Basis Data Terdistribusi

Implementasi Partisi Basis Data

Oleh: **Hafara Firdausi (05111540000043)** https://github.com/mocatfrio/bdt-2018/tree/master/Tugas-2

Outline

1. Deskripsi Server

Server yang digunakan memiliki deskripsi sebagai berikut :

- Sistem Operasi : Linux Mint 18.3 Sylvia
- Versi MySQL : MySQL Ver 14.14 Distrib 5.7.23
- RAM : 3,7 GB (3833 MB)
- CPU : 4 core

2. Implementasi Partisi 1 : Sakila Database

2.1 Deskripsi Dataset

- Dataset yang digunakan adalah **Sakila Database**. Dapat diunduh di http://downloads.mysql.com/docs/sakila-db.zip
- Cara impor:

    1. Masuk ke dalam mysql.

    ```
    mysql -u root -p
    ```

    2. Impor database dari skema yang telah diunduh. Format :

    ```
    mysql> SOURCE /path/to/file;
    ```

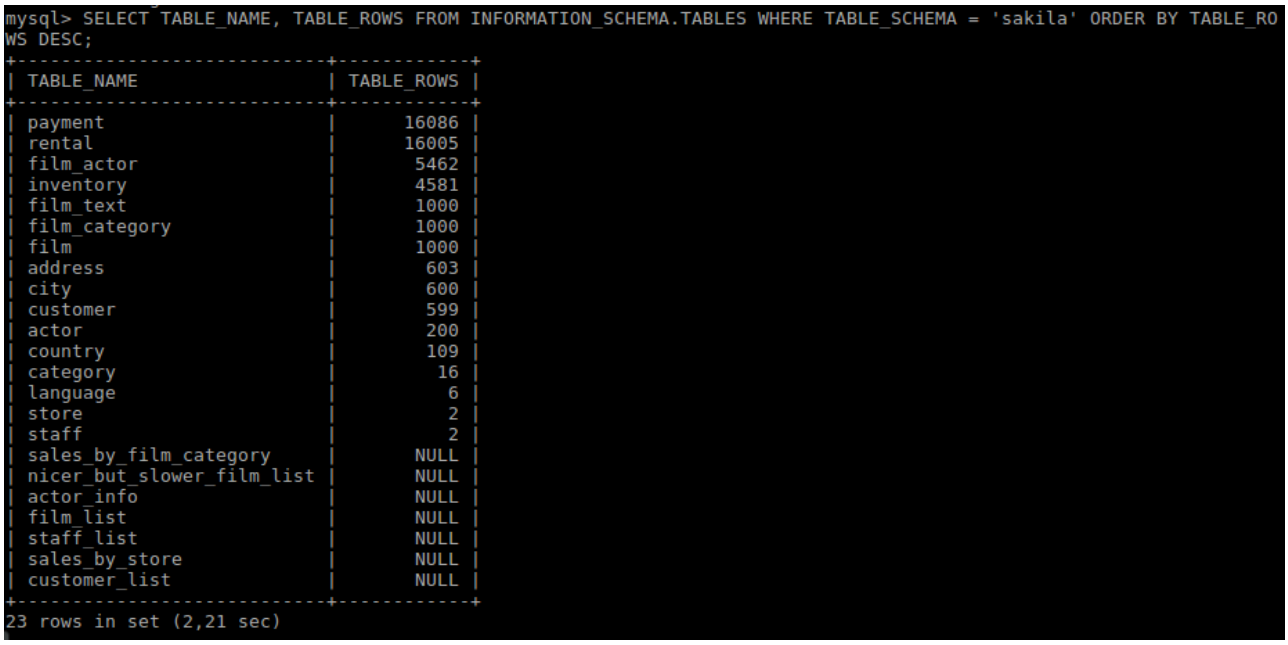
    Penggunaan :

    ```
    mysql> SOURCE sakila-db/sakila-scheme.sql;
    mysql> SOURCE sakila-db/sakila-data.sql;
    ```

- Database Sakila terdiri dari **23 tabel** (16 Tabel dan 7 View), yaitu:

```
mysql> use sakila;
Database changed
mysql> show tables;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
| city                       |
| country                    |
| customer                   |
| customer_list              |
| film                       |
| film_actor                 |
| film_category              |
| film_list                  |
| film_text                  |
| inventory                  |
| language                   |
| nicer_but_slower_film_list |
| payment                    |
| rental                     |
| sales_by_film_category     |
| sales_by_store             |
| staff                      |
| staff_list                 |
| store                      |
+----------------------------+
23 rows in set (0.00 sec)
```

- Masing-masing tabel memiliki jumlah baris data sebagai berikut:



```
mysql> SELECT TABLE_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'sakila' ORDER BY TABLE_ROWS DESC;
+----------------------------+------------+
| TABLE_NAME                 | TABLE_ROWS |
+----------------------------+------------+
| payment                    |      16086 |
| rental                     |      16005 |
| film_actor                 |       5462 |
| inventory                  |       4581 |
| film_text                  |       1000 |
| film_category              |       1000 |
| film                       |       1000 |
| address                    |        603 |
| city                       |        600 |
| customer                   |        599 |
| actor                      |        200 |
| country                    |        109 |
| category                   |         16 |
| language                   |          6 |
| store                      |          2 |
| staff                      |          2 |
| sales_by_film_category     |       NULL |
| nicer_but_slower_film_list |       NULL |
| actor_info                 |       NULL |
| film_list                  |       NULL |
| staff_list                 |       NULL |
| sales_by_store             |       NULL |
| customer_list              |       NULL |
+----------------------------+------------+
23 rows in set (2,21 sec)
```

Keterangan:

- Jumlah baris data semua tabel pada Sakila Database dapat dicari dengan:

```
SELECT TABLE_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'sakila' ORDER BY TABLE_ROWS DESC;
```

