

Perbandingan Performa Teknik Relational dan Prosedural dalam Database

Djoni Darmawikarta
djoni_darmawikarta@yahoo.ca

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

*Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.Com**.*

Pemanfaatan teknologi relasional didalam database membantu pencapaian performance (kecepatan pengolahan data) yang optimum dan kemudahan pengertian program sehingga pemeliharaannya jadi lebih nyaman. Dua manfaat ini tidaklah susah dicapai bila kita, sebagai developer, berani berpaling dari tradisi memprogram teknik prosedural ke relasional.

Untuk memperjelas dan meyakinkan keunggulan teknik pemrograman **relasional** dibanding **prosedural** dalam pengolahan data didalam database relasional, berikut contoh *sederhana* yang membuktikannya. Contoh menggunakan database MySQL.

Contoh kasus

Setiap akhir hari, besarnya transaksi penjualan diminta untuk diringkas per kode produk. Ringkasan (total penjualan) adalah netto, yaitu besar penjualan dikurangi diskon. Ringkasan disimpan dalam tabel untuk digunakan selanjutnya, misalnya untuk pelaporan pimpinan, masukan (input) aplikasi lain seperti pemesanan ke pabrikan produk dan system akutansi.

Tabel transaksi_penjualan, seperti ditunjukkan dilayar mysql monitor Gambar 1 berikut ini, memiliki tiga kolom yaitu kode_produk, besar_penjualan, dan diskon.

```
Command Prompt - mysql -uroot -p

C:\>mysql -uroot -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 5.0.1-alpha-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE test
Database changed
mysql> SHOW CREATE TABLE transaksi_penjualan \G
***** 1. row *****
      Table: transaksi_penjualan
Create Table: CREATE TABLE `transaksi_penjualan` (
  `kode_produk` int(4) default NULL,
  `besar_penjualan` decimal(8,2) default NULL,
  `diskon` int(2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql>
```

Gambar 1 Struktur tabel transaksi_penjualan.

Sedang tabel penjualan_harian memiliki dua kolom (kode_produk dan total_penjualan) seperti ditunjukkan di Gambar 2.

```
Command Prompt - mysql -uroot -p

C:\>mysql -uroot -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9 to server version: 5.0.1-alpha-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE test
Database changed
mysql> SHOW CREATE TABLE transaksi_penjualan \G
***** 1. row *****
      Table: transaksi_penjualan
Create Table: CREATE TABLE `transaksi_penjualan` (
  `kode_produk` int(4) default NULL,
  `besar_penjualan` decimal(8,2) default NULL,
  `diskon` int(2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE penjualan_harian \G
***** 1. row *****
      Table: penjualan_harian
Create Table: CREATE TABLE `penjualan_harian` (
  `kode_produk` int(4) default NULL,
  `total_penjualan` decimal(20,2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql>
```

Gambar 2 Struktur tabel penjualan_harian.

