

# MySQL “Membuat Ringkasan”

Oleh :

**Janner Simarmata**

[sijanner@yahoo.com](mailto:sijanner@yahoo.com)

<http://simarmata.cogia.net>

*Dipublikasikan dan didedikasikan  
untuk perkembangan pendidikan di Indonesia melalui*

**MateriKuliah.Com**

***Lisensi Pemakaian Artikel:***

*Seluruh artikel di **MateriKuliah.Com** dapat digunakan, dimodifikasi dan disebarakan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut Penulis. Hak Atas Kekayaan Intelektual setiap artikel di **MateriKuliah.Com** adalah milik Penulis masing-masing, dan mereka bersedia membagikan karya mereka semata-mata untuk perkembangan pendidikan di Indonesia. **MateriKuliah.Com** sangat berterima kasih untuk setiap artikel yang sudah Penulis kirimkan.*

## 1.1 Pendahuluan

Sistem basis data berguna untuk menyimpan dan mengambil record, dia juga dapat mengumpulkan informasi untuk meringkas data anda dalam bentuk yang lebih ringkas. Ringkasan berguna ketika anda ingin gambaran keseluruhan dari pada data detilnya. Ringkasan biasanya lebih mudah dimengerti dari pada daftar yang panjang. Ringkasan memungkinkan anda menjawab pertanyaan seperti “**Berapa banyak?**” atau “**Berapa totalnya?**” atau “**Berapa kisaran nilainya?**”. Jika anda menjalankan sebuah bisnis, anda ingin mengetahui berapa banyak pelanggan yang anda punya di tiap-tiap negara bagian, atau berapa banyak penjualan tiap bulan. Anda dapat menentukan perhitungan per negara bagian dengan membuat daftar pelanggan dan menghitungnya, tapi ini dapat dilakukan oleh MySQL. Demikian juga untuk menentukan penjualan per bulan, daftar pemesanan biasanya tidak berguna jika anda harus menjumlahkan total pemesanan secara manual.

Contoh diatas menunjukkan dua tipe ringkasan. Yang pertama (*jumlah pelanggan per negara bagian*) adalah ringkasan hitungan. Isi dari masing-masing record penting hanya untuk tujuan penempatannya pada kelompok yang sesuai atau kategori untuk penghitungan. Ringkasan seperti ini biasanya berbentuk histogram, dimana anda mengurutkan record kedalam sebuah kumpulan dan menghitung jumlah record dalam masing-masing kelompok. Contoh kedua (*jumlah penjualan per bulan*) adalah contoh ringkasan yang didasarkan pada isi record. Total penjualan dihitung dari nilai penjualan dalam masing-masing record pemesanan.

Namun bentuk lain dari ringkasan tidak menghasilkan baik cacah maupun jumlah, tapi hanya daftar nilai unik. Ini berguna jika anda tidak peduli berapa banyak record dari masing-masing nilai yang muncul, tapi hanya nilai yang mana yang muncul. Jika anda ingin mengetahui negara bagian dimana anda mempunyai pelanggan, anda membutuhkan daftar nama negara bagian yang terdapat pada record, bukan daftar yang berisi nilai negara bagian dari setiap record. Kadang-kadang berguna untuk mengaplikasikan satu teknik ringkasan yang berasal dari hasil ringkasan lain. Sebagai contoh, untuk menentukan berapa banyak negara bagian tempat tinggal pelanggan anda, buatlah daftar negara bagian secara unik, kemudian hitunglah jumlahnya.

Tipe ringkasan yang dapat anda lakukan tergantung pada jenis datanya. Ringkasan mencacah dapat dibuat dari sembarang nilai, baik itu angka, string, ataupun tanggal. Untuk ringkasan yang mengikutsertakan jumlah atau rata-rata, hanya nilai numerik yang dapat digunakan. Anda dapat menghitung record dari nama-nama negara bagian untuk menghasilkan analisis demografi dari tempat tinggal pelanggan, tapi anda tidak dapat menambah atau mencari rata-rata nama negara bagian.

Operasi ringkasan pada MySQL meliputi bentuk-bentuk SQL berikut ini :

- Untuk menghitung nilai ringkasan dari sekumpulan nilai individual, gunakan satu dari fungsi-fungsi yang dikenal sebagai fungsi agregat. Ini disebut demikian karena fungsi-fungsi ini beroperasi pada agregat (*kelompok*) nilai. Fungsi agregat termasuk COUNT ( ) , yang menghitung record atau nilai dalam hasil query; MIN ( ) dan MAX ( ) , yang mencari nilai terkecil dan terbesar; dan SUM ( ) dan AVG ( ) , yang menghasilkan jumlah dan rata-rata nilai. Fungsi-fungsi ini dapat digunakan untuk menghitung nilai untuk seluruh hasil, atau dengan klausa GROUP BY untuk mengelompokkan baris kedalam subset dan mendapatkan nilai agregat untuk setiap subset.
- Untuk mendapatkan daftar nilai unik, gunakan SELECT DISTINCT dari pada SELECT.
- Untuk menghitung berapa banyak nilai unik yang ada, gunakan COUNT(DISTINCT) daripada COUNT ( ) .

Tabel yang digunakan sebagai contoh disini adalah tabel driver\_log dan mail. Yang juga akan digunakan adalah tabel states, yang mempunyai baris yang berisi informasi tentang negara bagian di Amerika.

```
mysql> SELECT * FROM states ORDER BY name;
```

name	abbrev	statehood	pop
Alabama	AL	1819-12-14	4040587
Alaska	AK	1959-01-03	550043
Arizona	AZ	1912-02-14	3665228
Arkansas	AR	1836-06-15	2350725
California	CA	1850-09-09	29760021
Colorado	CO	1876-08-01	3294394
Connecticut	CT	1788-01-09	3287116

...

Kolom name dan abbrev berisi daftar nama lengkap negara bagian dan singkatannya. Kolom statehood menunjukkan tanggal dimana negara bagian itu masuk sebagai negara bagian Amerika. Sedang pop adalah populasi negara bagian terhitung pada April 1990, seperti yang dilaporkan oleh Biro Sensus Amerika.

## 1.2 Meringkas menggunakan COUNT()

### Penyelesaian

Gunakan fungsi COUNT( ).

