TUGAS PRAKTIKUM BASIS DATA

PERTEMUAN 9

Siti Aulia Farhani

2206022

Informatika E

INSTITUT TEKNOLOGI GARUT

TEKNIK INFORMATIKA

2024

9.1 GROPING DATA

berguna ketika digunakan bersama dengan fungsi Aggregate untuk menggabungkan dua atau lebih kelompok data menjadi satu fungsi data tunggal. Fungsi Aggregate kemudian mengambil hasil perhitungan data tunggal dari data yang tersimpan dalam kolom.

```
MariaDB [olshopku_siti_aulia_farhani]> create table transaksi(
    -> id_trans varchar(3),
    -> nama_pelanggan varchar(15),
    -> total_harga int(11),
    -> primary key(id_trans));
Query OK, 0 rows affected (0.184 sec)
```

```
MariaDB [olshopku_siti_aulia_farhani]> desc transaksi;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| id_trans       | varchar(3)  | NO   | PRI | NULL    |       |
| nama_pelanggan | varchar(15) | YES  |     | NULL    |       |
| total_harga    | int(11)     | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
3 rows in set (0.016 sec)
```

Fungsi Aggregate merujuk pada fungsi yang digunakan untuk melakukan perhitungan pada sekelompok nilai dan mengembalikan satu nilai hasil. Ini sering digunakan dalam operasi pengelompokan data seperti GROUP BY dalam pernyataan SQL untuk menghasilkan ringkasan atau statistik dari data yang dipilih. Misalnya, fungsi agregat seperti SUM, AVG, COUNT, MAX, dan MIN sering digunakan untuk melakukan operasi perhitungan pada nilai-nilai dalam kolom.

```
MariaDB [olshopku_siti_aulia_farhani]> insert into transaksi (id_trans, nama_pelanggan, total_harga) values
    -> ('001', 'Budi', 20000),
    -> ('002', 'Nana', 15000),
    -> ('003', 'Nini', 50000),
    -> ('004', 'Jaka', 30000),
    -> ('005', 'Dina', 10000);
Query OK, 5 rows affected (0.048 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [olshopku_siti_aulia_farhani]> select * from transaksi;
+----------+---------------+-------------+
| id_trans | nama_pelanggan | total_harga |
+----------+---------------+-------------+
| 001      | Budi           |       20000 |
| 002      | Nana           |       15000 |
| 003      | Nini           |       50000 |
| 004      | Jaka           |       30000 |
| 005      | Dina           |       10000 |
+----------+---------------+-------------+
5 rows in set (0.000 sec)
```

hasilnya merupakan data tunggal yang menunjukkan jumlah baris (atau jumlah transaksi) dalam tabel transaksi.

```
MariaDB [olshopku_siti_aulia_farhani]> select count(nama_pelanggan) as jumlah_baris from transaksi;
+--------------+
| jumlah_baris |
+--------------+
|            5 |
+--------------+
1 row in set (0.000 sec)
```

2. GROUP BY

Menampilkan data pelanggan dan berapa banyak transaksi pembelian barang yang telah dilakukannya

```
MariaDB [olshopku_siti_aulia_farhani]> select
    -> nama_pelanggan,
    -> count(nama_pelanggan) 'banyak_pembelian'
    -> from transaksi group by nama_pelanggan;
+----------------+------------------+
| nama_pelanggan | banyak_pembelian |
+----------------+------------------+
| Budi           |                1 |
| Dina           |                1 |
| Jaka           |                1 |
| Nana           |                1 |
| Nini           |                1 |
+----------------+------------------+
5 rows in set (0.001 sec)
```

