

BAB 10

VIEW TABLE & TEMPORARY TABLE

10.1. Tujuan Pembelajaran

1. Membuat view table dan temp table
2. Mengetahui cara kerja view table
3. Mengetahui kegunaan view table

10.2. Dasar Teori

Pengenalan View

Table

Pernyataan CREATE VIEW membuat tampilan baru dalam database. Berikut adalah sintaks dasar dari pernyataan CREATE VIEW:

```
CREATE [OR REPLACE] VIEW [db_name.]view_name [(column_list)]
AS
select-statement;
```

Gambar 10.1 Sintak Dasar View Table

Dalam sintaks ini:

Pertama, tentukan nama view yang ingin dibuat setelah kata kunci CREATE VIEW. Nama tampilan unik dalam database. Karena tampilan dan tabel dalam database yang sama berbagi ruang nama yang sama, nama tampilan tidak boleh sama dengan nama tabel yang sudah ada.

Kedua, gunakan opsi OR REPLACE jika ingin mengganti tampilan yang ada jika tampilan tersebut sudah ada. Jika tampilan tidak ada, OR REPLACE tidak berpengaruh.

Ketiga, tentukan daftar kolom untuk tampilan. Secara default, kolom tampilan berasal dari daftar pilih pernyataan SELECT. Namun, kalian bisa secara eksplisit menentukan daftar kolom untuk tampilan dengan mencantumkanannya dalam kurung mengikuti nama tampilan.

Terakhir, tentukan pernyataan SELECT yang mendefinisikan tampilan. Pernyataan SELECT dapat meminta data dari tabel atau tampilan. MySQL memungkinkan kalian untuk menggunakan klausa ORDER BY dalam pernyataan SELECT tetapi mengabaikannya jika kalian memilih dari tampilan dengan kueri yang memiliki klausa ORDER BY sendiri.

Secara default, pernyataan CREATE VIEW membuat tampilan di database saat ini. Jika kalian ingin secara eksplisit membuat tampilan dalam database tertentu, kalian bisa mengkualifikasikan nama tampilan dengan nama database.

Pengenalan Temporary Table

Di MySQL, tabel sementara adalah jenis tabel khusus yang memungkinkan kalian

menyimpan kumpulan hasil sementara, yang dapat digunakan kembali beberapa kali dalam satu sesi.

Tabel sementara sangat berguna saat tidak mungkin atau mahal untuk meminta data yang memerlukan pernyataan SELECT tunggal dengan klausa JOIN. Dalam hal ini, kalian dapat menggunakan tabel sementara untuk menyimpan hasil langsung dan menggunakan kueri lain untuk memprosesnya.

Tabel sementara MySQL memiliki fitur khusus berikut:

1. Tabel sementara dibuat dengan menggunakan pernyataan CREATE TEMPORARY TABLE. Perhatikan bahwa kata kunci TEMPORARY ditambahkan di antara kata kunci CREATE dan TABLE.
2. MySQL menghapus tabel sementara secara otomatis saat sesi berakhir atau koneksi diakhiri. Tentu saja, kalian dapat menggunakan pernyataan DROP TABLE untuk menghapus tabel sementara secara eksplisit saat kalian tidak lagi menggunakannya.
3. Tabel sementara hanya tersedia dan dapat diakses oleh klien yang membuatnya. Klien yang berbeda dapat membuat tabel sementara dengan nama yang sama tanpa menyebabkan kesalahan karena hanya klien yang membuat tabel sementara yang dapat melihatnya. Namun, dalam sesi yang sama, dua tabel sementara tidak dapat menggunakan nama yang sama.
4. Tabel sementara dapat memiliki nama yang sama dengan tabel normal dalam database. Misalnya, jika kalian membuat tabel sementara bernama karyawan di database sampel, tabel karyawan yang sudah ada menjadi tidak dapat diakses. Setiap kueri yang kalian ajukan terhadap tabel karyawan sekarang merujuk ke tabel sementara karyawan. Saat kalian menjatuhkan tabel karyawan sementara, tabel karyawan permanen tersedia dan dapat diakses.