### Pembahasan

Untuk menghitung jumlah baris pada seluruh tabel atau yang sesuai dengan kondisi tertentu, gunakan fungsi COUNT( ). Sebagai contoh, untuk menampilkan isi dari record dalam suatu tabel, anda dapat menggunakan query SELECT \*, tapi untuk menghitungnya, gunakan SELECT COUNT( ). Tanpa klausa WHERE, query menghitung semua record dalam tabel, seperti query berikut ini, yang menunjukkan berapa banyak baris isi dari tabel driver\_log :

```
mysql> SELECT COUNT(*) FROM driver_log;
```

COUNT(*)
10

Jika anda tidak tahu berapa banyak negara bagian di Amerika, query ini akan memberi tahu anda:

```
mysql> SELECT COUNT(*) FROM states;
```

COUNT(*)
50

Jika yang anda inginkan adalah penghitungan kasar dan anda mempunyai MySQL 3.23 atau setelahnya, untuk menghindari scan penuh gunakan SHOW TABLE STATUS dan periksa nilai baris pada output. Pada tabel InnoDB, hasil query akan nampak seperti berikut ini :

```
mysql> SHOW TABLE STATUS FROM cookbook LIKE 'states'\G
```

```
***** 1. row *****
```

```

Name: states
Type: InnoDB
Row_format: Dynamic
Rows: 50
Avg_row_length: 327
Data_length: 16384
Max_data_length: NULL
Index_length: 0
Data_free: 0
Auto_increment: NULL
Create_time: NULL
Update_time: NULL
Check_time: NULL
Create_options:
Comment: InnoDB free: 479232 kB

```

Untuk menghitung hanya jumlah baris yang sesuai dengan kondisi tertentu, tambahkan klausa WHERE yang sesuai ke query tersebut. Kondisi ini bisa sembarang, yang membuat COUNT() berguna untuk menjawab banyak bentuk pertanyaan :

- Berapa kali pengemudi bepergian lebih dari 200 mil sehari ?

```
mysql> SELECT COUNT(*) FROM driver_log WHERE miles > 200;
```

```

+-----+
| COUNT(*) |
+-----+
| 4         |
+-----+

```

- Berapa hari Suzi mengemudi ?

```
mysql> SELECT COUNT(*) FROM driver_log WHERE name = 'Suzi';
```

```

+-----+
| COUNT(*) |
+-----+
| 2         |
+-----+

```

- Berapa jumlah negara bagian Amerika pada awal abad 20 ?

```
mysql> SELECT COUNT(*) FROM states WHERE statehood < '1900-01-01';
```

```

+-----+
| COUNT(*) |
+-----+
| 45        |
+-----+

```

- Berapa banyak negara bagian yang masuk pada abad 19 ?

```
mysql> SELECT COUNT(*) FROM states
```

```
-> WHERE statehood BETWEEN '1800-01-01' AND '1899-12-31';
```

```

+-----+
| COUNT(*) |
+-----+
| 29        |
+-----+

```

Fungsi COUNT() sebenarnya mempunyai dua bentuk. Bentuk yang telah kita gunakan, COUNT(\*), menghitung baris. Bentuk lain, COUNT(ekspresi), mengambil nama kolom atau argumen ekspresi dan menghitung jumlah nilai yang tidak NULL. Query berikut ini

menunjukkan bagaimana untuk membuat baik menghitung baris dari suatu tabel dan menghitung jumlah nilai non NULL dalam salah satu kolomnya :

```
SELECT COUNT(*), COUNT(mycol) FROM mytbl;
```

Fakta bahwa COUNT(ekspresi) tidak menghitung nilai NULL berguna ketika membuat banyak penghitungan dari kelompok nilai yang sama. Untuk menghitung jumlah perjalanan Sabtu dan Minggu dalam tabel driver\_log dengan satu query, lakukan ini :

```
mysql> SELECT
-> COUNT(IF(DAYOFWEEK(trav_date)=7,1,NULL)) AS 'Saturday trips',
-> COUNT(IF(DAYOFWEEK(trav_date)=1,1,NULL)) AS 'Sunday trips'
-> FROM driver_log;
+-----+-----+
| Saturday trips | Sunday trips |
+-----+-----+
| 1              | 2            |
+-----+-----+
```

Atau untuk menghitung perjalanan akhir minggu dan hari kerja, lakukan ini :

```
mysql> SELECT
-> COUNT(IF(DAYOFWEEK(trav_date) IN (1,7),1,NULL)) AS 'weekend trips',
-> COUNT(IF(DAYOFWEEK(trav_date) IN (1,7),NULL,1)) AS 'weekday trips'
-> FROM driver_log;
+-----+-----+
| weekend trips | weekday trips |
+-----+-----+
| 3           | 7            |
+-----+-----+
```

Ekspresi IF( ) menentukan, untuk tiap nilai kolom, apakah nilai itu harus dihitung atau tidak. Jika ya, ekspresi akan mengevaluasi ke 1 dan COUNT( ) menghitungnya. Jika tidak, ekspresi mengevaluasi ke NULL dan COUNT( ) mengabaikannya. Efeknya adalah untuk menghitung jumlah nilai yang memenuhi kondisi yang diberikan sebagai argumen pertama dari IF( ) .

### 1.3 Meringkas dengan MIN() dan MAX()

#### Penyelesaian

Gunakan MIN( ) untuk mencari nilai terkecil, MAX( ) untuk mencari yang terbesar.

#### Pembahasan

Mencari nilai yang terbesar atau yang terkecil adalah semata masalah pengurutan, bedanya disini anda tidak mencari seluruh nilai yang diurutkan, tapi anda hanya memilih nilai tunggal pada akhir atau awal urutan. Operasi semacam ini bisa diterapkan pada pertanyaan tentang terkecil, terbesar, tertua, terbaru, paling mahal, paling murah, dan sebagainya. Salah satu cara untuk mencari nilai seperti itu adalah menggunakan fungsi MIN( ) dan MAX( ) . Cara lain adalah menggunakan LIMIT.

Karena MIN( ) dan MAX( ) menentukan nilai ekstrem dalam sekelompok nilai, mereka berguna untuk mencirikan jangkauan :

- Berapa jangkauan tanggal yang dinyatakan oleh baris dalam tabel mail? Manakah pesan yang terbesar dan terkecil ?

```
mysql> SELECT
-> MIN(t) AS earliest, MAX(t) AS latest,
```

```
-> MIN(size) AS smallest, MAX(size) AS largest
-> FROM mail;
```

earliest	latest	smallest	largest
2001-05-11 10:15:08	2001-05-19 22:21:51	271	2394482

- Berapakah jarak perjalanan terpendek dan terjauh dalam tabel driver\_log ?

```
mysql> SELECT MIN(miles) AS shortest, MAX(miles) AS longest
-> FROM driver_log;
```

shortest	longest
79	502

- Berapakah populasi terkecil dan terbesar di negara bagian di Amerika ?

```
mysql> SELECT MIN(pop) AS 'fewest people', MAX(pop) AS 'most people'
-> FROM states;
```

fewest people	most people
453588	29760021

- Manakah negara bagian yang terletak diurutkan pertama dan terakhir bila diurutkan berdasarkan huruf pertama ?

```
mysql> SELECT MIN(name), MAX(name) FROM states;
```

MIN(name)	MAX(name)
Alabama	Wyoming

MIN() dan MAX() tidak perlu diaplikasikan secara langsung ke nilai kolom. Mereka juga bekerja dengan ekspresi atau nilai yang dihasilkan dari nilai kolom. Misalnya, untuk mencari panjang dari nama negara bagian yang terpanjang dan terpendek, lakukan ini :

```
mysql> SELECT MIN(LENGTH(name)) AS shortest, MAX(LENGTH(name)) AS longest
-> FROM states;
```

shortest	longest
4	14

## 1.4 Meringkas dengan SUM() dan AVG()

### Penyelesaian

Gunakan fungsi SUM() atau AVG() .