Pelaksanaan pemrograman prosedural misalnya seperti di Script 1 berikut ini (dalam bentuk stored produre).

```
DELIMITER // ;
DROP PROCEDURE IF EXISTS prosedural_penj_har //
CREATE PROCEDURE prosedural_penj_har()
BEGIN
    DECLARE done INT(1) DEFAULT 0;
    DECLARE counter INT(10) DEFAULT 0;
    DECLARE kp1 INT(4) DEFAULT 0;
    DECLARE kp INT(4) DEFAULT 0;
    DECLARE bp1 DECIMAL(8,2) DEFAULT 0;
    DECLARE bp DECIMAL(8,2) DEFAULT 0;
    DECLARE d INT(2) DEFAULT 0;

    DECLARE penj_cursor CURSOR FOR SELECT * FROM transaksi_penjualan ORDER BY kode_produk;

    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

    OPEN penj_cursor;

    REPEAT
        FETCH penj_cursor INTO kp, bp, d;
        SET counter = counter + 1;

        IF NOT done THEN
            IF counter = 1 THEN
                SET bp1 = ( bp * (100-d)/100 );
                SET kp1 = kp;
            ELSEIF (kp < kp1) THEN
                INSERT INTO penjualan_harian
                    VALUES(kp1, bp1);
                SET bp1 = ( bp * (100-d)/100 );
                SET kp1 = kp;
            ELSE SET bp1 = bp1 + (bp * (100-d)/100);
            END IF;
        END IF;

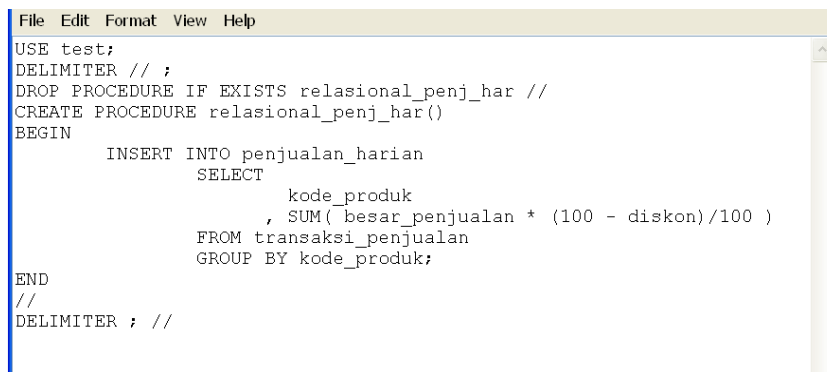
        IF done THEN INSERT INTO penjualan_harian
            VALUES(kp1, bp1);
        END IF;

    UNTIL done END REPEAT;

    CLOSE penj_cursor;
END
//
DELIMITER ; //
```

Script 1 Program prosedural (nama stored procedurenya adalah *prosedur_penj_har*)

Terlihat, program prosedural ini memproses transaksi row demi row dengan menggunakan cursor. Sedang pemrograman relasional adalah sebagai berikut (Script 2)

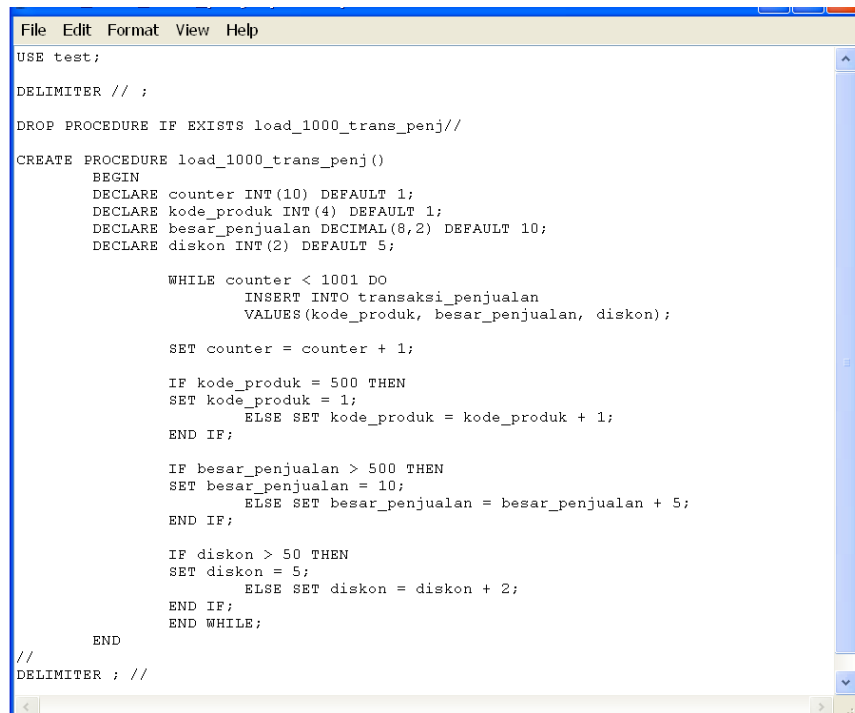


```
File Edit Format View Help
USE test;
DELIMITER // ;
DROP PROCEDURE IF EXISTS relasional_penj_har //
CREATE PROCEDURE relasional_penj_har()
BEGIN
    INSERT INTO penjualan_harian
        SELECT
            kode_produk
            , SUM( besar_penjualan * (100 - diskon)/100 )
        FROM transaksi_penjualan
        GROUP BY kode_produk;
END
//
DELIMITER ; //
```

Script 2 Program relasional (nama stored procedurenya adalah *relasional_penj_har*).

