULL. Fungsi mencakup dua ekspresi, sintaksnya adalah sebagai berikut:

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

Fungsi mengembalikan <ekspresi1> jika dia tidak bernilai NULL; sebaliknya, dia akan mengembalikan <ekspresi2>. Contoh berikut meliputi fungsi IFNULL() dimana ekspresi pertamanya adalah NULL:

```
SELECT IFNULL(10*NULL, 'ekspresi salah');
```

Karena ekspresi pertama (10*NULL) benrilai NULL, nilai NULL tidak dikembalikan. Malahan, ekspresi kedua yang dikembalikan. Dalam kasus ini, ekspresi kedua merupakan nilai literal, jadi dialah nilai yang dikembalikan.

Fungsi NULLIF() agak sedikit berbeda dari fungsi IFNULL(). Fungsi NULLIF() mengembalikan NULL jika <ekspresi1> sama dengan <ekspresi2>; sebaliknya, dia mengembalikan <ekspresi1>. Sintaksnya adalah sebagai berikut:

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

Contoh:

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

Sebagaimana Anda lihat, dalam pernyataan tersebut terdapat dua ekspresi. Karena keduanya sama (dua-duanya mengembalikan nilai 200), nilai NULL yang dikembalikan, bukannya nilai 200 yang dihasilkan ekspresi pertama.

3. Fungsi CASE()
   Fungsi ini menyediakan fleksibilitas yang lebih ketimbang sebelumnya meskipun agak kompleks. Fungsi CASE() mendukung dua format yang berbeda.
   Sintaks pertama adalah sebagai berikut:

```
CASE WHEN <ekspresi> THEN <hasil>
[{WHEN <ekspresi> THEN <hasil>}...]
[ELSE <hasil>]
END
```

Dalam sintaks tersebut ditunjukkan, Anda harus menspesifikasikan kata kunci CASE, diikuti minimal satu klausa WHEN…THEN. Klausa WHEN…THEN menspesifikasikan ekspresi yang dievaluasi dan hasil yang dikembalikan jika ekspresi tersebut bernilai true. Anda dapat menspesifikasikan klausa WHEN…THEN sebanyak kebutuhan Anda. Klausa berikut adalah ELSE, yang bersifat opsional. Klausa ELSE menyediakan nilai default jika ekspresi yang dievaluasi tidak bernilai true. Dan terakhir, fungsi CASE() diterminasi dengan menggunakan kata kunci END.

Contoh:

```
SELECT CASE
WHEN 10*2=30 THEN '30 benar'
WHEN 10*2=40 THEN '40 benar'
ELSE 'Seharusnya 10*2=20'
END;
```

Sebagaimana Anda lihat, klausa pertama WHEN…THEN menspesifikasikan ekspresi (10*20=30) dan sebuah hasil ('30 benar'). Hasil akan dikembalikan jika ekspresi bernilai true. Dalam pernyataan terdapat dua klausa WHEN…THEN; jika

ekspresi yang pertama bernilai true, maka hasil yang akan dikembalikan. Jika ekspresi kedua dalam klausa WHEN…THEN bernilai true, hasilnya akan dikembalikan. Jika tidak ada satupun dari klausa yang bernilai true, hasil dalam klausa ELSE yang akan dikemablikan. Dalam kasus ini, hasil dalam klausa ELSE yang dikembalikan karena tidak ada ekspresi dalam klausa WHEN...THEN yang bernilai true.