### Pembahasan

SUM() dan AVG() menghasilkan total dan rata-rata (*mean*) dari sekelompok nilai :

- Berapa jumlah total lalu lintas surat dan ukuran rata-rata dari tiap-tiap pesan ?

```
mysql> SELECT SUM(size) AS 'total traffic',
```

```
-> AVG(size) AS 'average message size'
-> FROM mail;
+-----+-----+
| total traffic | average message size |
+-----+-----+
| 3798185      | 237386.5625          |
+-----+-----+
```

- Berapa mil sopir-sopir dalam tabel driver\_log bepergian ? Berapa mil rata-rata perjalanan per hari ?

```
mysql> SELECT SUM(miles) AS 'total miles',
-> AVG(miles) AS 'average miles/day'
-> FROM driver_log;
+-----+-----+
| total miles | average miles/day |
+-----+-----+
| 2166       | 216.6000          |
+-----+-----+
```

- Berapa populasi total Amerika ?

```
mysql> SELECT SUM(pop) FROM states;
+-----+
| SUM(pop) |
+-----+
| 248102973 |
+-----+
```

SUM() dan AVG() adalah fungsi numerik, sehingga mereka tidak bisa digunakan dengan nilai string atau waktu. Dilain pihak, kadang-kadang anda dapat merubah nilai non numerik ke bentuk numerik. Misalkan sebuah tabel menyimpan nilai waktu yang mewakili waktu yang sudah berlangsung :

```
mysql> SELECT t1 FROM time_val;
+-----+
| t1      |
+-----+
| 15:00:00 |
| 05:01:30 |
| 12:30:20 |
+-----+
```

Untuk menghitung total waktu yang sudah berlangsung, gunakan TIME\_TO\_SEC() untuk merubah nilainya ke detik sebelum menjumlahkannya. Hasilnya juga akan dalam bentuk detik; apabila anda ingin hasil penjumlahan dalam bentuk waktu, gunakan SEC\_TO\_TIME() :

```
mysql> SELECT SUM(TIME_TO_SEC(t1)) AS 'total seconds',
-> SEC_TO_TIME(SUM(TIME_TO_SEC(t1))) AS 'total time'
-> FROM time_val;
+-----+-----+
| total seconds | total time |
+-----+-----+
| 117110       | 32:31:50  |
+-----+-----+
```

## 1.5 Menggunakan DISTINCT untuk Menghilangkan Duplikasi

### Penyelesaian

Gunakan `DISTINCT` untuk memilih nilai unik, atau `COUNT(DISTINCT)` untuk menghitungnya.

### Pembahasan

Sebuah operasi ringkasan yang tidak menggunakan fungsi agregat adalah untuk menentukan nilai atau baris yang mana yang terdapat pada sebuah dataset dengan menghilangkan duplikasi. Lakukan ini dengan `DISTINCT` (atau `DISTINCTROW`, yang sebenarnya sinonim). `DISTINCT` berguna untuk mengambil hasil query, dan sering dikombinasikan dengan `ORDER BY` untuk menempatkan nilai dalam urutan yang lebih baik. Sebagai contoh, jika anda ingin mengetahui nama pengemudi yang terdaftar pada tabel `driver_log`, gunakan query berikut ini :

```
mysql> SELECT DISTINCT name FROM driver_log ORDER BY name;
+-----+
| name |
+-----+
| Ben  |
| Henry|
| Suzi |
+-----+
```

Sebuah query tanpa `DISTINCT` menghasilkan nama yang sama, tapi tidak mudah dimengerti :

```
mysql> SELECT name FROM driver_log;
+-----+
| name |
+-----+
| Ben  |
| Suzi |
| Henry|
| Henry|
| Ben  |
| Henry|
| Suzi |
| Henry|
| Ben  |
| Henry|
+-----+
```

Jika anda ingin mengetahui berapa banyak pengemudi yang ada, gunakan `COUNT(DISTINCT)` :

```
mysql> SELECT COUNT(DISTINCT name) FROM driver_log;
+-----+
| COUNT(DISTINCT name) |
+-----+
| 3                     |
+-----+
```

`COUNT(DISTINCT)` mengabaikan nilai `NULL`. Jika anda juga ingin menghitung `NULL` sebagai satu nilai, lakukan ini :

```
COUNT(DISTINCT val) + IF(COUNT(IF(val IS NULL,1,NULL))=0,0,1)
```

Efek yang sama dapat dicapai menggunakan ekspresi-ekspresi berikut ini :



```
COUNT(DISTINCT val) + IF(SUM(ISNULL(val))=0,0,1)
COUNT(DISTINCT val) + (SUM(ISNULL(val))!=0)
```

COUNT(DISTINCT) tersedia pada MySQL 3.23.2. Sebelumnya, anda harus menggunakan beberapa macam langkah berdasarkan penghitungan jumlah baris dalam query SELECT DISTINCT. Salah satu cara untuk melakukan ini adalah memilih nilai unik kedalam tabel lain, kemudian gunakan COUNT(\*) untuk menghitung jumlah baris pada tabel.

Query DISTINCT sering berguna dalam hubungannya dengan fungsi agregat untuk mendapatkan karakterisasi data anda secara lebih komplet. Sebagai contoh, memakai COUNT(\*) pada tabel customer menunjukkan berapa banyak pelanggan yang anda punya, menggunakan DISTINCT pada nilai state dalam tabel memberitahu anda negara bagian yang mana tempat tinggal pelanggan anda, dan COUNT(DISTINCT) pada nilai state memberi tahu anda berapa banyak negara bagian yang mewakili tempat tinggal pelanggan anda.

Ketika digunakan dengan banyak kolom, DISTINCT menunjukkan kombinasi nilai yang berbeda dalam kolom dan COUNT(DISTINCT) menghitung jumlah kombinasi. Query berikut ini menunjukkan pasangan pengirim/penerima yang berbeda pada tabel mail, dan berapa banyak pasangan yang ada :

```
mysql> SELECT DISTINCT srcuser, dstuser FROM mail
-> ORDER BY srcuser, dstuser;
```

srcuser	dstuser
barb	barb
barb	tricia
gene	barb
gene	gene
gene	tricia
phil	barb
phil	phil
phil	tricia
tricia	gene
tricia	phil

```
mysql> SELECT COUNT(DISTINCT srcuser, dstuser) FROM mail;
```

COUNT(DISTINCT srcuser, dstuser)
10

DISTINCT juga bekerja dengan ekspresi tidak hanya nilai kolom. Untuk menentukan jumlah jam pada suatu hari selama pesan pada mail dikirim, hitung nilai HOUR() unik :

```
mysql> SELECT COUNT(DISTINCT HOUR(t)) FROM mail;
```

COUNT(DISTINCT HOUR(t))
12

Untuk mencari jam berapa saja, cetaklah :

```
mysql> SELECT DISTINCT HOUR(t) FROM mail ORDER BY 1;
```

```
+-----+
| HOUR(t) |
+-----+
| 7        |
| 8        |
| 9        |
| 10       |
| 11       |
| 12       |
| 13       |
| 14       |
| 15       |
| 17       |
| 22       |
| 23       |
+-----+
```

Catatan bahwa query ini tidak memberi tahu anda berapa banya pesan yang terkirim setiap jamnya.