Terlihat betapa jauh lebih sederhananya yang program relasional dibanding prosedural, karena yang relasional memanfaatkan teknologi databasenya dengan benar, meminta databasenya untuk melaksanakan kerumitan langkah-langkah dalam bentuk perintah SQL yang sederhana.

Untuk mengadu kecepatan dan memahami karakteristik kedua jenis program diatas, kita gunakan tiga macam volume data transaksi, yaitu 1000, 100000, dan 1000000. Script 3 adalah stored procedure untuk mengisikan 1000 transaksi. Untuk volume 100000 dan 1000000, parameter counter didalam stored procedure ini disesuaikan dengan kedua volume ini.



```
File Edit Format View Help
USE test;

DELIMITER // ;

DROP PROCEDURE IF EXISTS load_1000_trans_penj//

CREATE PROCEDURE load_1000_trans_penj()
BEGIN
    DECLARE counter INT(10) DEFAULT 1;
    DECLARE kode_produk INT(4) DEFAULT 1;
    DECLARE besar_penjualan DECIMAL(8,2) DEFAULT 10;
    DECLARE diskon INT(2) DEFAULT 5;

    WHILE counter < 1001 DO
        INSERT INTO transaksi_penjualan
        VALUES(kode_produk, besar_penjualan, diskon);

        SET counter = counter + 1;

        IF kode_produk = 500 THEN
            SET kode_produk = 1;
        ELSE SET kode_produk = kode_produk + 1;
        END IF;

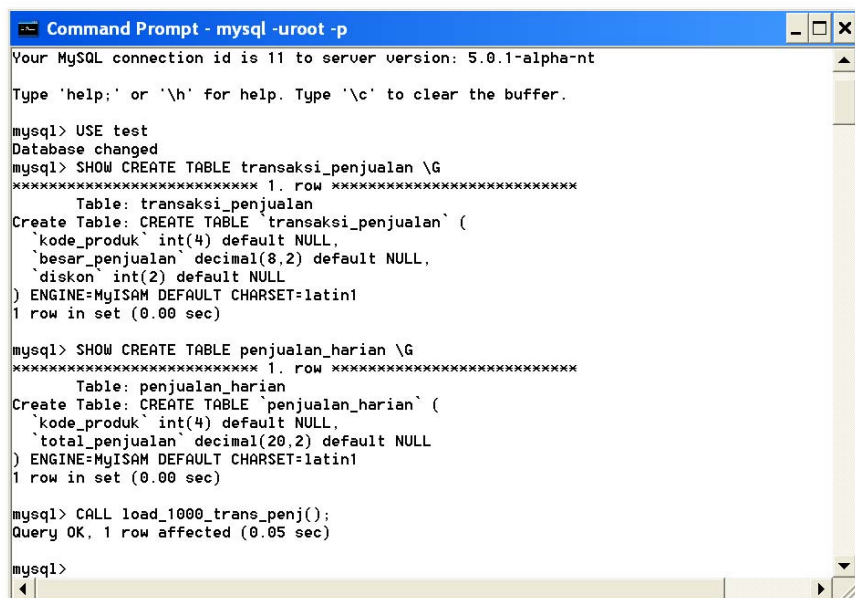
        IF besar_penjualan > 500 THEN
            SET besar_penjualan = 10;
        ELSE SET besar_penjualan = besar_penjualan + 5;
        END IF;

        IF diskon > 50 THEN
            SET diskon = 5;
        ELSE SET diskon = diskon + 2;
        END IF;
    END WHILE;
END

//
DELIMITER ; //
```

Script 3 Stored procedure untuk loading 1000 data transaksi.