- **Desc** digunakan untuk mengurutkan tabel dari yang memiliki data paling banyak.
- Jumlah baris data juga dapat dicari satu-satu dengan command `SELECT COUNT(*) FROM nama_tabel;` contoh:

```
mysql> select count(*) from payment;
+----------+
| count(*) |
+----------+
|    16049 |
+----------+
1 row in set (0.00 sec)
```

2.2 Proses Pembuatan Partisi

2.2.1 Menentukan Tabel

- Pemilihan tabel yang akan dipartisi ditentukan berdasarkan **jumlah data terbanyak** dan memiliki kemungkinan **pertambahan dan perubahan data yang terus-menerus** karena tabel bersifat transaksional sehingga sering digunakan pada query. Tabel Payment dan Rental memiliki jumlah data paling banyak diantara semua tabel yakni 16049 dan 16044, serta bersifat transaksional.
- Daftar tabel yang akan dipartisi:
  - Tabel **Payment**

- Tabel **Rental**

Tabel Payment

- Jenis partisi yang digunakan pada tabel Payment adalah partisi **HASH** dengan parameter **payment_id**.
- Jenis partisi **HASH** tidak memerlukan predikat/syarat karena memiliki perhitungan sendiri untuk menentukan letak penyimpanan data, yakni:

```
N = MOD(expr, num)
```

  Keterangan:

  - **N =** Hasil penentuan partisi letak penyimpanan data
  - **MOD =** Modulus (operasi yang menghasilkan sisa pembagian dari suatu bilangan terhadap bilangan lainnya)
  - **expr =** Data yang akan dimodulus (dalam kasus ini adalah **payment_id**)
  - **num =** Jumlah partisi

- Tabel akan dipartisi menjadi 6 bagian yaitu p_0, p_1, p_2, p_3, p_4, dan p_5, dengan detail sebagai berikut:

  - p_0 : Data dengan **payment_date** mod 6, hasilnya 0
  - p_1 : Data dengan **payment_date** mod 6, hasilnya 1
  - p_2 : Data dengan **payment_date** mod 6, hasilnya 2
  - p_3 : Data dengan **payment_date** mod 6, hasilnya 3
  - p_4 : Data dengan **payment_date** mod 6, hasilnya 4
  - p_5 : Data dengan **payment_date** mod 6, hasilnya 5

Tabel Rental

- Jenis partisi yang digunakan pada tabel Rental adalah metode **RANGE** dengan parameter tanggal pada **rental_date**.
- Predikat/syarat yang digunakan dalam membuat partisi:
  - p_0 : Data dengan tanggal **payment_date** antara 1 sampai dengan 10
  - p_1 : Data dengan tanggal **payment_date** antara 11 sampai dengan 20
  - p_2 : Data dengan tanggal **payment_date** lebih dari sama dengan 21
- Sehingga, berdasarkan syarat/predikat diatas, maka tabel akan dipartisi menjadi 3 bagian dengan nama partisi sebagai berikut:
  - day_from_1_to_10
  - day_from_11_to_20
  - day_from_21_and_up

2.2.2 Implementasi Partisi

Implementasi partisi dilakukan dengan mengubah script skema SQL yang sudah ada dan menjalankan ulang script SQL-nya.

1. Menambah dan mengubah script SQL untuk membuat partisi tabel **Payment**.

```
CREATE TABLE payment (
    payment_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    customer_id SMALLINT UNSIGNED NOT NULL,
    staff_id TINYINT UNSIGNED NOT NULL,
    rental_id INT DEFAULT NULL,
    amount DECIMAL(5,2) NOT NULL,
    payment_date DATETIME NOT NULL,
    last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY  (payment_id),
    KEY idx_fk_staff_id (staff_id),
    KEY idx_fk_customer_id (customer_id)
    -- CONSTRAINT fk_payment_rental FOREIGN KEY (rental_id) REFERENCES
rental (rental_id) ON DELETE SET NULL ON UPDATE CASCADE,
    -- CONSTRAINT fk_payment_customer FOREIGN KEY (customer_id)
REFERENCES customer (customer_id) ON DELETE RESTRICT ON UPDATE
CASCADE,
    -- CONSTRAINT fk_payment_staff FOREIGN KEY (staff_id) REFERENCES
staff (staff_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE payment
    PARTITION BY HASH (payment_id) PARTITIONS 6;
```

   Keterangan:

   - Pendefinisian `Foreign Key` dicomment/dihapus agar tidak dijalankan pada proses pembuatan partisi karena akan menyebabkan error.
   - Setelah meng-comment, jangan lupa menghapus tanda koma pada row sebelumnya.

2. Menambah dan mengubah script SQL untuk membuat partisi tabel **Rental**.

```
CREATE TABLE rental (
    rental_id INT NOT NULL AUTO_INCREMENT,
    rental_date DATETIME NOT NULL,
    inventory_id MEDIUMINT UNSIGNED NOT NULL,
    customer_id SMALLINT UNSIGNED NOT NULL,
    return_date DATETIME DEFAULT NULL,
    staff_id TINYINT UNSIGNED NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (rental_id, rental_date),
    UNIQUE KEY  (rental_date,inventory_id,customer_id),
    KEY idx_fk_inventory_id (inventory_id),
    KEY idx_fk_customer_id (customer_id),
```

```
    KEY idx_fk_staff_id (staff_id)
    -- CONSTRAINT fk_rental_staff FOREIGN KEY (staff_id) REFERENCES
staff (staff_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    -- CONSTRAINT fk_rental_inventory FOREIGN KEY (inventory_id)
REFERENCES inventory (inventory_id) ON DELETE RESTRICT ON UPDATE
CASCADE,
    -- CONSTRAINT fk_rental_customer FOREIGN KEY (customer_id)
REFERENCES customer (customer_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE rental
    PARTITION BY RANGE ( DAY(rental_date) ) (
        PARTITION day_from_1_to_10 VALUES LESS THAN (11),
        PARTITION day_from_11_to_20 VALUES LESS THAN (21),
        PARTITION day_from_21_and_up VALUES LESS THAN MAXVALUE
    );
```