## 1.6 Mencari Nilai yang Berhubungan dengan Nilai Minimum dan Maksimum Penyelesaian

Gunakan dua query dan sebuah variable SQL. Atau gunakan trik “**MAX-CONCAT**”. Atau gunakan join.

### Pembahasan

MIN() dan MAX() mencari titik akhir dari suatu nilai jangkaan, tapi kadang-kadang ketika mencari sebuah nilai minimum atau maksimum, anda juga tertarik pada nilai lain dari baris dimana nilai itu berada. Sebagai contoh, anda dapat mencari populasi negara bagian terbesar seperti ini :

```
mysql> SELECT MAX(pop) FROM states;
```

```
+-----+
| MAX(pop) |
+-----+
| 29760021 |
+-----+
```

Tapi ini tidak menunjukkan pada anda negara bagian yang mana populasi ini. Cara yang paling baik untuk mencari tahu adalah seperti berikut ini :

```
mysql> SELECT name, MAX(pop) FROM states WHERE pop = MAX(pop);
ERROR 1111 at line 1: Invalid use of group function
```

Mungkin setiap orang mencoba yang seperti ini cepat atau lambat, tapi ini tidak bekerja, karena fungsi agregat seperti MIN() dan MAX() tidak dapat digunakan dalam klausa WHERE. Maksud dari pernyataan ini adalah untuk menentukan rekord yang mana yang mempunyai nilai populasi maksimum, kemudian menampilkan nama negara bagian yang berhubungan. Masalahnya adalah bahwa ketika anda mengetahui apa yang kita maksudkan dengan menulis semacam itu, ini tidak masuk akal sama sekali pada MySQL. Query gagal karena MySQL menggunakan klausa WHERE untuk menentukan rekord yang mana yang dipilih, tapi dia tahu nilai dari fungsi agregat hanya setelah memilih rekord dari nilai fungsi yang ditentukan. Sehingga pernyataan itu sendiri bertentangan. Anda dapat mengatasi masalah ini menggunakan subselect, kecuali bahwa MySQL

tidak akan mempunyainya hingga versi 4.1. Sementara itu, anda dapat menggunakan pendekatan dua langkah menggunakan satu query yang memilih ukuran maksimum kedalam variable SQL, dan yang mengacu pada variable dalam klausa WHERE nya :

```
mysql> SELECT @max := MAX(pop) FROM states;
mysql> SELECT @max AS 'highest population', name FROM states WHERE pop =
@max;
```

highest population	name
29760021	California

Teknik ini juga bekerja bahkan jika nilai minimum atau maksimum itu sendiri sebenarnya tidak ada dalam baris, tapi hanya diperoleh darinya. Jika anda ingin mengetahui panjang dari ayat terpendek dalam versi Raja James, cara mencarinya adalah :

```
mysql> SELECT MIN(LENGTH(vtext)) FROM kjv;
```

MIN(LENGTH(vtext))
11

Jika anda ingin tanya “Ayat apa itu ?”, lakukan ini :

```
mysql> SELECT @min := MIN(LENGTH(vtext)) FROM kjv;
mysql> SELECT bname, cnum, vnum, vtext FROM kjv WHERE LENGTH(vtext) = @min;
```

bname	cnum	vnum	vtext
John	11	35	Jesus wept.

Teknik lain yang dapat digunakan untuk mencari nilai yang berhubungan dengan nilai minimum atau maksimum dapat ditemukan di MySQL Reference Mannual, dimana ini disebut “**MAX-CONCAT** trick”. Teknik ini melibatkan penambahan sebuah kolom untuk meringkas kolom menggunakan CONCAT(), mencari maksimum dari nilai hasil menggunakan MAX(), dan mengambil bagian nilai yang tidak diringkas dari hasil. Sebagai contoh, untuk mencari nama negara bagian dengan populasi terbesar, anda dapat memilih nilai kombinasi maksimum dari kolom pop dan name, kemudian ambil bagian name dari hasilnya. Paling mudah untuk melakukan langkah demi langkah. Pertama tentukan nilai populasi maksimum untuk mencari seberapa lebar itu :

```
mysql> SELECT MAX(pop) FROM states;
```

MAX(pop)
29760021

Ini adalah delapan karakter. Penting untuk mengetahui hal ini, karena masing-masing kolom dalam nilai kombinasi nama plus populasi harus muncul pada posisi yang tetap sehingga nama negara bagian dapat diambil kemudian. *(Dengan mengambil kolom pop dengan panjang delapan, nilai name akan dimulai pada karakter kesembilan).* Tapi bagaimanapun juga, kita harus hati-hati bagaimana kita mengambil populasi. Nilai yang dihasilkan dari CONCAT() adalah string,

sehingga nilai nama plus populasi akan diperlakukan seperti MAX( ) untuk tujuan pengurutan. Jika kita meratakirkan nilai pop dengan mengambilnya dari kanan dengan RPAD( ), kita akan mendapatkan nilai kombinasi seperti berikut ini :

```
mysql> SELECT CONCAT(RPAD(pop,8,' '),name) FROM states;
+-----+
| CONCAT(RPAD(pop,8,' '),name) |
+-----+
| 4040587 Alabama              |
| 550043 Alaska                |
| 3665228 Arizona              |
| 2350725 Arkansas             |
| ...                           |
```

Nilai diatas akan diurutkan secara leksikal. Ini tidak masalah untuk mencari yang terbesar dari sekumpulan nilai string dengan MAX( ). Tapi nilai pop adalah angka, sehingga kita ingin nilai itu dalam urutan numerik. Untuk membuat urutan leksikal berhubungan dengan urutan numerik, kita harus meratakan nilai populasi dengan mengambil dari kiri dengan LPAD( ) :

```
mysql> SELECT CONCAT(LPAD(pop,8,' '),name) FROM states;
+-----+
| CONCAT(LPAD(pop,8,' '),name) |
+-----+
| 4040587Alabama               |
| 550043Alaska                 |
| 3665228Arizona               |
| 2350725Arkansas              |
| ...                           |
```

Selanjutnya, gunakan ekspresi CONCAT( ) dengan MAX( ) untuk mencari nilai dengan bagian populasi terbesar :

```
mysql> SELECT MAX(CONCAT(LPAD(pop,8,' '),name)) FROM states;
+-----+
| MAX(CONCAT(LPAD(pop,8,' '),name)) |
+-----+
| 29760021California            |
+-----+
```

Untuk mendapatkan hasil akhir (*nama negara bagian yang berhubungan dengan populasi terbesar*), ambil dari nilai kombinasi maksimum substring yang dimulai dari karakter kesembilan :

```
mysql> SELECT SUBSTRING(MAX(CONCAT(LPAD(pop,8,' '),name)),9) FROM states;
+-----+
| SUBSTRING(MAX(CONCAT(LPAD(pop,8,' '),name)),9) |
+-----+
| California                                     |
+-----+
```

Terlihat jelas, menggunakan variable SQL untuk menyimpan hasil antara menjadi lebih mudah. Dalam hal ini, juga lebih efisien karena ini menghindari overhead untuk menyambung nilai kolom untuk mengurutkan dan pemisahan hasil untuk ditampilkan.

