Praktikum 4.4

Praktikum Basis Data
Program Studi Manajemen Informatika
FMIPA, Universitas Riau

June 10, 2022

1 Pengantar

Pada praktikum 4.4 ini, kita akan mengenal dan mempelajari tentang konsep *View* di dalam SQL (MySQL).

2 Pengenalan View

Di dalam MySQL, View dapat didefenisikan sebagai **tabel virtual**. Tabel ini bisa berasal dari tabel lain, atau gabungan dari beberapa tabel. Tujuan dari pembuatan *View* adalah untuk kenyamanan (mempermudah penulisan *query*), untuk keamanan (menyembunyikan beberapa kolom yang bersifat rahasia), atau dalam beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan *query* tersebut secara berulang).

3 Create View

Statemen `CREATE VIEW` membuat view baru di dalam database. Berikut ini adalah sintaks dasar statemen `CREATE VIEW`:

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

Sekarang kita coba buat sampel tabel sederhana:

```
CREATE DATABASE belajar_view;
USE belajar_view;

CREATE TABLE `customers` (
  `customerNumber` int(11) NOT NULL,
  `customerName` varchar(50) NOT NULL,
  `contactLastName` varchar(50) NOT NULL,
  `contactFirstName` varchar(50) NOT NULL,
  `phone` varchar(50) NOT NULL,
  `addressLine1` varchar(50) NOT NULL,
  `addressLine2` varchar(50) DEFAULT NULL,
  `city` varchar(50) NOT NULL,
  `state` varchar(50) DEFAULT NULL,
  `postalCode` varchar(15) DEFAULT NULL,
  `country` varchar(50) NOT NULL,
```

```
  `creditLimit` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`customerNumber`)
) ENGINE=InnoDB;

CREATE TABLE `payments` (
  `customerNumber` int(11) NOT NULL,
  `checkNumber` varchar(50) NOT NULL,
  `paymentDate` date NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  PRIMARY KEY (`customerNumber`,`checkNumber`),
  CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`customerNumber`)
      REFERENCES `customers` (`customerNumber`)
) ENGINE=InnoDB;


insert  into `customers`(`customerNumber`,`customerName`,`
    contactLastName`,`contactFirstName`,`phone`,`addressLine1`,`
    addressLine2`,`city`,`state`,`postalCode`,`country`,`
    creditLimit`) values
(1,'Atelier graphique','Schmitt','Carine ','40.32.2555','54, rue
    Royale',NULL,'Nantes',NULL,'44000','France','21000.00'),
(2,'Signal Gift Stores','King','Jean','7025551838','8489 Strong St.
    ',NULL,'Las Vegas','NV','83030','USA','71800.00');


insert  into `payments`(`customerNumber`,`checkNumber`,`paymentDate
    `,`amount`) values
(1,'HQ336336','2004-10-19','6066.78'),
(1,'JM555205','2003-06-05','14571.44'),
(2,'OM314933','2004-12-18','1676.14'),
(2,'BO864823','2004-12-17','14191.12');
```

Kemudian kita kueri untuk mengambil data dari tabel *customers* dan *payments* menggunakan **INNER JOIN**:

```
SELECT
    customerName,
    checkNumber,
    paymentDate,
    amount
FROM
    customers
INNER JOIN
    payments USING (customerNumber);
```

Maka akan menampilkan hasil berikut:

```
+--------------------+-------------+-------------+----------+
| customerName       | checkNumber | paymentDate | amount   |
+--------------------+-------------+-------------+----------+
| Atelier graphique  | HQ336336    | 2004-10-19  |  6066.78 |
| Atelier graphique  | JM555205    | 2003-06-05  | 14571.44 |
| Signal Gift Stores | BO864823    | 2004-12-17  | 14191.12 |
| Signal Gift Stores | OM314933    | 2004-12-18  |  1676.14 |
+--------------------+-------------+-------------+----------+
```

Jika suatu saat kita ingin mendapatkan informasi yang sama termasuk *customerName, checkNumber, paymentDate, dan amount*, kita perlu menggunakan kueri yang sama lagi.

Cara yang lebih direkomendasikan untuk melakukannya adalah dengan menyimpan kueri di database dan menetapkan nama untuk kueri tersebut. Inilah yang disebut dengan *View* / tampilan.

Untuk membuat *View* kita menggunakan statemen CREATE VIEW. Statemen ini membuat View *customerPayments* berdasarkan kueri di atas:

```
CREATE VIEW customerPayments
AS
SELECT
    customerName,
    checkNumber,
    paymentDate,
    amount
FROM
    customers
INNER JOIN
    payments USING (customerNumber);
```

Setelah kita menjalankan pernyataan CREATE VIEW, MySQL akan membuat *View* dan menyimpannya dalam database. Sekarang, Kita dapat mereferensikan *view* sebagai tabel dalam statemen SQL. Misalnya, kita dapat meminta data dari *customerPayments* menggunakan statement SELECT:

```
SELECT * FROM customerPayments;
```

Maka akan menampilkan hasil berikut:

```
+--------------------+-------------+-------------+----------+
| customerName       | checkNumber | paymentDate | amount   |
+--------------------+-------------+-------------+----------+
| Atelier graphique  | HQ336336    | 2004-10-19  |  6066.78 |
| Atelier graphique  | JM555205    | 2003-06-05  | 14571.44 |
| Signal Gift Stores | BO864823    | 2004-12-17  | 14191.12 |
| Signal Gift Stores | OM314933    | 2004-12-18  |  1676.14 |
+--------------------+-------------+-------------+----------+
```

4 Drop Views

Statemen DROP VIEW akan menghapus VIEW dari database. Berikut sintaks dasar dari statemen DROP VIEW

```
DROP VIEW [IF EXISTS] view_name;
```

Untuk menghapus beberapa VIEW dalam satu statemen, Kita bisa menggunakan sintaks berikut:

```
DROP VIEW [IF EXISTS] view_name1 [,view_name2]...;
```

Sekarang kita coba menghapus view customerPayments yang telah kita buat sebelumnya

```
DROP VIEW IF EXISTS customerPayments;
```

5 Show Views

MySQL memperlakukan VIEWS sebagai tabel dengan tipe VIEW. Oleh karena itu, untuk menampilkan semua VIEWS dalam database saat ini, KIta menggunakan pernyataan SHOW FULL TABLES:

```
SHOW FULL TABLES
WHERE table_type = 'VIEW';
```

6 Rename Views

Berikut sintaks untuk mengubah nama VIEW:

```
RENAME TABLE productPayments
TO newProductPayments;
```