Semua script diatas harus di-compile terlebih dahulu sebelum dapat dijalankan. Contoh meng-compile dan menjalankannya (*call* statement) adalah seperti dicontohkan di Gambar 3 dibawah ini untuk *load_1000_trans_penj*.



```
Command Prompt - mysql -uroot -p
Your MySQL connection id is 11 to server version: 5.0.1-alpha-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE test
Database changed
mysql> SHOW CREATE TABLE transaksi_penjualan \G
***** 1. row *****
      Table: transaksi_penjualan
Create Table: CREATE TABLE `transaksi_penjualan` (
  `kode_produk` int(4) default NULL,
  `besar_penjualan` decimal(8,2) default NULL,
  `diskon` int(2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

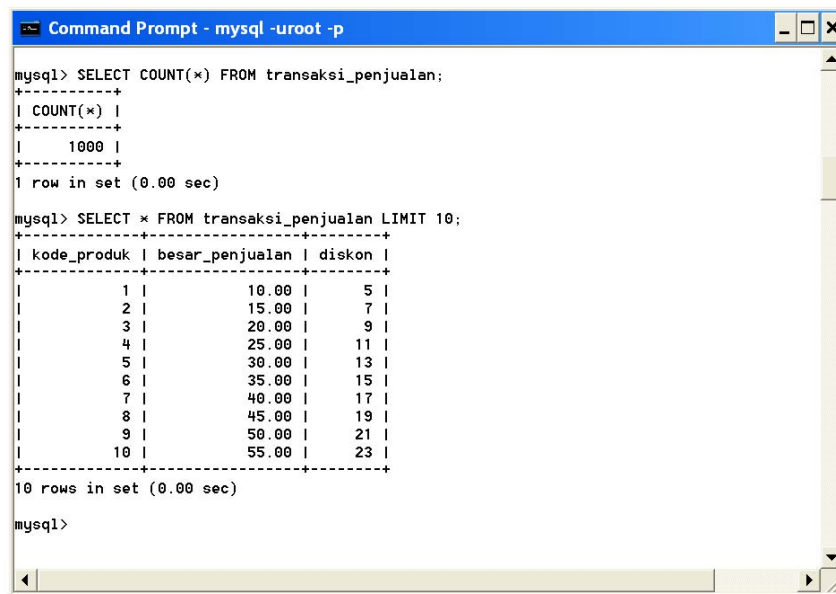
mysql> SHOW CREATE TABLE penjualan_harian \G
***** 1. row *****
      Table: penjualan_harian
Create Table: CREATE TABLE `penjualan_harian` (
  `kode_produk` int(4) default NULL,
  `total_penjualan` decimal(20,2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql> CALL load_1000_trans_penj();
Query OK, 1 row affected (0.05 sec)

mysql>
```

Gambar 3 Compile dan Call stored procedure *load_1000_trans_penj*.

Pastikan data sudah masuk kedalam tabel, misalnya dengan 2 buah query seperti di Gambar 4 berikut (banyaknya data dan contoh isinya)



```
Command Prompt - mysql -uroot -p

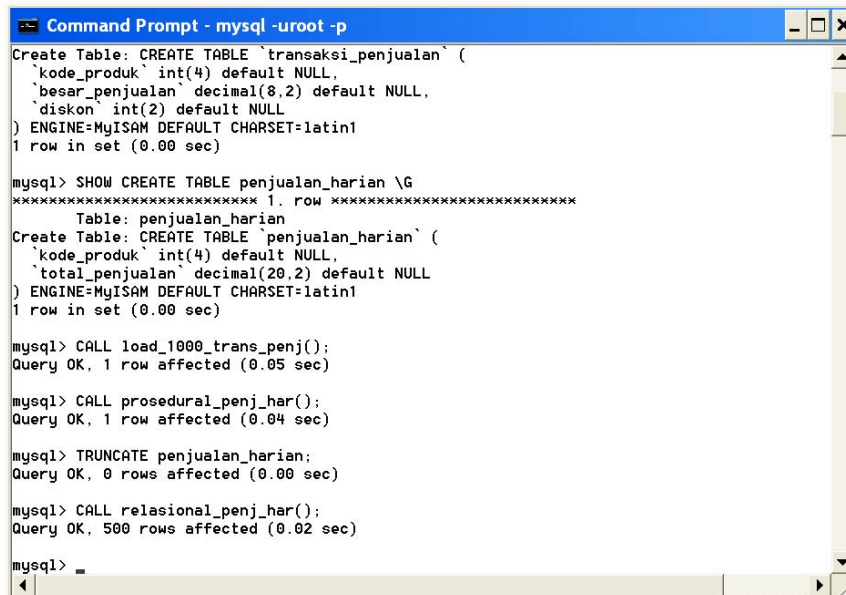
mysql> SELECT COUNT(*) FROM transaksi_penjualan;
+-----+
| COUNT(*) |
+-----+
|      1000 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM transaksi_penjualan LIMIT 10;
+-----+-----+-----+
| kode_produk | besar_penjualan | diskon |
+-----+-----+-----+
| 1 | 10.00 | 5 |
| 2 | 15.00 | 7 |
| 3 | 20.00 | 9 |
| 4 | 25.00 | 11 |
| 5 | 30.00 | 13 |
| 6 | 35.00 | 15 |
| 7 | 40.00 | 17 |
| 8 | 45.00 | 19 |
| 9 | 50.00 | 21 |
| 10 | 55.00 | 23 |
+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

Gambar 4 Memastikan data yang diinginkan sudah masuk kedalam tabel transaksi_penjualan.

Berikut eksekusi program prosedural dan relasional untuk 1000 buah transaksi dan kecepatan keduanya. Setelah ekeekusi prosedural, tabel penjualan_harian dikosongkan terlebih dahulu sebelum eksekusi relasional agar pengukuran setara (keduanya mulai dari tabel penjualan_harian yang kosong).