Namun cara lain untuk memilih kolom dari baris yang mengandung nilai minimum atau maksimum adalah menggunakan join. Pilih nilai dan masukkan ke tabel lain, kemudian hubungkan tabel itu ke tabel asal untuk memilih baris yang cocok dengan nilainya. Untuk mencari rekord dimana negara bagian dengan populasi terbesar, gunakan join seperti ini :

```
mysql> CREATE TEMPORARY TABLE t
-> SELECT MAX(pop) as maxpop FROM states;
mysql> SELECT states.* FROM states, t WHERE states.pop = t.maxpop;
```

name	abbrev	statehood	pop
California	CA	1850-09-09	29760021

## 1.7 Mengontrol Case Sensitivity String untuk MIN() dan MAX()

### Penyelesaian

Rubah case sensitivity dari string.

### Pembahasan

Ketika diaplikasikan pada nilai string, MIN() dan MAX() menghasilkan hasil yang ditentukan berdasarkan aturan pengurutan leksikal. Satu factor dalam pengurutan string adalah case sensitivitas, sehingga MIN() dan MAX() juga dipengaruhi oleh hal ini. Disini kita menggunakan tabel textblob\_val yang berisi dua kolom dengan nilai yang identik :

```
mysql> SELECT tstr, bstr FROM textblob_val;
```

tstr	bstr
aaa	aaa
AAA	AAA
bbb	bbb
BBB	BBB

Tapi, meskipun nilainya nampak mirip, mereka tidak berperilaku sama. bstr adalah kolom BLOB dan case sensitive. tstr adalah kolom TEXT dan tidak case sensitive. Sebagai hasilnya, MIN() dan MAX() tidak akan menghasilkan hasil yang sama untuk kedua kolom :

```
mysql> SELECT MIN(tstr), MIN(bstr) FROM textblob_val;
```

MIN(tstr)	MIN(bstr)
aaa	AAA

Untuk membuat tstr case sensitive, gunakan BINARY :

```
mysql> SELECT MIN(BINARY tstr) FROM textblob_val;
```

MIN(BINARY tstr)
AAA

Untuk membuat bstr tidak case sensitive, anda dapat mengubah nilai ke case huruf tertentu (*huruf besar atau kecil*) :

```
mysql> SELECT MIN(LOWER(bstr)) FROM textblob_val;
+-----+
| MIN(LOWER(bstr)) |
+-----+
| aaa              |
+-----+
```

Sayangnya, melakukan hal ini juga mengubah nilai yang ditampilkan. Jika hal ini tidak masalah, gunakan teknik berikut ini (*dan lihat bahwa ini akan menghasilkan hasil yang berbeda*) :

```
mysql> SELECT @min := MIN(LOWER(bstr)) FROM textblob_val;
mysql> SELECT bstr FROM textblob_val WHERE LOWER(bstr) = @min;
+-----+
| bstr |
+-----+
| aaa  |
| AAA  |
+-----+
```

## 1.8 Membagi sebuah Ringkasan menjadi Subkelompok Penyelesaian

Gunakan klausa GROUP BY untuk mengatur baris menjadi kelompok-kelompok.

### Pembahasan

Query ringkasan yang ditunjukkan selama ini menghitung nilai ringkasan seluruh baris dalam kumpulan hasil. Sebagai contoh, query berikut ini menentukan jumlah rekord perjalanan harian dalam tabel driver\_log, dan oleh karena itu jumlah total hari dimana pengemudi di jalan adalah :

```
mysql> SELECT COUNT(*) FROM driver_log;
+-----+
| COUNT(*) |
+-----+
| 10       |
+-----+
```

Tapi kadang-kadang diinginkan untuk memisahkan kumpulan baris menjadi subkelompok dan meringkas masing-masing kelompok. Ini dikerjakan menggunakan fungsi agregat dalam hubungannya dengan klausa GROUP BY. Untuk menentukan jumlah hari dimana pengemudi bekerja, kelompokkan baris berdasarkan nama pengemudi, hitung berapa banyak baris untuk setiap nama dan tampilkan nama-nama itu dengan jumlahnya :

```
mysql> SELECT name, COUNT(name) FROM driver_log GROUP BY name;
+-----+-----+
| name | COUNT(name) |
+-----+-----+
| Ben  | 3           |
| Henry | 5           |
| Suzi | 2           |
+-----+-----+
```

Query diatas meringkas kolom yang sama yang digunakan untuk pengelompokan (*name*), tapi ini tidak selalu penting. Misalkan anda ingin pencirian yang cepat dari tabel driver\_log, yang

menunjukkan masing-masing orang yang terdaftar dalam tabel, total jarak dan rata-rata jumlah mil per hari. Dalam hal ini, anda tetap menggunakan kolom name untuk menempatkan baris dalam kelompok, tapi fungsi ringkasan beroperasi pada nilai miles :

```
mysql> SELECT name,
-> SUM(miles) AS 'total miles',
-> AVG(miles) AS 'miles per day'
-> FROM driver_log GROUP BY name;
```

name	total miles	miles per day
Ben	362	120.6667
Henry	911	182.2000
Suzi	893	446.5000

Gunakan kolom pengelompokan sebanyak mungkin untuk mencapai ringkasan sehalus yang anda inginkan. Query berikut ini menghasilkan ringkasan kasar yang menunjukkan berapa banyak pesan yang dikirimkan oleh masing-masing pengirim pesan yang terdaftar pada tabel mail :

```
mysql> SELECT srcuser, COUNT(*) FROM mail
-> GROUP BY srcuser;
```

srcuser	COUNT(*)
barb	3
gene	6
phil	5
tricia	2

Untuk lebih spesifik dan mencari berapa banyak pesan yang dikirimkan oleh masing-masing pengirim dari tiap-tiap host, gunakan dua kolom pengelompokan. Ini akan menghasilkan sebuah hasil dengan kelompok bersarang (*kelompok dalam kelompok*) :

```
mysql> SELECT srcuser, srchost, COUNT(*) FROM mail
-> GROUP BY srcuser, srchost;
```

srcuser	srchost	COUNT(*)
barb	saturn	2
barb	venus	1
gene	mars	2
gene	saturn	2
gene	venus	2
phil	mars	3
phil	venus	2
tricia	mars	1
tricia	saturn	1