Sintaks pernyataan CREATE TEMPORARY TABLE mirip dengan sintaks pernyataan CREATETABLE kecuali untuk kata kunci TEMPORARY:

```
CREATE TEMPORARY TABLE table_name(  
    column_1_definition,  
    column_2_definition,  
    ...,  
    table_constraints  
);
```

Gambar 10.2 Sintak Dasar Temporary Table

Untuk membuat tabel sementara yang strukturnya didasarkan pada tabel yang sudah ada, kalian tidak dapat menggunakan pernyataan `CREATE TEMPORARY TABLE ... LIKE`. Sebagai gantinya, kalian menggunakan sintaks berikut:

```
CREATE TEMPORARY TABLE temp_table_name
SELECT * FROM original_table
LIMIT 0;
```

Gambar 10.3 Sintak Dasar Temporary Table Menggunakan Limit


10.3. Software

- XAMPP
- MySQL

10.4. Pembelajaran View Table & Temporary Table

10.4.1 Pembelajaran View Table

Sebelum membuat view, buatlah terlebih dahulu tabel baru dengan nama `orderDetails`. Untuk strukturnya seperti gambar dibawah ini.

Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan
orderNumber 	int(5)			Tidak	Tidak ada
productCode	varchar(5)	utf8mb4_general_ci		Ya	NULL
quantityOrdered	tinyint(3)			Ya	NULL
priceEach	int(5)			Ya	NULL

Gambar 10.4 Struktur View Table OrderDetails

Membuat Tabel View

Setelah berhasil membuat tabel baru, silahkan ketikkan query untuk membuat view tabel seperti gambar dibawah ini.

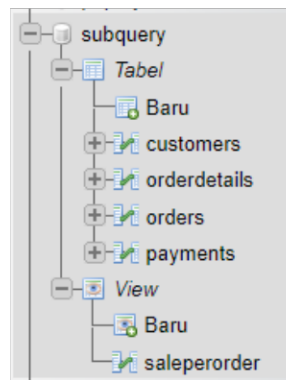
✓ MySQL memberikan hasil kosong (atau nol baris). (Pencarian dila

```
1 CREATE VIEW salePerOrder AS
2   SELECT
3     orderNumber,
4     SUM(quantityOrdered * priceEach) total
5   FROM orderdetails
6   GROUP BY orderdetails.orderNumber
7   ORDER BY total DESC;
```

Gambar 10.5 Query View Table SalePerOrder

Query diatas digunakan untuk membuat view tabel, untuk penjelasan nya seperti dibawah ini.

1. CREATE VIEW salePerOrder AS.....total, artinya kita memberitahu SQL untuk **membuat view tabel dengan nama salePerOrder** dengan nama lain total.
2. SELECT orderNumber, SUM(quantityOrdered * priceEach), artinya kita memberitahuSQL untuk menampilkan kolom orderNumber dan penjumlahan hasil kali antara quantityOrdered dengan priceEach.
3. FROM orderdetails, artinya kita memberitahu SQL bahwa kolom yang kita sebutkandiatas berasal dari tabel orderDetails.
4. GROUP BY orderdetails.orderNumber, artinya kita ingin pernyataan mengelompokkanbaris yang memiliki nilai yang sama ke dalam baris ringkasan.
5. ORDER BY total DESC, artinya kita menggunakan untuk mengurutkan result-set daribesar ke kecil.



Gambar 10.6 Melihat View Table

Jika kalian menggunakan perintah SHOW TABLE untuk melihat semua tabel di database subquery, kalian akan melihat view salePerOrder muncul di daftar.

Jika kalian ingin menanyakan total penjualan untuk setiap pesanan penjualan, kalian hanya perlu menjalankan pernyataan SELECT sederhana terhadap tampilan SalePerOrder sebagaiberikut:

```
SELECT * FROM saleperorder;
```

Gambar 10.7 Query Menampilkan View Table SalePerOrder

Outputnya akan menampilkan:

orderNumber	total
55555	3000000
11111	1500000
22222	800000
33333	500000
44444	500000

Gambar 10.8 Tampilan View Table SalePerOrder

Hasilnya, kita akan mendapat sebuah tabel dengan nama salePerOrder yang dimana menampilkan orderNumber dan penjumlahan dari perkalian quantityOrder dikali dengan priceEach. Misalnya kita hitung kuantitas dan harga yang ada di baris pertama dari tabel orderDetails.

orderNumber	productCode	quantityOrdered	priceEach
11111	ad231	30	50000
22222	fkj45	20	40000
33333	oqh21	50	10000
44444	oiw85	25	20000
55555	pia43	50	60000

Gambar 10.9 Tampilan Tabel OrderDetails

Jika kita kalikan 30 dan 50000, maka kita akan mendapat hasil 1500000. Karena kita melakukan sorting secara desc, yang akan menampilkan hasil yang terbesar lalu diurutkansampai terkecil. Maka hasil 1500000 akan berada di baris kedua