Versi fungsi CASE() berikut ini berbeda dari sebelumnya, sintaksnya adalah:

```
CASE <ekspresi>
WHEN <nilai> THEN <hasil>
[{WHEN <nilai> THEN <hasil>}...]
[ELSE <hasil>]
END
```

Perbedaan utama dalam versi ini adalah bahwa ekspresi yang dispesifikasikan setelah kata kunci CASE, dan klausa WHEN...THEN terdapat nilai-nilai yang dihasilkan dari ekspresi tersebut. Contoh:

```
SELECT CASE 10*2
WHEN 20 THEN '20 benar'
WHEN 30 THEN '30 benar'
WHEN 40 THEN '40 benar'
END;
```

Sebagaimana Anda lihat, fungsi CASE() meliputi ekspresi (10*2) setelah kata kunci CASE. Tiga klausa WHEN…THEN mengikuti kata kunci CASE, masing-masing berisi kemungkinan nilai yang merepresentasikan nilai yang dikembalikan oleh ekspresi.

Dalam contoh pernyataan di atas akan dikembalikan nilai '20 benar', karena hasil evaluasi 10*2 adalah 20. Jika tidak ada satupun dalam klausa WHEN…THEN yang bernilai true, maka akan menghasilkan kosong. Kecuali Anda menspesifikasikan hasil setelah klausa ELSE, maka hasil setelah ELSE itulah yang dikembalikan.

FUNGSI *CAST*

Fungsi *cast* mengijinkan Anda untuk mengkonversi nilai ke tipe data tertentu atau untuk menugaskan sebuah set karakter ke sebuah nilai. Sintaksnya adalah:

```
CAST(<ekspresi> AS <tipe>)
```

Fungsi mengkonversi nilai yang dikembalikan oleh ekspresi ke tipe konversi yang dispesifikasi, dimana mengikuti kata kunci AS. Fungsi CAST() mendukung sejumlah tipe konversi yang terbatas. Tipe-tipe ini sama dengan tipe-tipe data, namun mereka spesifik ke fungsi CAST() dan fungsi CONVERT() dan digunakan untuk tujuan yang berbeda, dimana untuk menspesifikasikan bagaimana data tersebut dispesifikasikan. Tipe data, di lain hal, menspesifikasikan tipe data yang bisa dimasukkan ke sebuah kolom.

Tipe konversi yang tersedia pada fungsi CAST adalah sebagai berikut:
- BINARY
- CHAR
- DATE
- DATETIME
- SIGNED [INTEGER]
- TIME
- UNSIGNED [INTEGER]

Misal, Anda mungkin memiliki sebuah nilai numerik (salah satu dari nilai literal atau nilai yang dikembalikan oleh sebuah ekspresi) yang ingin Anda konversi ke tipe konversi DATE. Contoh pernyataananya adalah sebagai berikut:

```
SELECT CAST(20061231 AS DATE);
```

Hasil yang dikembalikan adalah dalam format DATE yaitu (2006-12-31).

Fungsi CONVERT() mengijinkan Anda untuk mengkonversi tanggal dengan cara yang sama sebagaimana fungsi CAST(), hanya formatnya agak sedikit berbeda.

Sintaksnya dalah sebagai berikut:

```
CONVERT(<ekspresi>, <tipe>)
```

Perhatikan bahwa Anda perlu menspesifikasikan hanya ekspresi dan tipe konversi, tanpa kata kunci AS, namun Anda harus memisahkan keduanya dengan koma. Tipe konversi yang dapat Anda gunakan dalam fungsi ini sama dengan pada fungsi CAST(). Contoh:

```
SELECT CONVERT(20061231, DATE);
```

Perhatikan bahwa Anda hanya perlu menspesifikasikan nilai numerik dan tipe konversi DATE. Fungsi CONVERT, bagaimanapun, juga meliputi bentuk lain, sebagaimana terlihat dalam sintaks berikut:

```
CONVERT(<ekspresi> USING <character set>)
```

Bentuk ini digunakan untuk menugaskan *character set* ke ekspresi yang ditentukan. Misal, contoh berikut mengkonversi string ke *character set* latin 2:

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

Dalam pernyataan ini, dalam fungsi terdapat ekspresi (string literal), diikuti oleh kata kunci USING dan nama dari *character set* (latin2). Nilai yang dikembalikan adalah string dalam *character set* yang baru. (Dalam hal ini, nilainya sama dengan yang muncul dalam fungsi.)