## 1.9 Mendapatkan Nilai Distinct Tanpa Menggunakan DISTINCT

Jika anda menggunakan GROUP BY tanpa memilih nilai dari sembarang fungsi agregat, anda akan mendapatkan efek yang sama seperti DISTINCT tanpa menggunakan DISTINCT secara eksplisit :

```
mysql> SELECT name FROM driver_log GROUP BY name;
```

```
+-----+
| name  |
+-----+
| Ben   |
| Henry |
| Suzi  |
+-----+
```

Biasanya dengan query semacam ini anda akan memilih sebuah nilai ringkasan (misalnya, dengan mengambil COUNT(name) untuk menghitung rekord dari masing-masing nama), tapi sebaiknya ini tidak dilakukan. Efeknya adalah untuk menghasilkan daftar nilai pengelompokan yang unik. Saya lebih suka menggunakan DISTINCT, karena ini akan membuat querynya lebih jelas. (Secara internal, MySQL memetakan bentuk DISTINCT dari query ke bentuk GROUP BY). Contoh didepan telah menggunakan COUNT(), SUM() dan AVG() untuk ringkasan per kelompok. Anda dapat juga menggunakan MIN() atau MAX(). Dengan klausa GROUP BY, mereka akan memberi tahu anda nilai terkecil atau terbesar per kelompok. Query berikut ini mengelompokkan baris-baris tabel mail berdasarkan pengirim pesan, menampilkan pesan terbesar yang pernah dikirim dan tanggal kirim dari pesan yang terakhir :

```
mysql> SELECT srcuser, MAX(size), MAX(t) FROM mail GROUP BY srcuser;
```

```
+-----+-----+-----+
| srcuser | MAX(size) | MAX(t) |
+-----+-----+-----+
| barb    | 98151     | 2001-05-14 14:42:21 |
| gene    | 998532    | 2001-05-19 22:21:51 |
| phil    | 10294     | 2001-05-17 12:49:23 |
| tricia  | 2394482   | 2001-05-14 17:03:01 |
+-----+-----+-----+
```

Anda dapat mengelompokkan berdasarkan banyak kolom dan menampilkan maksimum dari tiap nilai kombinasi dalam kolom-kolom itu. Query ini mencari ukuran pesan terbesar yang dikirim antara pasangan pengirim dan penerima yang terdaftar pada tabel mail :

```
mysql> SELECT srcuser, dstuser, MAX(size) FROM mail GROUP BY srcuser,
dstuser;
```

```
+-----+-----+-----+
| srcuser | dstuser | MAX(size) |
+-----+-----+-----+
| barb    | barb    | 98151     |
| barb    | tricia  | 58274     |
| gene    | barb    | 2291      |
| gene    | gene    | 23992     |
| gene    | tricia  | 998532    |
| phil    | barb    | 10294     |
| phil    | phil    | 1048      |
| phil    | tricia  | 5781      |
| tricia  | gene    | 194925    |
| tricia  | phil    | 2394482   |
+-----+-----+-----+
```

Ketika menggunakan fungsi agregat untuk menghasilkan nilai ringkasan per kelompok, hati-hati dengan perangkat berikut ini. Misalkan anda ingin mengetahui perjalanan terjauh per pengemudi dalam tabel driver\_log. Ini dapat dihasilkan dengan query berikut ini :

```
mysql> SELECT name, MAX(miles) AS 'longest trip'
```



```
-> FROM driver_log GROUP BY name;
```

name	longest trip
Ben	152
Henry	300
Suzi	502

Tapi bagaimana jika anda juga ingin menunjukkan tanggal terjadinya perjalanan terpanjang dari masing-masing pengemudi? Dapatkah anda hanya menambahkan trav\_date ke daftar kolom keluaran ? Ini tidak akan bekerja :

```
mysql> SELECT name, trav_date, MAX(miles) AS 'longest trip'
```

```
-> FROM driver_log GROUP BY name;
```

name	trav_date	longest trip
Ben	2001-11-30	152
Henry	2001-11-29	300
Suzi	2001-11-29	502

Query ini menghasilkan sebuah hasil, tapi jika anda membandingkannya dengan tabel lengkap (*nampak dibawah*), anda akan melihat bahwa meskipun tanggal untuk Ben dan Henry benar, tanggal untuk Suzi salah :

rec_id	name	trav_date	miles	
1	Ben	2001-11-30	152	<-- Ben's longest trip
2	Suzi	2001-11-29	391	
3	Henry	2001-11-29	300	<-- Henry's longest trip
4	Henry	2001-11-27	96	
5	Ben	2001-11-29	131	
6	Henry	2001-11-26	115	
7	Suzi	2001-12-02	502	<-- Suzi's longest trip
8	Henry	2001-12-01	197	
9	Ben	2001-12-02	79	
10	Henry	2001-11-30	203	

Jadi apa yang terjadi ? Kenapa ringkasan query menghasilkan hasil yang salah ? Ini terjadi karena ketika anda memasukkan klausa GROUP BY dalam sebuah query, satu-satunya nilai yang dapat anda pilih adalah kolom-kolom yang dikelompokkan atau nilai-nilai ringkasan yang dihitung dari nilai-nilai tersebut. Jika anda menampilkan kolom tambahan, mereka tidak terikat pada kolom-kolom yang dikelompokkan dan nilai-nilai yang ditampilkan tidak dapat ditentukan. (*Untuk query yang baru saja ditunjukkan, nampak bahwa MySQL bisa mengambil tanggal pertama dari masing-masing pengemudi, apakah ini cocok atau tidak dengan nilai jarak maksimum dari pengemudi itu*). Penyelesaian umum terhadap masalah menampilkan isi baris yang berhubungan dengan nilai kelompok minimum atau maksimum mengikutsertakan join. Jika anda tidak ingin membaca terlebih dulu, atau anda tidak ingin menggunakan tabel lain, pertimbangkan menggunakan trik MAX-CONCAT yang dibahas dimuka. Ini akan menghasilkan hasil yang benar, meskipun querynya nampak jelek :

```
mysql> SELECT name,
```

```
-> SUBSTRING(MAX(CONCAT(LPAD(miles,3,' '), trav_date)),4) AS date,
-> LEFT(MAX(CONCAT(LPAD(miles,3,' '), trav_date)),3) AS 'longest trip'
-> FROM driver_log GROUP BY name;
```

name	date	longest trip
Ben	2001-11-30	152
Henry	2001-11-29	300
Suzi	2001-12-02	502

## 1.10. Nilai Ringkasan dan Nilai NULL

### Penyelesaian

Pahami bagaimana fungsi agregat menangani nilai NULL.