3. GROUP BY WITH ORDER BY

GROUP BY dapat ditambah dengan ORDER BY untuk mengurutkan hasil Query dengan syarat : ORDER BY tidak dapat digunakan pada Query yang hanya mengandung fungsi Aggregate, yaitu tanpa ada GROUP BY

```
MariaDB [olshopku_siti_aulia_farhani]> select
    -> nama_pelanggan,
    -> count(nama_pelanggan) 'banyak_pembelian'
    -> from transaksi
    -> group by nama_pelanggan
    -> order by banyak_pembelian desc;
+----------------+------------------+
| nama_pelanggan | banyak_pembelian |
+----------------+------------------+
| Nini           |                1 |
| Jaka           |                1 |
| Budi           |                1 |
| Dina           |                1 |
| Nana           |                1 |
+----------------+------------------+
5 rows in set (0.001 sec)
```

4. HAVING

HAVING adalah klausa yang digunakan bersama dengan pernyataan "GROUP BY" dalam SQL untuk menerapkan kondisi filter pada hasil pengelompokan. Klausa "HAVING" digunakan untuk memfilter hasil pengelompokan berdasarkan kriteria yang ditentukan setelah pengelompokan dilakukan. Ini memungkinkan pengguna untuk menerapkan kondisi ke grup data yang dihasilkan oleh klausa "GROUP BY". Misalnya, klausa "HAVING" dapat digunakan untuk menentukan bahwa hanya grup data yang memenuhi kondisi tertentu yang akan dimasukkan dalam hasil pengelompokan.

```
MariaDB [olshopku_siti_aulia_farhani]> SELECT
    -> nama_pelanggan,
    -> count(nama_pelanggan) 'banyak_pembelian'
    -> from transaksi
    -> group by nama_pelanggan
    -> having banyak_pembelian =1
    -> ORDER BY banyak_pembelian DESC;
+----------------+------------------+
| nama_pelanggan | banyak_pembelian |
+----------------+------------------+
| Nini           |                1 |
| Jaka           |                1 |
| Budi           |                1 |
| Dina           |                1 |
| Nana           |                1 |
+----------------+------------------+
5 rows in set (0.001 sec)
```

9.2. Triger

Trigger adalah prosedur yang secara otomatis dijalankan atau "di-trigger" oleh sistem basis data ketika suatu peristiwa tertentu terjadi. Peristiwa tersebut bisa berupa operasi manipulasi data seperti INSERT, UPDATE, DELETE, atau bahkan peristiwa non-data seperti penghapusan tabel atau perubahan skema. Triggers biasanya digunakan untuk menjaga integritas data, menerapkan aturan bisnis, atau memperbarui data terkait secara otomatis setelah perubahan tertentu dilakukan.

```
MariaDB [olshopku_siti_aulia_farhani]> create table produk (
    -> kode_produk varchar(6) not null primary key,
    -> nama_produk varchar(100) not null,
    -> harga int(11) not null);
Query OK, 0 rows affected (0.209 sec)

MariaDB [olshopku_siti_aulia_farhani]> describe produk;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| kode_produk | varchar(6)   | NO   | PRI | NULL    |       |
| nama_produk | varchar(100) | NO   |     | NULL    |       |
| harga       | int(11)      | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.028 sec)
```

Kita akan membuat fitur yang mencatat log perubahan harga barang pada sebuah database penjualan, dimana terdapat tabel produk sebagai tabel untuk menyimpan informasi produk yang memiliki field kode_produk, nama_produk dan harga

```
MariaDB [olshopku_siti_aulia_farhani]> create table log_harga_produk (
    -> log_id int(11) not null primary key auto_increment,
    -> kode_produk varchar(8) not null,
    -> harga_lama int(11) not null,
    -> harga_baru int(11) not null,
    -> waktu_perubahan datetime not null)
    -> ;
Query OK, 0 rows affected (0.157 sec)
```