Membuat Tabel View Dari Tabel View Yang Sudah Ada

MySQL memungkinkan kalian membuat tampilan berdasarkan tampilan lain. Misalnya, kalian bisa membuat tampilan yang disebut `bigSalesOrder` berdasarkan tampilan `salePerOrder` untuk memperlihatkan setiap pesanan penjualan yang totalnya lebih besar dari 500.000 sebagai berikut:

```
✓ MySQL memberikan hasil kosong (atau nol)

1 CREATE VIEW bigSalesOrder AS
2   SELECT
3     orderNumber,
4     ROUND(total) as total
5   FROM
6     salePerOrder
7   WHERE
8     total > 500000;
```

Gambar 10.10 Query View Table `BigSalesOrder`

Sekarang, kalian dapat mengkueri data dari tampilan `bigSalesOrder` sebagai berikut:

```
SELECT orderNumber, total FROM bigSalesOrder;
```

Gambar 10.11 Query Menampilkan `OrderNumber` dan `Total` dari `BigSalesOrder`

Outputnya akan menampilkan:

orderNumber	total
55555	3000000
11111	1500000
22222	800000

Gambar 10.12 Tampilan `OrderNumber` dan `Total` dari `BigSalesOrder`

Kenapa outputnya hanya 3, bukannya 5 record? Karena kita ingin menampilkan `orderNumber` dan `total` jika **hasilnya lebih dari 500000**, karena 2 record tidak memenuhi kondisi tersebut maka tidak akan ditampilkan.

Membuat Tabel View Dengan Subquery

Contoh berikut menggunakan pernyataan `CREATE VIEW` untuk membuat tampilan yang pernyataan `SELECT`-nya menggunakan subquery. Tampilan berisi produk yang harga belinya lebih tinggi dari harga rata-rata semua produk.

✓ MySQL memberikan hasil kosong (atau nol)

```

1 CREATE VIEW aboveAvgProducts AS
2     SELECT
3         productCode,
4         quantityOrdered,
5         priceEach
6     FROM
7         orderdetails
8     WHERE
9         priceEach > (
10            SELECT
11                AVG(priceEach)
12            FROM
13                orderdetails)
14     ORDER BY priceEach DESC;

```

Gambar 10.13 Query View Table AboveAvgProduct

Query diatas digunakan untuk membuat view tabel dengan menggunakan subquery, bilamasih bingung silahkan ditanyakan.

Data kueri dari Produk Rata-Rata di atas ini sederhana sebagai berikut:

```
SELECT * FROM aboveAvgProducts;
```

Gambar 10.14 Query Menampilkan View Table AboveAvgProduct

Outputnya akan menampilkan:

productCode	quantityOrdered	priceEach
pia43	50	60000
ad231	30	50000
fkj45	20	40000

Gambar 10.15 Tampilan View Table AboveAvgProduct

Misalnya, kita ingin melihat ada atau tidaknya perubahan tabel orderDetails dengan mengubah isi data atau record yang ada di tabel orderDetails. Bisa menggunakan queryseperti ini.

```
UPDATE `orderdetails` SET `quantityOrdered` = '50' WHERE `orderdetails`.`orderNumber` = 11111;
```

Gambar 10.16 Query Update QuantityOrdered

Query diatas kita gunakan hanya untuk mengubah nilai quantityOrdered dengan nilai baru = 50 dengan orderNumber = 11111.

Outputnya akan menampilkan seperti ini di tabel orderDetails:

orderNumber	productCode	quantityOrdered	priceEach
11111	ad231	50	50000
22222	fkj45	20	40000
33333	oqh21	50	10000
44444	oiw85	25	20000
55555	pia43	50	60000

Gambar 10.17 Menampilkan Tabel OrderDetails

Di baris pertama terlihat, kalau query yang kita buat sudah berhasil melakukan pembaruan quantityOrdered. **Apakah mengubah view tabelnya?** Iya, karena record yang kita masukkan berbeda. Jadi data akan otomatis diperbarui atau di *update*.

Coba lihat output view tabel yang datanya sudah di update, maka akan menampilkan output seperti dibawah ini:

orderNumber	total
55555	3000000
11111	2500000
22222	800000
33333	500000
44444	500000

Gambar 10.18 Menampilkan Isi View Table Terbaru

Kalau belum menemukan perbedaannya, silahkan lihat perbandingan tabel view salePerOrder dibawah ini.

Sebelum di update

orderNumber	total
55555	3000000
11111	1500000
22222	800000
33333	500000

Gambar 10.19 Menampilkan Isi View Table Sebelum Diupdate

Sesudah di update

orderNumber	total
55555	3000000
11111	2500000
22222	800000
33333	500000
44444	500000

Gambar 10.20 Menampilkan Isi View Table Sesudah Diupdate

10.4.2 Pembelajaran Temporary Table

Sesuai dengan namanya, temporary table digunakan untuk membuat tabel sementara didalam MySQL. Tabel ini hanya ada untuk sementara waktu, atau tepatnya hanya untuk 1 session MySQL saja. Setelah itu tabel ini secara otomatis akan dihapus. Temporary tabel tidak akan ditampilkan walaupun kita menjalankan query SHOW TABLES.