MENGELOLA TIPE DATA YANG BERBEDA

MySQL mencakup fungsi-fungsi yang mengijinkan Anda mengelola data string, numerik, dan tanggal/waktu. Anda dapat menggunakan fungsi-fungsi ini dengan bantuan fungsi-fungsi yang telah Anda lihat sebelumnya. Dalam kebanyakan kasus, sebuah fungsi dapat di-*embedded* (tanam/sisipkan) sebagai sebuah argumen dalam fungsi lain, yang membuat penggunaan semua fungsi lebih handal.

FUNGSI STRING

Fungsi string mengijinkan Anda untuk memanipulasi dan menyaring nilai-nilai string. MySQL mendukung sejumlah besar fungsi-fungsi string.

1. Fungsi ASCII() dan ORD()
   Fungsi ASCII() mengijinkan Anda untuk mengidentifikasi nilai numerik dari karakter pertama dari string. Sintaksnya adalah:

   ```
   ASCII(<string>)
   ```

   Untuk menggunakan fungsi ASCII(), Anda hanya perlu mengidentifikasi stringnya, contoh:

   ```
   SELECT ASCII('informatika');
   ```

   Dalam pernyataan tersebut di atas mengembalikan nilai numerikdari karakter pertama, yaitu i. Nilai numerik dari i adalah 105.

   Fungsi ASCII() bekerja hanya pada karakter *single-byte* (dengan nilai mulai 0 sampai 255). Untuk karakter *multi-byte*, Anda harus menggunakan fungsi ORD(), dimana sintaksnya adalah sebagai berikut:

   ```
   ORD(<string>)
   ```

   Fungsi ORD() bekerja seperti fungsi ASCII() keculai bahwa ia juga mendukung karakter *multibyte*. Contoh:

   ```
   SELECT ORD(37);
   ```

   Pada pernyataan di atas akan mengembalikan nilai numerik dari karakter pertama. Untuk nomor 3, nilai numeriknya adalah 51. Jika Anda menentukan sebuah angka, daripada string biasa Anda tidak perlu menyertakan tanda petik

tunggal. Dengan tambahan, jika argumen fungsi merupakan karakter *single-byte*, hasilnya sama dengan fungsi ASCII().

2. Fungsi CHAR_LENGTH(), CHARACTER_LENGTH(), and LENGTH()
   Fungsi CHAR_LENGTH() dan CHARACTER_LENGTH(), dimana sinonim, mengembalikan jumlah karakter dari string yang dispesifikasikan. Sintaksnya:

```
CHAR_LENGTH(<string>)
```

   Contoh:

```
SELECT CHAR_LENGTH('teknik informatika');
```

   Pernyataan tersebut mengembalikan nilai 18, dimana merupakan jumlah karakter dalam string, termasuk spasi.

   Fungsi LENGTH() juga mengembalikan panjang dari string, hanya saja panjangnya diukur dalam satuan byte, bukan karakter. Sintaksnya adalah:

```
LENGTH(<string>)
```

   Jika Anda menggunakan fungsi LENGTH() dengan karakter *single-byte*, hasilnya akan sama dengan fungsi CHAR_LENGTH(), sebagaimana nampak pada pernyataan berikut:

```
SELECT LENGTH('teknik informatika');
```

   Dalam hal ini, hasilnya sekali lagi adalah 18. Jika ini merupakan string karakter *double-byte*, hasilnya akan menjadi 36, karena fungsi LENGTH() mengukurnya dalam satuan byte, bukan karakter.

