KONSEP MEMBANGUN APLIKASI MULTIPLATFORM DENGAN OPTIMALISASI PENGGUNAAN VIEW, FUNCTION DAN TRIGGER PADA RDBMS POSTGRESQL

Joko Triyono
Jurusan Teknik Informatika, Fakultas Teknologi Industri,
Institut Sains & Teknologi AKPRIND Yogyakarta
Jl. Kalisahak No.28, Komplek Balapan Yogyakarta, 55222
E-mail: zainjack@gmail.com

Abstrak

Perkembangan Teknologi informasi mengalami lompatan yang luar biasa dasawarsa ini, dengan semakin pesat perkembangan bahasa pemrograman, database serta sistem operasi menyebabkan ada ketimpangan dalam implementasi di dunia nyata. Beberapa sistem informasi yang telah di bangun akan menjadi sangat ketinggalan ketika teknologi yang baru muncul, kadangkala model algoritma yang ada juga berbeda sehingga perlu di lakukan penyesuaian agar sistem bisa kompatibel, belum lagi dengan perkembangan device dan software yang luar biasa cepatnya. RDBMS saat ini sangat terbuka dan bisa di akses oleh software aplikasi apapun sehingga perlu di lakukan standarisasi dalam transaksi dan view agar sesuai dengan aturan bisnis. Optimalisasi dalam penelitian ini dilakukan dengan membebankan semaksimal mungkin algoritma transaksi di lakukan di sisi RDBMS, sehingga aplikasi akan melakukan seminimal mungkin perintah SQL, dengan mengkombinasikan penggunaan view, function dan trigger secara tepat. Dengan metode ini akan memudahkan bagi pengembang aplikasi, karena algoritma telah dilakukan di sisi rdbms, sehingga menggunakan software apapun akan didapatkan algoritma yang sama.

Kata Kunci : algoritma; aplikasi; database; multiplatform; optimalisasi

Pendahuluan

Sistem informasi menjadi bagian yang tidak terpisahkan dalam semua sisi kehidupan saat ini, dengan sistem informasi yang tepat maka bisa diwujudkan efektifitas kerja yang lebih baik. Tidak terjadinya duplikasi-duplikasi informasi dan menjadi lebih terstruktur dan tertata dalam penyajian informasi kepada khalayak dengan media view yang berbeda-beda baik itu berbasis *web*, *smartphone* maupun *device* yang lain. Hampir di semua sektor kegiatan, sistem informasi menjadi tolok ukur dalam menilai atau mengukur efektifitas kerja.

Dengan kondisi seperti saat ini, kekuatan sistem informasi sangat menentukan, sistem informasi yang baik pasti akan di dukung oleh sebuah atau lebih *database* yang baik. Dimana *database* merupakan bagian yang tidak terpisahkan dari keberadaan sistem informasi. Perubahan-perubahan teknologi dalam pemrogaman akan menyebabkan sering berpindahnya atau digantikannya sistem informasi ke sistem informasi yang lebih baru yang tentunya juga akan berpengaruh terhadap keberadaan *database* yang mendukung sistem informasi. Teknologi *database* tidak secepat perkembangan teknologi sistem informasi, tetapi *database* yang baik akan selalu *support* terhadap perubahan teknologi sistem informasi dengan menerapkan teknik-teknik tertentu.

Dalam penelitian ini akan di paparkan optimalisasi dengan membebankan semaksimal mungkin algoritma transaksi di lakukan di sisi RDBMS, sehingga aplikasi akan melakukan seminimal mungkin perintah SQL. Penggunaan view, function dan trigger diharapkan akan memudahkan bagi pengembang aplikasi, karena algoritma telah dilakukan di sisi rdbms, sehingga menggunakan software apapun akan didapatkan algoritma yang sama.

Metode Penelitian

Tulisan ini dilakukan dalam skala laboratorium di Laboratorium Jaringan dan Internet Institut Sains dan Teknologi AKPRIND Yogyakarta, model atau model yang digunakan dalam penelitian ini adalah sebuah database pembelian dari sebuah toko, dimana toko ini melakukan pembelian produk untuk menambah stok dari beberapa pemasok dan menjual produk ke pelanggan secara langsung, sistem informasi digunakan sebagai interface dari para pemegang kendali baik itu bagian penjualan, bagian pembelian maupun manager.