### Pembahasan

Sebagian besar fungsi agregat mengabaikan nilai NULL. Misalkan anda mempunyai tabel expt yang mencatat hasil eksperimen untuk subyek yang masing-masing harus diberi empat tes dan yang mendaftarkan nilai tes sebagai NULL untuk tes yang belum terdaftar :

```
mysql> SELECT subject, test, score FROM expt ORDER BY subject, test;
```

subject	test	score
Jane	A	47
Jane	B	50
Jane	C	NULL
Jane	D	NULL
Marvin	A	52
Marvin	B	45
Marvin	C	53
Marvin	D	NULL

Dengan menggunakan klausa GROUP BY untuk mengatur baris berdasarkan nama subyek, jumlah tes yang diambil oleh masing-masing subyek, demikian juga total, rata-rata, nilai terendah dan tertinggi dapat dihitung seperti ini :

```
mysql> SELECT subject,
-> COUNT(score) AS n,
-> SUM(score) AS total,
-> AVG(score) AS average,
-> MIN(score) AS lowest,
-> MAX(score) AS highest
-> FROM expt GROUP BY subject;
```

subject	n	total	average	lowest	highest
Jane	2	97	48.5000	47	50
Marvin	3	150	50.0000	45	53

Anda dapat melihat dari hasilnya dalam kolom berlabel n (*jumlah tes*) yang menghitung hanya lima nilai. Kenapa ? Karena nilai dalam kolom itu berhubungan dengan banyaknya nilai tes

bukan NULL untuk masing-masing subyek. Kolom ringkasan lainnya menampilkan hasil yang dihitung juga hanya dari nilai bukan NULL.

Masuk akal bagi fungsi agregat untuk mengabaikan nilai NULL. Jika mereka mengikuti aturan aritmetika SQL umumnya, menambahkan NULL ke sembarang nilai lainnya akan menghasilkan NULL. Ini akan membuat fungsi agregat sangat sulit untuk digunakan karena anda harus menyaring nilai NULL setiap kali anda membuat ringkasan untuk menghindari mendapatkan hasil NULL. Dengan mengabaikan nilai NULL, fungsi agregat menjadi lebih nyaman. Tapi, hati-hati bahwa meskipun fungsi agregat bisa mengabaikan nilai NULL, beberapa masih dapat menghasilkan NULL sebagai hasil. Ini terjadi jika tidak ada lagi yang harus diringkas. Query berikut ini sama dengan yang terdahulu, dengan sedikit perbedaan. Ini memilih hanya nilai tes NULL, sehingga tidak ada yang bisa dioperasikan oleh fungsi agregat :

```
mysql> SELECT subject,
-> COUNT(score) AS n,
-> SUM(score) AS total,
-> AVG(score) AS average,
-> MIN(score) AS lowest,
-> MAX(score) AS highest
-> FROM expt WHERE score IS NULL GROUP BY subject;
```

subject	n	total	average	lowest	highest
Jane	0	0	NULL	NULL	NULL
Marvin	0	0	NULL	NULL	NULL

Bahkan dalam keadaan seperti ini, fungsi ringkasan masih mengembalikan nilai yang paling dibutuhkan. Jumlah tes dan total nilai per subyek adalah nol dan ditampilkan dengan cara ini. AVG(), dilain pihak, mengembalikan NULL. Rata-rata adalah rasio, yang dihitung sebagai jumlah nilai dibagi jumlah nilai. Ketika tidak ada lagi nilai yang diringkas, rasionya adalah 0/0, yang tidak terdefinisikan. Oleh karena itu NULL adalah hasil yang paling masuk akal untuk dikembalikan oleh AVG(). Demikian juga dengan MIN() dan MAX() yang tidak harus mengerjakan apa-apa, sehingga mereka mengembalikan NULL. Jika anda tidak ingin fungsi-fungsi ini menghasilkan NULL dalam keluaran query, gunakan IFNULL() untuk memetakan hasilnya secara benar :

```
mysql> SELECT subject,
-> COUNT(score) AS n,
-> SUM(score) AS total,
-> IFNULL(AVG(score),0) AS average,
-> IFNULL(MIN(score),'Unknown') AS lowest,
-> IFNULL(MAX(score),'Unknown') AS highest
-> FROM expt WHERE score IS NULL GROUP BY subject;
```

subject	n	total	average	lowest	highest
Jane	0	0	0	Unknown	Unknown
Marvin	0	0	0	Unknown	Unknown

COUNT() berbeda dalam hal nilai NULL dari pada fungsi agregat lainnya. Seperti fungsi agregat, COUNT(ekspresi) menghitung hanya nilai bukan NULL, tapi COUNT(\*) menghitung baris,

tanpa memandang isinya. Anda dapat melihat perbedaan antara bentuk-bentuk COUNT ( ) seperti ini :

```
mysql> SELECT COUNT(*), COUNT(score) FROM expt;
+-----+-----+
| COUNT(*) | COUNT(score) |
+-----+-----+
| 8        | 5            |
+-----+-----+
```

Ini memberitahu kita bahwa ada delapan baris dalam tabel expt tapi hanya lima diantaranya yang mempunyai nilai score. Perbedaan bentuk-bentuk COUNT ( ) dapat sangat berguna untuk menghitung nilai yang hilang; kita cari selisihnya :

```
mysql> SELECT COUNT(*) - COUNT(score) AS missing FROM expt;
+-----+
| missing |
+-----+
| 3        |
+-----+
```

Penghitungan yang hilang dan yang tidak hilang dapat juga ditentukan untuk subkelompok. Query berikut ini melakukannya untuk masing-masing subyek. Ini menyediakan cara cepat untuk memerkirakan sejauh mana eksperimen telah diselesaikan :

```
mysql> SELECT subject,
-> COUNT(*) AS total,
-> COUNT(score) AS 'non-missing',
-> COUNT(*) - COUNT(score) AS missing
-> FROM expt GROUP BY subject;
+-----+-----+-----+-----+
| subject | total | non-missing | missing |
+-----+-----+-----+-----+
| Jane    | 4     | 2           | 2       |
| Marvin    | 4     | 3           | 1       |
+-----+-----+-----+-----+
```

## 1.11. Memilih Hanya Kelompok dengan Ciri Tertentu

### Penyelesaian

Gunakan klausa HAVING.

### Pembahasan

Anda mengenal penggunaan WHERE untuk menentukan kondisi yang harus dipenuhi oleh tiap-tiap rekord supaya dipilih oleh query. Ini alamiah, oleh karena itu, untuk menggunakan WHERE untuk menulis kondisi yang melibatkan nilai ringkasan. Satu-satunya masalah adalah bahwa ini tidak akan bekerja. Jika anda ingin mengidentifikasi pengemudi dalam tabel driver\_log yang mengemudi lebih dari tiga hari, anda akan berpikir untuk menulis query seperti ini :

```
mysql> SELECT COUNT(*), name
-> FROM driver_log
-> WHERE COUNT(*) > 3
-> GROUP BY name;
ERROR 1111 at line 1: Invalid use of group function
```

Masalahnya disini adalah bahwa WHERE menentukan batasan awal yang menentukan baris yang mana yang harus dipilih, tapi nilai dari COUNT ( ) dapat di tentukan hanya setelah baris-barisnya telah dipilih. Penyelesaiannya adalah untuk meletakkan ekspresi COUNT ( ) dalam klausa HAVING. HAVING adalah analog dengan WHERE, tapi ini diaplikasikan ke ciri kelompok dari pada ke rekord tunggal. Yaitu, HAVING beroperasi pada kumpulan baris yang telah dipilih dan dikelompokkan, mengaplikasikan batasan tambahan berdasarkan pada hasil fungsi agregat yang tidak diketahui selama proses pemilihan awal. Oleh karena itu query didepan harusnya ditulis seperti ini :

```
mysql> SELECT COUNT(*), name
-> FROM driver_log
-> GROUP BY name
-> HAVING COUNT(*) > 3;
+-----+-----+
| COUNT(*) | name |
+-----+-----+
| 5        | Henry |
+-----+-----+
```