```
Command Prompt - mysql -uroot -p

Create Table: CREATE TABLE `transaksi_penjualan` (
  `kode_produk` int(4) default NULL,
  `besar_penjualan` decimal(8,2) default NULL,
  `diskon` int(2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE penjualan_harian \G
+-----+-----+-----+
| Table: penjualan_harian |
+-----+-----+-----+
Create Table: CREATE TABLE `penjualan_harian` (
  `kode_produk` int(4) default NULL,
  `total_penjualan` decimal(20,2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql> CALL load_1000_trans_penj();
Query OK, 1 row affected (0.05 sec)

mysql> CALL prosedural_penj_har();
Query OK, 1 row affected (0.04 sec)

mysql> TRUNCATE penjualan_harian;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL relasional_penj_har();
Query OK, 500 rows affected (0.02 sec)

mysql>
```

Gambar 5 Perbandingan kecepatan prosedural dan relasional untuk 1000 transaksi penjualan.

Berikut berturut-turut pelaksanaan pengukuran kecepatan untuk volume 100000 dan 1000000.

```
mysql> TRUNCATE transaksi_penjualan;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> TRUNCATE penjualan_harian;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> CALL load_100000_trans_penj();  
Query OK, 1 row affected (2.20 sec)  
  
mysql> CALL prosedural_penj_har();  
Query OK, 1 row affected (1.18 sec)  
  
mysql> TRUNCATE penjualan_harian;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> CALL relasional_penj_har();  
Query OK, 500 rows affected (0.27 sec)  
  
mysql> TRUNCATE transaksi_penjualan;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> TRUNCATE penjualan_harian;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> CALL load_1000000_trans_penj();  
Query OK, 1 row affected (31.90 sec)  
  
mysql> CALL prosedural_penj_har();  
Query OK, 1 row affected (42.33 sec)  
  
mysql> TRUNCATE penjualan_harian;  
Query OK, 0 rows affected (0.70 sec)  
  
mysql> CALL relasional_penj_har();  
Query OK, 500 rows affected (1.72 sec)
```

Gambar 6 Perbandingan kecepatan prosedural dan relasional untuk 10000 dan 1000000 transaksi penjualan.

Ringkasan pengukuran kecepatan untuk ketiga macam volume ada di Tabel 1 berikut.

Tabel 1 Ringkasan kecepatan Prosedural dan Relasional untuk volume transaksi 1000, 100000, dan 1000000.

VOLUME	PROSEDURAL	RELASIONAL
1000	0.04	0.02
100000	1.18	0.27
1000000	42.33	1.72

Terlihat bahwa makin besar volume transaksi (banyaknya baris data didalam tabel) makin nyata keunggulan kecepatan program relasional.

Kesimpulan

Optimumkan keampuhan teknologi relasional didalam database melalui pemrograman teknik relasional, setidaknya untuk kemudahan pemeliharaan dan kecepatan ekekusi program dibanding teknik prosedural, terutama untuk pengolahan data volume besar.