Menggunakan subquery MySQL select

```
CREATE TEMPORARY TABLE tempOrderDetails (  
    productCode VARCHAR(5),  
    quantityOrdered tinyint(3)  
);
```

Gambar 10.21 Query Membuat Temporary Table TempOrderDetails

Setelah query pembuatan tabel, saya kembali memeriksa isi tabel dalam database subquery. Hasilnya? Tabel **tempOrderDetails** tidak kelihatan. Ini merupakan salah satu ciri dari temporary table, dimana kita tidak bisa melihatnya menggunakan query **SHOW TABLES**.

Untuk membuktikan bahwa tabel ini benar-benar ada, mari kita input beberapa data dan melihat hasilnya.

Outputnya akan menampilkan:

productCode	quantityOrdered
afg09	20
avk21	30
mcr	15

Gambar 10.22 Tampilan TempOrderDetails

Selanjutnya kita akan coba untuk mengakses dengan sesi yang berbeda, disini saya akan mencoba membuka cmd untuk membuat sesi yang baru (tanpa menutup localhost yang sedang running di browser).

```

MariaDB [(none)]> use subquery;
Database changed
MariaDB [subquery]> show tables;
+-----+
| Tables_in_subquery |
+-----+
| aboveavgproducts   |
| bigsalesorder       |
| customers           |
| orderdetails        |
| orders              |
| payments            |
| saleperorder        |
+-----+
7 rows in set (0.001 sec)

MariaDB [subquery]> _

```

Gambar 10.23 Melihat Tabel Menggunakan CMD

Setelah cmd terbuka, saya gunakan database yang sudah dibuat. Lalu saya melakukan query untuk melihat tabel yang tersedia di database subquery. Dan tidak menemukan tabel **tempOrderDetails**. Bagaimana jika kita coba akses langsung?

```

MariaDB [subquery]> select * from tempOrderDetails;
ERROR 1146 (42S02): Table 'subquery.temporderdetails' doesn't exist
MariaDB [subquery]>

```

Gambar 10.24 Query Menampilkan TempOrderDetails

Seperti yang terlihat, user berbeda tidak bisa melihat tabel tempOrderDetails. Lebih jauh lagi, bagaimana jika saya membuat kembali tabel tempOrderDetails menggunakan user ini? Berikut percobaannya:

```

MariaDB [subquery]> CREATE TEMPORARY TABLE tempOrderDetails (
->   productCode VARCHAR(5),
->   quantityOrdered tinyint(3)
-> );
Query OK, 0 rows affected (0.141 sec)

```

Gambar 10.25 Query Membuat Temporary Table TempOrderDetails

```

MariaDB [subquery]> show tables;
+-----+
| Tables_in_subquery |
+-----+
| aboveavgproducts    |
| bigsalesorder        |
| customers            |
| orderdetails         |
| orders              |
| payments             |
| saleperorder         |
+-----+
7 rows in set (0.001 sec)

```

Gambar 10.26 Query Menampilkan Tabel

```

MariaDB [subquery]> INSERT INTO tempOrderDetails VALUES ('afg09', 20);
Query OK, 1 row affected (0.108 sec)

MariaDB [subquery]> INSERT INTO tempOrderDetails VALUES ('avk21', 30);
Query OK, 1 row affected (0.101 sec)

MariaDB [subquery]> INSERT INTO tempOrderDetails VALUES ('mcr12', 15);
Query OK, 1 row affected (0.101 sec)

MariaDB [subquery]> select * from tempOrderDetails;
+-----+-----+
| productCode | quantityOrdered |
+-----+-----+
| afg09       | 20              |
| avk21       | 30              |
| mcr12       | 15              |
+-----+-----+
3 rows in set (0.000 sec)

```

Gambar 10.27 Mengisi Record Temporary Table

Dari contoh ini kita bisa melihat fitur penting dari temporary table, yakni hanya tersedia, dan hanya bisa diakses oleh 1 session saja.

10.5. Latihan

1. Mengapa temporary tabel tidak bisa diakses dalam sesi yang berbeda? Jelaskan beserta contohnya!
2. Jelaskan kapan kondisi yang tepat bila ingin menggunakan view tabel!
3. Perhatikan contoh diatas, jelaskan alasan temporary tabel tidak terlihat sebelum ada data yang dimasukkan!