Kebutuhan Sistem

Bahan dan alat yang dibutuhkan untuk penelitian ini meliputi hardware dan *software*, diantaranya yaitu:

- Hardware, laptop Lenovo dengan spesifikasi Intel® Core™ i5-5200U CPU @ 2.20GHz × 4 Ram 7,7 GiB 64bit Hardisk 500GB.
- Software Sistem operasi Ubuntu 16.04 LTS.
- Software Web Server, PHP
- Database PostgreSQL 9.5.4

Metode Pengumpulan Data

Metode yang digunakan dalam pengumpulan data pada penelitian ini terdiri dari beberapa metode, yaitu:
Metode Observasi, Metode observasi ini digunakan untuk pengumpulan data dengan pengamatan secara langsung maupun tidak langsung terhadap obyek yang diteliti.
Metode Studi Kepustakaan, Metode studi kepustakaan merupakan sebuah cara dalam pengumpulan data dengan mempelajari bahan pustaka baik berupa dokumen tertulis ataupun berupa gambar dengan membandingkan beberapa referensi.
Metode Eksperimen, Metode ini digunakan dengan mengadakan uji coba dan simulasi yang telah dibuat menggunakan RDBMS PostgreSQL, menguji secara langsung melalui terminal (SQL Manipulation) maupun menggunakan aplikasi berbasis web menggunakan PHP.

Perancangan Sistem

Peraturan Bisnis

Peraturan bisnis pada obyek penelitian adalah bahwa semua transaksi baik itu penjualan mupun pembelian barang harus dilakukan di RDBMS dan menggunakan aturan dasar hanya menjual barang yang ada di stok sedangkan untuk pembelian, maka bisa dilakukan penambahan produk jika pada saat dilakukan pembelian tidak diketemukan produk tersebut. Transaksi dilakukan secara langsung terhadap tabel paling bawah dan sinkronisasi terhadap tabel-tabel yang terkait digunakan *function* dan *trigger*. Penampilan data atau *select* digunakan *view* sehingga data yang diperoleh dari beberapa tabel harus sudah di olah menjadi sebuah *view*.

Desain Sistem

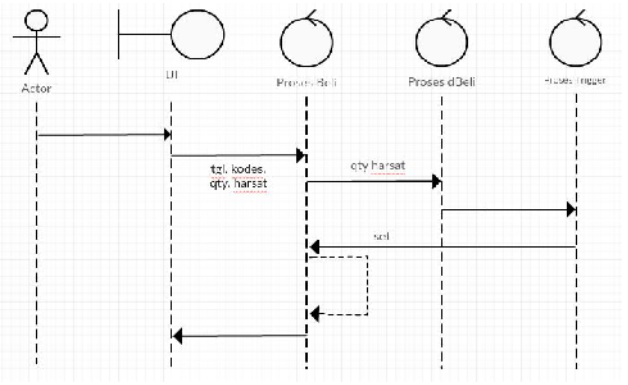
Berdasarkan peraturan bisnis yang telah dijelaskan, maka dalam tulisan ini di implementasikan dalam sebuah gambar design seperti yang di tunjukkan pada gambar 2.

Dalam penelitian ini akan di bangun:

- Database dan tabel-tabel yang digunakan
- Penentuan relational databases
- Pembuatan *view*
- Pembuatan *function* dan *procedure*
- Pembuatan *trigger* untuk transaksi pembelian dan penjualan

Sequence Diagram

Sequence Diagram digunakan untuk lebih menjelaskan proses-proses yang terjadi pada transaksi sistem. *Sequence Diagram* proses pembelian barang meliputi proses penambahan, pengubahan dan penghapusan pembelian seperti pada gambar 1.



Gambar 1 *Sequence Diagram* Pembelian

Aktivitas yang terjadi pada gambaran *sequence diagram* pembelian bisa dijelaskan sebagai berikut:

Transaksaksi Pembelian

1. Aktor melakukan transaksi di *User Interface* (UI), dengan mengirimkan ke proses beli berupa tgl, kode pemasok, qty, jumlah, harga dan kode barang

2. Pada proses beli, di masukkan ke tabel berupa tgl, kode pemasok dan di dapatkan id beli, id beli diteruskan ke proses dbeli bersama qty, jumlah, harga dan kode barang.
3. Pada proses dbeli akan dilakukan transaksi ke table dbeli, setelah di lakukan transaksi dbeli maka akan diteruskan ke *trigger after insert*
4. Proses *Trigger After Insert*, pada proses ini akan dilakukan update tabel beli berdasarkan new.idbeli terhadap qty dan jumlah.

Transaksi Update Pembelian

1. Aktor melakukan transaksi di *User Interface* (UI), dengan mengirimkan ke proses dbeli berupa idbeli, qty, jumlah, harga dan kode barang
2. Pada proses dbeli akan dilakukan transaksi update ke table dbeli, setelah di lakukan transaksi dbeli maka akan diteruskan ke *trigger after Update*
3. Proses *Trigger After Update*, pada proses ini akan dilakukan update tabel beli berdasarkan new.idbeli terhadap qty dan jumlah dengan cara data yang ada di tabel beli di kurangi dengan data lama yang diupdate di tabel dbeli lalu dilanjutkan dengan update dengan data baru.