Keterangan:

- Pendefinisian `Foreign Key` dicomment/dihapus agar tidak dijalankan pada proses pembuatan partisi karena akan menyebabkan error.
- Setelah meng-comment, jangan lupa menghapus tanda koma pada row sebelumnya.
- Parameter partisi pada tabel Rental adalah `rental_date`, dimana `rental_date` bukanlah PRIMARY KEY. Maka `rental_date` ditambahkan menjadi PRIMARY KEY agar dapat disertakan pada masing-masing tabel partisi. Menambahkan pada baris `PRIMARY KEY (rental_id, rental_date)`.
- Fungsi partisi untuk RANGE berdasarkan tanggal menggunakan perintah `DAY()`;

3. Menjalankan ulang script skema SQL yang telah diubah dan diatur konfigurasi partisinya.

```
mysql -u root -p
```

```
SOURCE sakila-db/sakila-scheme.sql;
```

Cek apakah script SQL telah berjalan dengan baik.

```
Query OK, 0 rows affected (3,88 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0,77 sec)

Query OK, 0 rows affected (2,47 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0,44 sec)
```

Jika ada error, maka dicari tahu terlebih dahulu letak errornya dan diselesaikan.

4. Cek partisi yang sudah dibuat dengan command `EXPLAIN SELECT * FROM nama_tabel\G`

```
mysql> EXPLAIN SELECT * FROM payment\G;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: payment
   partitions: p0,p1,p2,p3,p4,p5
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0,00 sec)

ERROR:
No query specified

mysql> EXPLAIN SELECT * FROM rental\G;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: rental
   partitions: day_from_1_to_10,day_from_11_to_20,day_from_21_and_up
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0,00 sec)

ERROR:
No query specified
```

2.3 Benchmarking

Untuk mengecek apakah proses partisi telah benar-benar berhasil, maka dilakukan proses pengujian yakni dengan cara memasukkan beberapa data baru (`INSERT`) dan menguji dengan query `SELECT` dari partisi tertentu.

2.3.1 Tabel Payment

1. Memasukkan beberapa data baru (minimal 10 data) pada masing-masing tabel. Karena partisi tabel Payment terdiri atas 6 bagian, maka total jumlah data baru yang diinsert ada 60 data. Data bisa dimasukkan melalui terminal/tools SQL (MySQL Workbench, DBeaver, atau semacamnya).

```
INSERT INTO payment (customer_id, staff_id, rental_id, amount,
payment_date, last_update) VALUES
    (1,2,1234,'6.99','2005-05-28 10:35:23','2006-02-15 22:12:30'),
    (1,1,7841,'4.99','2005-07-28 09:04:45','2006-02-15 22:12:30'),
    (1,2,8033,'4.99','2005-07-28 16:18:23','2006-02-15 22:12:30'),
```

```
(1,1,8074,'0.99','2005-07-28 17:33:39','2006-02-15 22:12:30'),
(1,2,8116,'0.99','2005-07-28 19:20:07','2006-02-15 22:12:30'),
(1,2,8326,'2.99','2005-07-29 03:58:49','2006-02-15 22:12:30'),
(1,2,13068,'0.99','2005-08-19 09:55:16','2006-02-15 22:12:30'),
(1,2,13176,'2.99','2005-08-19 13:56:54','2006-02-15 22:12:30'),
(1,1,14762,'0.99','2005-08-21 23:33:57','2006-02-15 22:12:30'),
(1,1,7096,'5.99','2005-07-27 04:54:42','2006-02-15 22:12:30'),

(2,1,14825,'1.99','2005-08-22 01:27:57','2006-02-15 22:12:30'),
(2,2,15298,'2.99','2005-08-22 19:41:37','2006-02-15 22:12:30'),
(2,1,15315,'5.99','2005-08-22 20:03:46','2006-02-15 22:12:30'),
(2,1,435,'1.99','2005-05-27 17:17:09','2006-02-15 22:12:30'),
(2,1,830,'2.99','2005-05-29 22:43:55','2006-02-15 22:12:30'),
(2,1,1546,'8.99','2005-06-16 01:34:05','2006-02-15 22:12:30'),
(2,1,1726,'6.99','2005-06-16 15:19:10','2006-02-15 22:12:30'),
(2,2,1911,'6.99','2005-06-17 05:15:15','2006-02-15 22:12:30'),
(2,1,2628,'2.99','2005-06-19 08:34:53','2006-02-15 22:12:30'),
(2,1,4180,'4.99','2005-07-07 10:23:25','2006-02-15 22:12:30'),

(3,1,4725,'4.99','2005-07-08 12:47:11','2006-02-15 22:12:30'),
(3,2,12556,'4.99','2005-08-18 14:49:55','2006-02-15 22:12:30'),
(3,1,13403,'8.99','2005-08-19 22:18:07','2006-02-15 22:12:30'),
(3,2,13610,'2.99','2005-08-20 06:14:12','2006-02-15 22:12:30'),
(3,2,14699,'8.99','2005-08-21 20:50:48','2006-02-15 22:12:30'),
(3,2,15038,'0.99','2005-08-22 09:37:27','2006-02-15 22:12:30'),
(3,1,15619,'2.99','2005-08-23 07:10:14','2006-02-15 22:12:30'),
(3,1,7096,'5.99','2005-07-27 04:54:42','2006-02-15 22:12:30'),
(3,2,7503,'10.99','2005-07-27 20:23:12','2006-02-15 22:12:30'),
(3,2,7703,'7.99','2005-07-28 03:59:21','2006-02-15 22:12:30'),

(4,1,1297,'4.99','2005-06-15 09:31:28','2006-02-15 22:12:30'),
(4,1,1633,'0.99','2005-06-16 08:08:40','2006-02-15 22:12:30'),
(4,2,1707,'2.99','2005-06-16 14:01:27','2006-02-15 22:12:30'),
(4,2,1735,'0.99','2005-06-16 15:51:52','2006-02-15 22:12:30'),
(4,2,2043,'0.99','2005-06-17 14:31:12','2006-02-15 22:12:30'),
(4,1,2642,'5.99','2005-06-19 09:39:01','2006-02-15 22:12:30'),
(4,1,7660,'2.99','2005-07-28 02:10:10','2006-02-15 22:12:30'),
(4,2,7718,'2.99','2005-07-28 04:37:59','2006-02-15 22:12:30'),
(4,1,8741,'3.99','2005-07-29 18:44:57','2006-02-15 22:12:30'),
(4,1,9100,'5.99','2005-07-30 08:46:09','2006-02-15 22:12:30'),

(5,1,731,'0.99','2005-05-29 07:25:16','2006-02-15 22:12:30'),
(5,1,1085,'6.99','2005-05-31 11:15:43','2006-02-15 22:12:30'),
(5,1,1142,'1.99','2005-05-31 19:46:38','2006-02-15 22:12:30'),
(5,1,1502,'3.99','2005-06-15 22:03:14','2006-02-15 22:12:30'),
(5,2,1631,'2.99','2005-06-16 08:01:02','2006-02-15 22:12:30'),
(5,2,2063,'4.99','2005-06-17 15:56:53','2006-02-15 22:12:30'),
(5,2,2570,'2.99','2005-06-19 04:20:13','2006-02-15 22:12:30'),
(5,2,3126,'4.99','2005-06-20 18:38:22','2006-02-15 22:12:30'),
(5,2,3677,'4.99','2005-07-06 09:11:58','2006-02-15 22:12:30'),
(5,2,4889,'2.99','2005-07-08 20:04:43','2006-02-15 22:12:30'),

(6,1,11023,'2.99','2005-08-02 05:36:38','2006-02-15 22:12:30'),
(6,1,11398,'3.99','2005-08-02 18:55:15','2006-02-15 22:12:30'),
```

```
    (6,1,11591,'6.99','2005-08-17 02:29:41','2006-02-15 22:12:30'),
    (6,1,11727,'0.99','2005-08-17 08:12:20','2006-02-15 22:12:30'),
    (6,1,11853,'0.99','2005-08-17 13:39:32','2006-02-15 22:12:30'),
    (6,2,12254,'2.99','2005-08-18 04:05:29','2006-02-15 22:12:30'),
    (6,2,13451,'6.99','2005-08-20 00:18:25','2006-02-15 22:12:30'),
    (6,1,14329,'7.99','2005-08-21 08:22:56','2006-02-15 22:12:30'),
    (6,1,14377,'4.99','2005-08-21 09:49:28','2006-02-15 22:12:30'),
    (6,1,14329,'7.99','2005-08-21 08:22:56','2006-02-15 22:12:30');
```