3. Fungsi CONCAT() dan CONCAT_WS()
   Fungsi ini digunakan untuk menggabungkan string-string.

```
CONCAT(<string1>,<string2>,<string3>, dst)
```

   Contoh:

```
SELECT CONCAT('Ibu',' ','dan',' ','Ayah');
```

   Hasilnya akan berupa output string: **Ibu dan Ayah**

   Sedangkan untuk fungsi CONCAT_WS(), sintaksnya:

```
CONCAT_WS(<pemisah>,<string2>,<string3>, dst)
```

   Contoh:

```
SELECT CONCAT_WS(' ','Ibu','dan','ayah');
```

   Hasilnya sama dengan pernyataan sebelumnya, yaitu: **Ibu dan Ayah**

4. Fungsi INSTR() dan LOCATE()
   Fungsi INSTR() digunakan untuk mencari sebuah substring dalam string, dimana hasilnya adalah berupa numerik yang merupakan posisi tempat ditemukannya substring tersebut.

```
INSTR(<string>,<substring>)
```

   Contoh:

```
SELECT INSTR('ibu dan ayah', 'dan');
```

   Akan mengembalikan nilai 5.

   Sedangkan untuk fngsi LOCATE(), sintaksnya:

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

Atau,

```
LOCATE(<substring>,<string>,<posisi>)
```

Contoh:

```
SELECT LOCATE('dan', 'ibu dan ayah dan anak');
```

Akan mengembalikan nilai 5. Perhatikan contoh berikutnya:

```
SELECT LOCATE('dan', 'ibu dan ayah dan anak', 6);
```

Akan menghasilkan nilai 14.

5. Fungsi LCASE(), LOWER(), UCASE(), dan UPPER()
   Fungsi LCASE() atau LOWER() digunakan untuk mengubah string menjadi huruf kecil, sebaliknya UCASE() atau UPPER() untuk mengubah string ke huruf kapital (besar).

   Sintaksnya:

   ```
   LCASE(<string>) atau UCASE(<string>)
   ```

   Contoh:

   ```
   SELECT LCASE('Teknik Informatika');
   ```

6. Fungsi LEFT() dan RIGHT()
   Fungsi LEFT() digunakan untuk mengambil sejumlah string dari arah kiri, sedangkan fungsi RIGHT() dari arah kanan. Sintaksnya:

   ```
   LEFT(<string>,<jumlah>) atau RIGHT(<string>,<jumlah>)
   ```

   Contoh:

   ```
   SELECT LEFT('Teknik Informatika', 6);
   ```

   Hasilnya: **Teknik**

   ```
   SELECT RIGHT('Teknik Informatika', 11);
   ```

   Hasilnya: **Informatika**

7. Fungsi REPEAT() dan REVERSE()
   Fungsi REPEAT() digunakan untuk mengulang string sejumlah yang diinginkan. Sintaksnya:

   ```
   REPEAT(<string>, <jumlah>);
   ```

   Contoh:

   ```
   SELECT REPEAT('Halo ', 3);
   ```

   Hasilnya: **Halo Halo Halo**

   Sedangkan REVERSE() digunakan untuk membalik posisi string. Sintaksnya:

   ```
   REVERSE (<string>)
   ```

   Contoh:

   ```
   SELECT REVERSE('Kasur Rusak');
   ```

   Hasilnya: **kasuR rusaK**

8. Fungsi SUBSTRING()
   Fungsi ini digunakan untuk mengambil sejumlah substring dari suatu string sebanyak jumlah yang diinginkan mulai dari posisi tertentu.

```
SUBSTRING (<string>, <posisi>, <jumlah>)
```

   Contoh:

```
SELECT SUBSTRING('Teknik Informatika', 8, 4);
```

   Hasilnya: **Info**

FUNGSI NUMERIK

1. Fungsi CEIL(), CEILING() dan FLOOR()
   Fungsi CEIL() atau CEILING() mengembalikan nilai bilangan bulat terdekat yang lebih besar dari nilai numerik yang dimasukkan. Sedangkan fungsi FLOOR() sebaliknya, yaitu mengembalikan nilai bilangan bulat terdekat yang lebih kecil dari nilai yang dimasukkan. Sintaksnya:

```
CEIL(<bilangan>) atau FLOOR(<bilangan>)
```