lalu kita akan membuat sebuah tabel log_harga_produk untuk menyimpan informasi perubahan harga produk, informasi yang akan kita simpan adalah kode_produk, harga_lama, harga_baru dan watu_perubahan.

```
MariaDB [olshopku_siti_aulia_farhani]> create table log_harga_produk (
    -> log_id int(11) not null primary key auto_increment,
    -> kode_produk varchar(8) not null,
    -> harga_lama int(11) not null,
    -> harga_baru int(11) not null,
    -> waktu_perubahan datetime not null);
ERROR 1050 (42S01): Table 'log_harga_produk' already exists
MariaDB [olshopku_siti_aulia_farhani]> describe log_harga_produk;
+-----------------+------------+------+-----+---------+----------------+
| Field           | Type       | Null | Key | Default | Extra          |
+-----------------+------------+------+-----+---------+----------------+
| log_id          | int(11)    | NO   | PRI | NULL    | auto_increment |
| kode_produk     | varchar(8) | NO   |     | NULL    |                |
| harga_lama      | int(11)    | NO   |     | NULL    |                |
| harga_baru      | int(11)    | NO   |     | NULL    |                |
| waktu_perubahan | datetime   | NO   |     | NULL    |                |
+-----------------+------------+------+-----+---------+----------------+
5 rows in set (0.020 sec)
```

- Line 2 – Kita membuat sebuah trigger baru dengan nama before_produk_update
- Line 3 – Pada Trigger ini kita menggunakan event BEFORE UPDATE
- Line 6 – Query SQL untuk melakukan insert data ke tabel log_harga_produk

```
MariaDB [olshopku_siti_aulia_farhani]> delimiter $$
MariaDB [olshopku_siti_aulia_farhani]> create trigger before_produk_update
    -> before update on produk
    -> for each row
    -> begin
    -> insert into log_harga_produk
    -> set kode_produk = old.kode_produk,
    -> harga_baru=new.harga,
    -> harga_lama=old.harga,
    -> waktu_perubahan=now();
    -> end$$
Query OK, 0 rows affected (0.101 sec)
```

seperti yang anda lihat bahkan setelah kita melakukan proses update data lalu muncul sebuah record baru pada tabel log_harga_produk tentang informasi perubahan data produk yang sudah di update.

```
MariaDB [olshopku_siti_aulia_farhani]> insert into produk values ('BR001', 'SEMINGGU JAGO CODEIGNITER', '120000');
Query OK, 1 row affected (0.065 sec)

MariaDB [olshopku_siti_aulia_farhani]> insert into produk values ('BR002', 'SEMINGGU JAGO PHP MYSQL', '80000');
Query OK, 1 row affected (0.047 sec)
```

```
MariaDB [olshopku_siti_aulia_farhani]> select * from produk;
+-------------+---------------------------+--------+
| kode_produk | nama_produk               | harga  |
+-------------+---------------------------+--------+
| BR001       | SEMINGGU JAGO CODEIGNITER | 120000 |
| BR002       | SEMINGGU JAGO PHP MYSQL   |  80000 |
+-------------+---------------------------+--------+
2 rows in set (0.001 sec)
```

```
MariaDB [olshopku_siti_aulia_farhani]> update produk set harga=9000 where kode_produk='BR001';
Query OK, 1 row affected (0.055 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [olshopku_siti_aulia_farhani]> select * from produk;
+-------------+---------------------------+-------+
| kode_produk | nama_produk               | harga |
+-------------+---------------------------+-------+
| BR001       | SEMINGGU JAGO CODEIGNITER |  9000 |
| BR002       | SEMINGGU JAGO PHP MYSQL   | 80000 |
+-------------+---------------------------+-------+
2 rows in set (0.000 sec)
```

```
MariaDB [olshopku_siti_aulia_farhani]> select * from log_harga_produk;
+--------+-------------+------------+------------+---------------------+
| log_id | kode_produk | harga_lama | harga_baru | waktu_perubahan     |
+--------+-------------+------------+------------+---------------------+
|      1 | BR001       |     120000 |       9000 | 2024-05-27 10:01:25 |
+--------+-------------+------------+------------+---------------------+
1 row in set (0.000 sec)
```

untuk menampilkan list trigger pada sebuah database bisa menggunakan perintah berikut :

SHOW TRIGGERS