Keterangan:

- Data tidak ditulis secara spesifik akan masuk ke tabel mana karena **payment_id** merupakan kolom yang auto increment sehingga tidak perlu didefinisikan secara manual ketika INSERT

2. Melakukan pengujian menggunakan query SELECT dengan kasus untuk mencari data dengan **payment_id = 43**, dimana `43 mod 6 = 1` sehingga data seharusnya disimpan pada partisi p_1.

   - Menjalankan query SELECT data dengan payment_id = 43 dari partisi yang benar.

     ```
     SELECT * FROM payment PARTITION(p1) WHERE payment_id = 43;
     ```

     Hasil:

     ```
     mysql> SELECT * FROM payment PARTITION(p1) WHERE payment_id = 43;
     +------------+-------------+----------+-----------+--------+---------------------+---------------------+
     | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
     +------------+-------------+----------+-----------+--------+---------------------+---------------------+
     |         43 |           5 |        1 |      1142 |   1.99 | 2005-05-31 19:46:38 | 2006-02-15 22:12:30 |
     +------------+-------------+----------+-----------+--------+---------------------+---------------------+
     1 row in set (0,00 sec)
     ```

   - Menjalankan query SELECT data dengan payment_id = 43 dari partisi yang salah.

     ```
     SELECT * FROM payment PARTITION(p0) WHERE payment_id = 43;
     SELECT * FROM payment PARTITION(p2) WHERE payment_id = 43;
     SELECT * FROM payment PARTITION(p3) WHERE payment_id = 43;
     SELECT * FROM payment PARTITION(p4) WHERE payment_id = 43;
     SELECT * FROM payment PARTITION(p5) WHERE payment_id = 43;
     ```

     Hasil:

```
mysql> SELECT * FROM payment PARTITION(p0) WHERE payment_id = 43;
Empty set (0,00 sec)

mysql> SELECT * FROM payment PARTITION(p2) WHERE payment_id = 43;
Empty set (0,00 sec)

mysql> SELECT * FROM payment PARTITION(p3) WHERE payment_id = 43;
Empty set (0,00 sec)

mysql> SELECT * FROM payment PARTITION(p4) WHERE payment_id = 43;
Empty set (0,00 sec)

mysql> SELECT * FROM payment PARTITION(p5) WHERE payment_id = 43;
Empty set (0,00 sec)
```

2.3.2 Tabel Rental

> Catatan Penting: Untuk tabel Rental perlu dilakukan ADD UNIQUE INDEX terlebih dahulu pada **rental_date** supaya bisa dilakukan INSERT data.

```
ALTER TABLE rental
DROP INDEX rental_date,
ADD UNIQUE INDEX rental_date (rental_id ASC, rental_date ASC,
inventory_id ASC, customer_id ASC);
```

```
mysql> ALTER TABLE rental DROP INDEX rental_date, ADD UNIQUE INDEX rental_date (rental_id ASC, rental_date ASC, inventor
y_id ASC, customer_id ASC);
Query OK, 0 rows affected (1,09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

1. Memasukkan beberapa data baru (minimal 10 data) pada masing-masing tabel. Karena partisi tabel Rental terdiri atas 3 bagian, maka total jumlah data baru yang diinsert ada 30 data. Data bisa dimasukkan melalui terminal/tools SQL (MySQL Workbench, DBeaver, atau semacamnya).