   Contoh:

```
SELECT CEIL( 9.327 );
```

   Hasilnya: **10**

```
SELECT FLOOR( 9.327 );
```

   Hasilnya: **9**

2. Fungsi MOD()
   Fungsi mengembalikan nilai mod (sisa hasil bagi) dari bilangan tertentu terhadap bilangan lainnya.

   Sintaksnya:

```
MOD(<bilangan1>, <bilangan2>)
```

   Contoh: 23 mod 4

```
SELECT MOD(23, 4);
```

   Hasilnya: **3**

3. Fungsi PI()
   Digunakan untuk memunculkan nilai PI.

```
SELECT PI();
```

   Hasilnya: **3.141593**

4. Fungsi POW() atau POWER()
   Fungsi mengembalikan nilai pangkat terhadap suatu bilangan.

   Contoh: 5 pangkat 2 ATAU 5^2

```
SELECT POW(5, 2);
```

   Hasilnya: **25**

5. Fungsi ROUND() dan TRUNCATE()
   Fungsi ROUND() digunakan untuk membulatkan bilangan ke atas sesuai dengan jumlah angka desimal (di belakang koma) yang ditentukan.

```
SELECT ROUND(4.27943, 2);
```

Hasilnya: **4.28**

Sedangkan fungsi TRUNCATE() digunakan untuk memotong sejumlah angka sesuai dengan jumlah angka desimal yang ditentukan.

```
SELECT TRUNCATE(4.27943, 2);
```

Hasilnya: **4.27**

6. Fungsi SQRT()
   Digunakan untuk mencari nilai akar dari suatu bilangan.

```
SELECT SQRT(16);
```

Hasilnya: **4**

FUNGSI WAKTU DAN TANGGAL

1. Fungsi ADDDATE(), DATE_ADD(), SUBDATE(), DATE_SUB(), dan EXTRACT()

   Fungsi ADDATE() dan DATE_ADD(), dimana merupakan fungsi yang hampir sama, mengijinkan Anda menambahkan interval tanggal pada nilai tanggal saat ini, berikut sintaksnya:

```
ADDDATE(<tanggal>, INTERVAL <ekspresi> <tipe>)
```

   Sebagaimana Anda lihat di sintaks, dalam fungsi tersebut terdapat dua argument, nilai <tanggal> dan klausa INTERVAL. Nilai tanggal dapat berisi tanggal atau nilai literal tanggal/waktu atau nilai yang diturunkan dari sebuah ekspresi. Nilai ini berlaku sebagai nilai akar/awal dimana waktu ditambahkan. Klausa INTERVAL membutuhkan sebuah <ekspresi>, dimana harus merupakan sebuah nilai waktu dalam format yang dapat diterima, dan nilai <tipe>.
   Dalam tabel berikut ini daftar tipe-tipe yang dapat Anda tentukan dalam klausa INTERVAL dan format untuk ekspresi yang digunakan dalam tipe tersebut:

<type>	<expression> format
MICROSECOND	<microseconds>
SECOND	<seconds>
MINUTE	<minutes>
HOUR	<hours>
DAY	<days>
MONTH	<months>
YEAR	<years>
SECOND_MICROSECOND	'<seconds>.<microseconds>'
MINUTE_MICROSECOND	'<minutes>.<microseconds>'
MINUTE_SECOND	'<minutes>:<seconds>'

<type>	<expression> format
HOUR_MICROSECOND	'<hours>.<microseconds>'
HOUR_SECOND	'<hours>:<minutes>:<seconds>'
HOUR_MINUTE	'<hours>:<minutes>'
DAY_MICROSECOND	'<days>.<microseconds>'
DAY_SECOND	'<days> <hours>:<minutes>:<seconds>'
DAY_MINUTE	'<days> <hours>:<minutes>'
DAY_HOUR	'<days> <hours>'
YEAR_MONTH	'<years>-<months>'

SELENGKAPNYA DI MODUL BEGINNING MYSQL