Ketika anda menggunakan HAVING, anda masih dapat memasukkan klausa WHERE tapi hanya untuk memilih baris, bukan untuk mengecek nilai ringkasan. HAVING dapat mengacu ke alias, sehingga query sebelumnya dapat ditulis seperti ini :

```
mysql> SELECT COUNT(*) AS count, name
-> FROM driver_log
-> GROUP BY name
-> HAVING count > 3;
+-----+-----+
| count | name |
+-----+-----+
| 5      | Henry |
+-----+-----+
```

## 1.12 Menentukan Apakah Nilainya Unik

### Penyelesaian

Gunakan HAVING dalam hubungannya dengan COUNT ( )

### Pembahasan

Anda dapat menggunakan HAVING untuk mencari nilai yang unik dalam situasi dimana DISTINCT tidak bisa digunakan. DISTINCT ( ) menghilangkan duplikasi, tapi tidak menunjukkan nilai yang mana yang sebenarnya diduplikasikan dari data asli. HAVING dapat memberitahu anda nilai yang mana yang unik atau tidak unik. Query berikut ini menunjukkan hari-hari dimana hanya satu pengemudi yang aktif, dan hari-hari dimana lebih dari satu pengemudi yang aktif. Ini didasarkan pada penggunaan HAVING dan COUNT ( ) untuk menentukan nilai trav\_date yang mana yang unik atau tidak unik :

```
mysql> SELECT trav_date, COUNT(trav_date)
-> FROM driver_log
-> GROUP BY trav_date
-> HAVING COUNT(trav_date) = 1;
+-----+-----+
| trav_date | COUNT(trav_date) |
+-----+-----+
```

2001-11-26	1
2001-11-27	1
2001-12-01	1

```
mysql> SELECT trav_date, COUNT(trav_date)
-> FROM driver_log
-> GROUP BY trav_date
-> HAVING COUNT(trav_date) > 1;
```

trav_date	COUNT(trav_date)
2001-11-29	3
2001-11-30	2
2001-12-02	2

Teknik ini juga bekerja untuk kombinasi nilai. Sebagai contoh, untuk mencari pesan pasangan pengirim/penerima antara siapa yang hanya mengirim satu pesan, mencari kombinasi yang hanya terjadi sekali dalam table mail :

```
mysql> SELECT srcuser, dstuser
-> FROM mail
-> GROUP BY srcuser, dstuser
-> HAVING COUNT(*) = 1;
```

srcuser	dstuser
barb	barb
gene	tricia
phil	barb
tricia	gene
tricia	phil

Perlu dicatat bahwa query ini tidak mencetak hasil perhitungan. Dua contoh pertama melakukan hal ini, untuk memperlihatkan bahwa hasil perhitungan digunakan dengan benar, tapi anda dapat menggunakan perhitungan dalam klausa HAVING tanpa mengikutsertakannya dalam daftar kolom keluaran.

### 1.13 Pengelompokan Hasil Ekspresi Penyelesaian

Letakkan ekspresi dalam klausa GROUP BY. Untuk MySQL versi lama yang tidak mendukung ekspresi GROUP BY, gunakan yang lainnya.

#### Pembahasan

GROUP BY berbagi property dengan GROUP BY yang seperti pada MySQL 3.23.2 yang dapat mengacu ke ekspresi. Ini berarti anda dapat menggunakan perhitungan sebagai dasar untuk pengelompokan. Sebagai contoh, untuk mencari distribusi panjang nama Negara bagian, GROUP BY LENGTH(name) :

```
mysql> SELECT LENGTH(name), COUNT(*)
-> FROM states GROUP BY LENGTH(name);
```

LENGTH(name)	COUNT(*)
4	3
5	3
6	5
7	8
8	12
9	4
10	4
11	2
12	4
13	3
14	2

Sebelum MySQL 3.23.2, anda tidak dapat menggunakan ekspresi dalam klausa ORDER BY, sehingga query diatas akan gagal, cara lain untuk mengatasi masalah ini diberikan dalam hubungannya dengan ORDER BY, dan metoda yang sama bisa bekerja dengan ORDER BY. Satu cara adalah dengan memberi nama alias pada ekspresi dalam daftar keluaran kolom dan mengacu ke alias dalam klausa ORDER BY:

```
mysql> SELECT LENGTH(name) AS len, COUNT(*)
-> FROM states GROUP BY len;
```

len	COUNT(*)
4	3
5	3
6	5
7	8
8	12
9	4
10	4
11	2
12	4
13	3
14	2

Cara lain adalah dengan menulis klausa ORDER BY untuk mengacu ke posisi kolom keluaran :

```
mysql> SELECT LENGTH(name), COUNT(*)
-> FROM states GROUP BY 1;
```

LENGTH(name)	COUNT(*)
4	3
5	3
6	5
7	8
8	12
9	4
10	4
11	2
12	4
13	3
14	2

Tentu saja, cara alternatif penulisan query ini bekerja pada MySQL 3.23.2 dan selanjutnya dan anda bisa menemukannya lebih dapat dibaca. Anda dapat mengelompokkan berdasar banyak ekspresi. Untuk mencari hari-hari dalam satu tahun dimana lebih dari satu Negara bagian bergabung ke AS, kelompokkan bulan dan hari statehood, kemudian gunakan HAVING dan COUNT ( ) untuk mencari kombinasi tidak unik :

```
mysql> SELECT MONTHNAME(statehood), DAYOFMONTH(statehood), COUNT(*)
-> FROM states GROUP BY 1, 2 HAVING COUNT(*) > 1;
```

MONTHNAME(statehood)	DAYOFMONTH(statehood)	COUNT(*)
February	14	2
June	1	2
March	1	2
May	29	2
November	2	2

### Referensi:

Paul DuBois, *"MySQL Cookbook"*, O'Reilly - Oktober 2002

### BIOGRAFI PENULIS



**Janner Simarmata.** Lahir di Aek Nabara, 07 Januari 1976. Tamat dari STM GKPS Pematang Siantar tahun 1995. Menyelesaikan program S1 pada jurusan Teknik Informatika di STMIK BANDUNG pada tahun 2000. Pernah mengajar di beberapa Perguruan Tinggi Swasta seperti: STMIK Mikroskil, STMIK Multimedia Prima, Unika Santo Thomas Sumatera Utara. Pada tahun 2004 melanjutkan studi pada program S2 (M.Kom) pada jurusan Ilmu Komputer Universitas Gadjadara sampai sekarang.

Informasi lebih lanjut tentang penulis:

KEYWORD: *Janner Simarmata*

Email: *sijanner@yahoo.com*