```
INSERT INTO rental (rental_date, inventory_id, customer_id,
return_date, staff_id, last_update) VALUES
-- insert data ke partisi 1 (day_from_1_to_10)
('2005-05-01 22:54:33',1525,459,'2005-05-28 19:40:33',1,'2006-02-15
21:30:53'),
('2005-05-02 23:03:39',1711,408,'2005-06-01 22:12:39',1,'2006-02-15
21:30:53'),
('2005-05-03 23:04:41',2452,333,'2005-06-03 01:43:41',2,'2006-02-15
21:30:53'),
('2005-05-04 23:05:21',2079,222,'2005-06-02 04:33:21',1,'2006-02-15
21:30:53'),
('2005-05-05 23:08:07',2792,549,'2005-05-27 01:32:07',1,'2006-02-15
21:30:53'),
('2005-05-06 23:11:53',3995,269,'2005-05-29 20:34:53',2,'2006-02-15
21:30:53'),
('2005-05-07 23:31:46',2346,239,'2005-05-27 23:33:46',2,'2006-02-15
21:30:53'),
('2005-05-08 00:00:40',2580,126,'2005-05-28 00:22:40',1,'2006-02-15
21:30:53'),
```

```
('2005-05-09 00:02:21',1824,399,'2005-05-31 22:44:21',2,'2006-02-15
21:30:53'),
('2005-05-10 00:09:02',4443,142,'2005-06-02 20:56:02',2,'2006-02-15
21:30:53'),

-- insert data ke partisi 2 (day_from_11_to_20)
('2005-05-11 00:19:27',1584,261,'2005-05-30 05:44:27',2,'2006-02-15
21:30:53'),
('2005-05-12 00:22:55',2294,334,'2005-05-30 04:28:55',1,'2006-02-15
21:30:53'),
('2005-05-13 00:31:15',2701,446,'2005-05-26 02:56:15',1,'2006-02-15
21:30:53'),
('2005-05-14 00:39:22',3049,319,'2005-06-03 03:30:22',1,'2006-02-15
21:30:53'),
('2005-05-15 00:43:11',389,316,'2005-05-26 04:42:11',2,'2006-02-15
21:30:53'),
('2005-05-16 01:06:36',830,575,'2005-05-27 00:43:36',1,'2006-02-15
21:30:53'),
('2005-05-17 17:17:04',617,468,'2005-05-31 19:47:04',1,'2006-02-15
21:30:53'),
('2005-05-18 17:22:10',373,343,'2005-05-31 19:47:10',1,'2006-02-15
21:30:53'),
('2005-05-19 17:30:42',3343,384,'2005-06-03 22:36:42',1,'2006-02-15
21:30:53'),
('2005-05-20 17:46:33',4281,310,'2005-05-27 15:20:33',1,'2006-02-15
21:30:53'),

-- insert data ke partisi 1 (day_from_21_until_up)
('2005-05-21 17:54:12',794,108,'2005-05-30 12:03:12',2,'2006-02-15
21:30:53'),
('2005-05-22 18:18:19',3627,196,'2005-06-04 00:01:19',2,'2006-02-15
21:30:53'),
('2005-05-23 18:28:09',2833,317,'2005-06-03 22:46:09',2,'2006-02-15
21:30:53'),
('2005-05-24 18:30:05',3289,242,'2005-05-30 19:40:05',1,'2006-02-15
21:30:53'),
('2005-05-25 18:40:20',1044,503,'2005-05-29 20:39:20',2,'2006-02-15
21:30:53'),
('2005-05-26 18:43:49',4108,19,'2005-06-03 18:13:49',2,'2006-02-15
21:30:53'),
('2005-05-27 18:45:19',3725,227,'2005-05-28 17:18:19',1,'2006-02-15
21:30:53'),
('2005-05-28 18:57:24',2153,500,'2005-06-02 20:44:24',1,'2006-02-15
21:30:53'),
('2005-05-29 19:07:40',2963,93,'2005-05-27 22:16:40',2,'2006-02-15
21:30:53'),
('2005-05-30 19:12:42',4502,506,'2005-06-01 23:10:42',1,'2006-02-15
21:30:53'),
('2005-05-31 19:13:25',749,455,'2005-05-29 20:17:25',1,'2006-02-15
21:30:53');
```

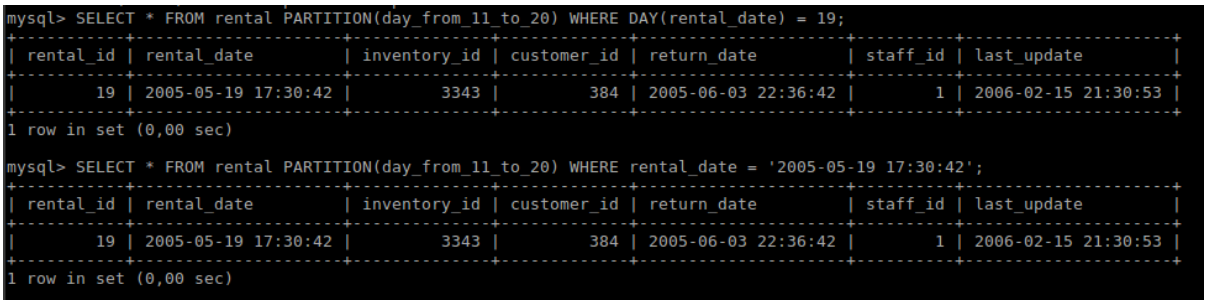
2. Melakukan pengujian menggunakan query SELECT dengan kasus untuk mencari data dengan **rental_date tanggal 19**, dimana tanggal 19 seharusnya tersimpan pada partisi `day_from_11_to_20`.

    - Menjalankan query SELECT data dengan rental_date = 19 dari partisi yang benar.

```
-- SELECT berdasarkan tanggal saja
SELECT * FROM rental PARTITION(day_from_11_to_20) WHERE
DAY(rental_date) = 19;

-- SELECT berdasarkan timestamp lengkap (lebih spesifik)
SELECT * FROM rental PARTITION(day_from_11_to_20) WHERE
rental_date = '2005-05-19 17:30:42';
```

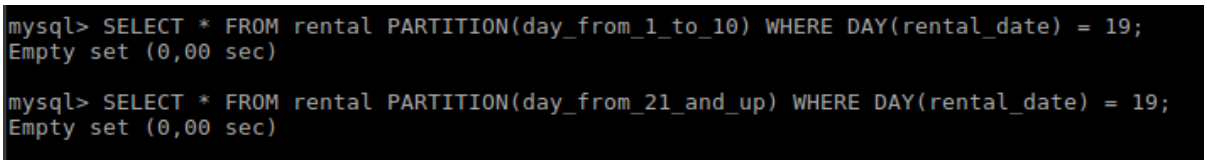
    Hasil:



    - Menjalankan query SELECT data dengan rental_date = 19 dari partisi yang salah.

```
SELECT * FROM rental PARTITION(day_from_1_to_10) WHERE
DAY(rental_date) = 19;
SELECT * FROM rental PARTITION(day_from_21_and_up) WHERE
DAY(rental_date) = 19;
```

    Hasil:



3. Implementasi Partisi 2 : Measures Dataset

3.1 Deskripsi Dataset

- Dataset yang digunakan adalah **Measures Dataset** dari Vertabelo. Dapat diunduh di https://drive.google.com/file/d/0B2Ksz9hP3LtXRUppZHdhT1pBaWM/view
- Dataset Measures terdiri dari **2 tabel**, yaitu:

```
mysql> use measures;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------+
| Tables_in_measures   |
+----------------------+
| measures             |
| partitioned_measures |
+----------------------+
2 rows in set (0.00 sec)
```

- Masing-masing tabel memiliki jumlah baris data sebagai berikut:

No	Nama Tabel	Jumlah Data
1	measures	1846124
2	partitioned_measures	1846124

Keterangan:

- Tabel **partitioned_measures** adalah bentuk partisi dari tabel **measures**, sehingga keduanya memiliki jumlah data yang sama.

3.2 Cara Impor Dataset

1. Unduh Measures Dataset dari link yang telah disebutkan di atas.
2. Masuk ke dalam MySQL.

```
mysql -u root -p
```

3. Membuat database bernama **vertabelo**. Dataset tidak dapat diimpor jika database belum dibuat.

```
CREATE DATABASE vertabelo;
```

4. Keluar dari MySQL dengan mengetikkan `quit;`
5. Mengimpor Measures Dataset

```
mysql -u root -p -D vertabelo < nama_file_dataset.sql
```

Contoh :

```
mysql -u root -p -D vertabelo < sample_1_8_M_rows_data.sql
```

3.3 Benchmarking

Ada 2 jenis benchmark, yakni SELECT Queries Benchmark dan Big Delete Benchmark, yang dilakukan sebanyak 10 kali pada masing-masing tabel, kemudian kecepatan query-nya dicatat dan dirata-rata.

3.3.1 Select Queries Benchmark

Langkah-Langkah

1. Sebelum melakukan select queries benchmark, **index** pada kedua tabel harus dihapus terlebih dahulu supaya terlihat performa asli query.

```
ALTER TABLE vertabelo.measures
DROP INDEX measure_timestamp;

ALTER TABLE vertabelo.partitioned_measures
DROP INDEX measure_timestamp;
```

2. SELECT query pada tabel **measures** (tanpa partisi)

```
SELECT SQL_NO_CACHE
    COUNT(*)
FROM
    vertabelo.measures
WHERE
    measure_timestamp >= '2016-01-01'
        AND DAYOFWEEK(measure_timestamp) = 1;
```

3. SELECT query pada tabel **partitioned_measures**

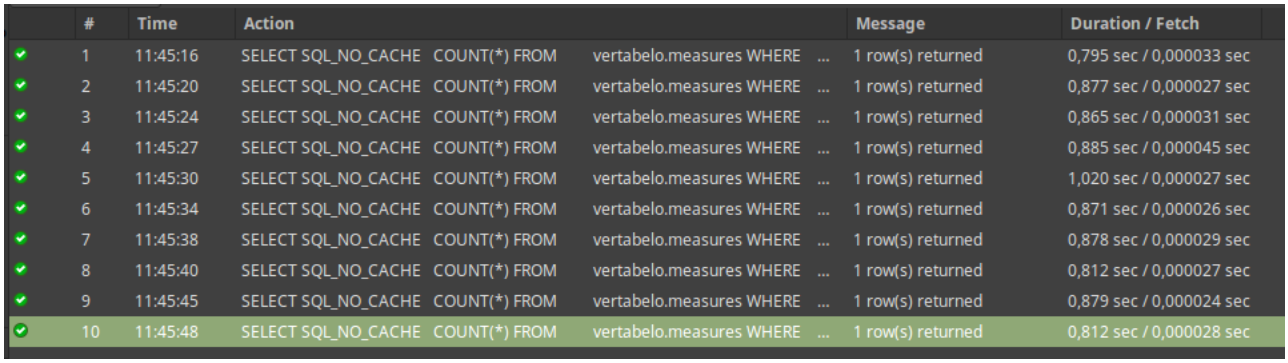
```
SELECT SQL_NO_CACHE
    COUNT(*)
FROM
    vertabelo.partitioned_measures
WHERE
    measure_timestamp >= '2016-01-01'
        AND DAYOFWEEK(measure_timestamp) = 1;
```

Hasil

No. Pengujian	Tabel Tanpa Partisi (detik)	Tabel Dengan Partisi (detik)
1	0.795	0.521

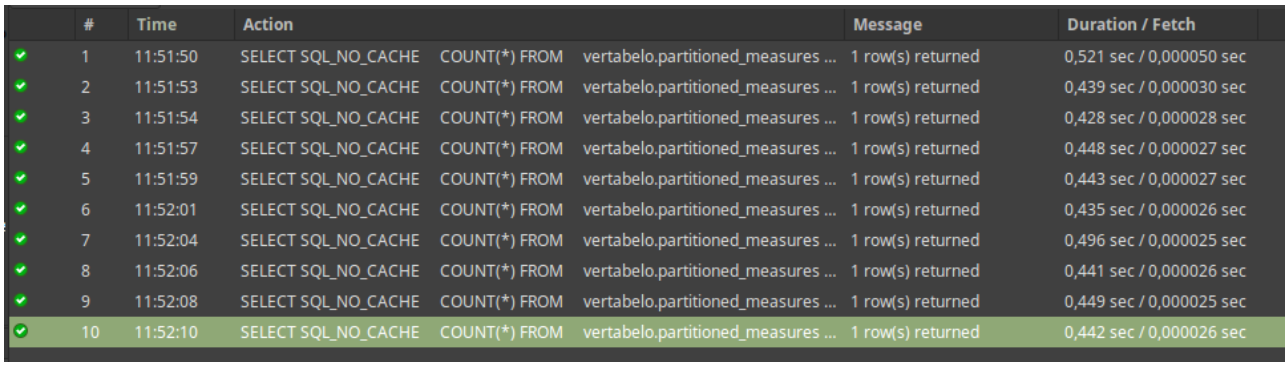
No. Pengujian	Tabel Tanpa Partisi (detik)	Tabel Dengan Partisi (detik)
2	0.877	0.439
3	0.865	0.428
4	0.885	0.448
5	1.020	0.443
6	0.871	0.435
7	0.878	0.496
8	0.812	0.441
9	0.879	0.449
10	0.812	0.442
Rata-Rata	**0.8694**	**0.4542**

- Hasil select query terhadap tabel **measures** (tanpa partisi)



#	Time	Action	Message	Duration / Fetch
1	11:45:16	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,795 sec / 0,000033 sec
2	11:45:20	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,877 sec / 0,000027 sec
3	11:45:24	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,865 sec / 0,000031 sec
4	11:45:27	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,885 sec / 0,000045 sec
5	11:45:30	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	1,020 sec / 0,000027 sec
6	11:45:34	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,871 sec / 0,000026 sec
7	11:45:38	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,878 sec / 0,000029 sec
8	11:45:40	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,812 sec / 0,000027 sec
9	11:45:45	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,879 sec / 0,000024 sec
10	11:45:48	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.measures WHERE ...	1 row(s) returned	0,812 sec / 0,000028 sec

- Hasil select query terhadap tabel **partitioned_measures**



#	Time	Action	Message	Duration / Fetch
1	11:51:50	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,521 sec / 0,000050 sec
2	11:51:53	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,439 sec / 0,000030 sec
3	11:51:54	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,428 sec / 0,000028 sec
4	11:51:57	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,448 sec / 0,000027 sec
5	11:51:59	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,443 sec / 0,000027 sec
6	11:52:01	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,435 sec / 0,000026 sec
7	11:52:04	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,496 sec / 0,000025 sec
8	11:52:06	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,441 sec / 0,000026 sec
9	11:52:08	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,449 sec / 0,000025 sec
10	11:52:10	SELECT SQL_NO_CACHE COUNT(*) FROM vertabelo.partitioned_measures ...	1 row(s) returned	0,442 sec / 0,000026 sec

3.3.2 Big Delete Benchmark

Langkah-Langkah

1. Sebelum melakukan big delete benchmark, **index** pada `measure_timestamp` harus ditambahkan kembali terlebih dahulu.

```
ALTER TABLE vertabelo.measures
ADD INDEX index1 (measure_timestamp ASC);

ALTER TABLE vertabelo.partitioned_measures
ADD INDEX index1 (measure_timestamp ASC);
```

2. Big delete pada tabel **measures** (tanpa partisi)

```
DELETE
FROM vertabelo.measures
WHERE measure_timestamp < '2015-01-01';
```

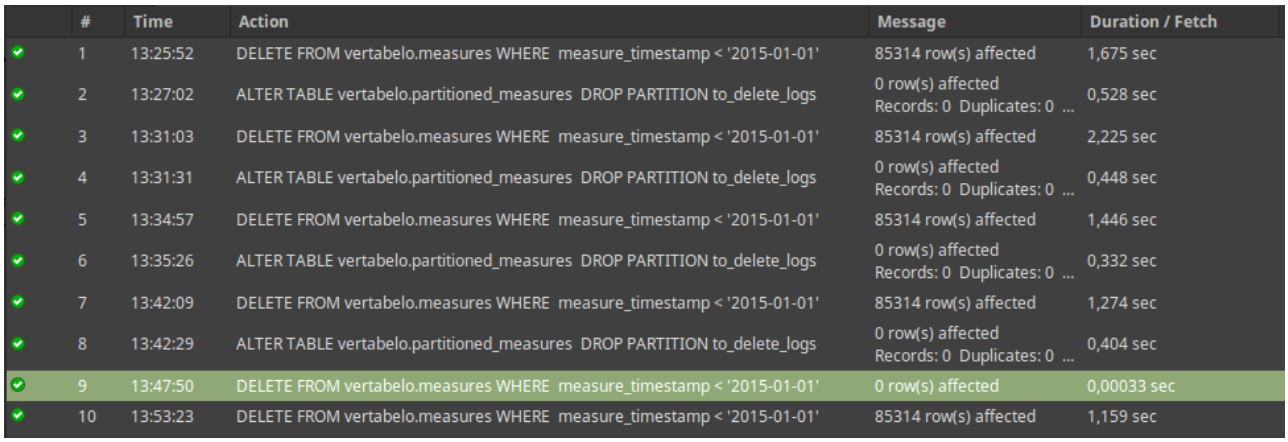
3. Big delete pada tabel **partitioned_measures**

```
ALTER TABLE vertabelo.partitioned_measures
DROP PARTITION to_delete_logs;
```

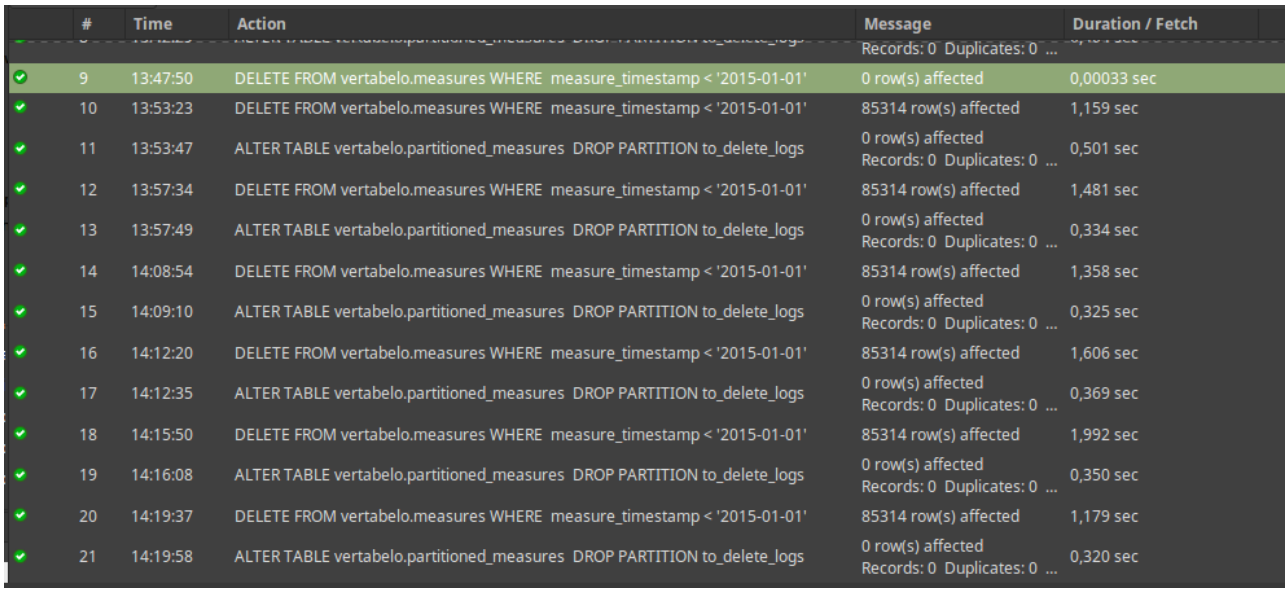
Hasil

No. Pengujian	Tabel Tanpa Partisi (detik)	Tabel Dengan Partisi (detik)
1	1.675	0.528
2	2.225	0.448
3	1.446	0.332
4	1.274	0.404
5	1.159	0.501
6	1.481	0.334
7	1.358	0.325
8	1.606	0.369
9	1.992	0.350
10	1.179	0.320
Rata-Rata	**1.5395**	**0.3911**

- Hasil big delete terhadap tabel **measures** dan **partitioned_measures** (bergantian/berselingan)



#	Time	Action	Message	Duration / Fetch
1	13:25:52	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,675 sec
2	13:27:02	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,528 sec
3	13:31:03	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	2,225 sec
4	13:31:31	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,448 sec
5	13:34:57	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,446 sec
6	13:35:26	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,332 sec
7	13:42:09	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,274 sec
8	13:42:29	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,404 sec
9	13:47:50	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	0 row(s) affected	0,00033 sec
10	13:53:23	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,159 sec



#	Time	Action	Message	Duration / Fetch
			Records: 0 Duplicates: 0 ...	
9	13:47:50	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	0 row(s) affected	0,00033 sec
10	13:53:23	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,159 sec
11	13:53:47	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,501 sec
12	13:57:34	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,481 sec
13	13:57:49	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,334 sec
14	14:08:54	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,358 sec
15	14:09:10	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,325 sec
16	14:12:20	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,606 sec
17	14:12:35	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,369 sec
18	14:15:50	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,992 sec
19	14:16:08	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,350 sec
20	14:19:37	DELETE FROM vertabelo.measures WHERE measure_timestamp < '2015-01-01'	85314 row(s) affected	1,179 sec
21	14:19:58	ALTER TABLE vertabelo.partitioned_measures DROP PARTITION to_delete_logs	0 row(s) affected Records: 0 Duplicates: 0 ...	0,320 sec

3.4 Kesimpulan

Dari hasil benchmarking di atas, dapat disimpulkan bahwa:

- **Kecepatan select query pada tabel yang dipartisi hampir dua kali lebih cepat** dibandingkan dengan kecepatan select query pada tabel yang tidak dipartisi.
- **Proses menghapus data hampir empat kali lebih cepat** dilakukan pada tabel yang dipartisi karena hanya perlu menghapus partisi tertentu yang berisi data-data yang ingin dihapus. Sedangkan pada tabel yang tidak dipartisi, proses penghapusan data lebih lama karena harus mencari dan memilah terlebih dahulu data yang ingin dihapus.

Referensi

- Claria, Francisco, (10 Februari 2017), Everything You Need to Know About MySQL Partitions, [online], (http://www.vertabelo.com/blog/technical-articles/everything-you-need-to-know-about-mysql-partitions, diakses tanggal 25 September 2018)
- MySQL, Sakila Installation Documentation, (https://dev.mysql.com/doc/sakila/en/sakila-installation.html, diakses tanggal 25 September 2018)