Transaksi Delete Pembelian

1. Aktor melakukan transaksi di *User Interface* (UI), dengan mengirimkan ke proses dbeli berupa idbeli, dan kode barang
2. Pada proses dbeli akan dilakukan transaksi delete ke table dbeli, sebelum di lakukan transaksi dbeli maka akan diteruskan ke *trigger Before Deleted*
3. Proses *Trigger Before Delete*, pada proses ini akan dilakukan update tabel beli berdasarkan idbeli terhadap qty dan jumlah dengan cara : data yang ada di tabel beli di kurangi dengan data lama yang didelete di dbeli.

Perancangan Database

Perancangan Tabel

Tabel-tabel yang digunakan dalam penelitian ini terdiri dari 4 buah tabel yang saling berelasi, berikut tabel-tabel yang digunakan.

Tabel Supplier

Tabel ini untuk menampung data-data supplier yang menjadi pemasok di toko dengan *primary key* kodes.

```
              Table "public.supplier"
     Column     |         Type          | Modifiers
----------------+-----------------------+-----------
 kodes          | character varying(5)  | not null
 nama           | character varying(50) |
 alamat         | character varying(50) |
 totaltransaksi | numeric               | default 0
Indexes:
    "supplier_pkey" PRIMARY KEY, btree (kodes)
Referenced by:
    TABLE "beli" CONSTRAINT "beli_kodes_fkey" FOREIGN KEY (kodes) REFERENCES supplier(kodes)
```

Tabel Produk

Tabel ini untuk menampung data-data produk yang dijual dengan sebuah *primary key* kode.

```
             Table "public.produk"
   Column   |         Type          | Modifiers
------------+-----------------------+-----------
 kode       | character varying(5)  | not null
 nama       | character varying(50) |
 satuan     | character varying(15) |
 hargajual  | numeric               | default 0
 jual       | integer               | default 0
 beli       | integer               | default 0
 stokonhand | integer               | default 0
Indexes:
    "produk_pkey" PRIMARY KEY, btree (kode)
Referenced by:
    TABLE "dbeli" CONSTRAINT "dbeli_kode_fkey" FOREIGN KEY (kode) REFERENCES produk(kode)
```

Tabel Beli

Table Beli menampung data induk pembelian dengan id serial sebagai *primary key* dan memiliki *foreign key* kodes yang bereferensi ke tabel supplier.

```
                              Table "public.beli"
 Column |         Type         |                     Modifiers
--------+----------------------+---------------------------------------------------
 id     | integer              | not null default nextval('beli_id_seq'::regclass)
 tgl    | date                 |
 qty    | integer              | default 0
```

```
 jumlah | numeric              | default 0
 kodes  | character varying(5) |
Indexes:
    "beli_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "beli_kodes_fkey" FOREIGN KEY (kodes) REFERENCES supplier(kodes)
Referenced by:
    TABLE "dbeli" CONSTRAINT "dbeli_idb_fkey" FOREIGN KEY (idb) REFERENCES beli(id)
```

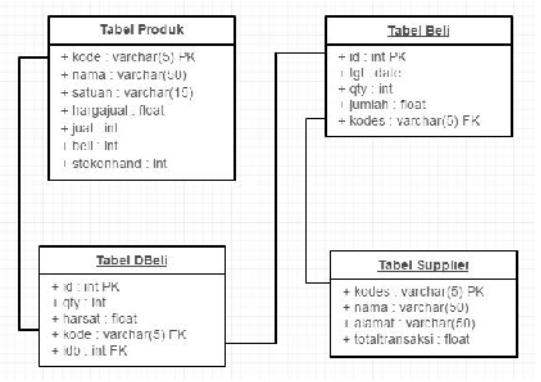
Tabel dBeli

Tabel dBeli menampung data detail pembelian dengan id serial sebagai primary key dan memiliki *foreign key* kode yang mereferensi ke tabel produk serta *foreign key idb* yang mereferensi ke tabel beli.

```
                                Table "public.dbeli"
 Column |         Type         |                      Modifiers
--------+----------------------+-----------------------------------------------------
 id     | integer              | not null default nextval('dbeli_id_seq'::regclass)
 qty    | integer              | default 0
 harsat | numeric              | default 0
 kode   | character varying(5) |
 idb    | integer              |
Indexes:
    "dbeli_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "dbeli_idb_fkey" FOREIGN KEY (idb) REFERENCES beli(id)
    "dbeli_kode_fkey" FOREIGN KEY (kode) REFERENCES produk(kode)
Triggers:
    tr_beli_detail AFTER INSERT OR DELETE OR UPDATE ON dbeli FOR EACH ROW EXECUTE PROCEDURE
f_beli_detail()
```

Perancangan Diagram Relational

Rancangan ini untuk menjelaskan hubungan antar tabel yang terdapat dalam sistem toko. Gambar 2 menjelaskan tentang diagram relasional database.



Gambar 2. Relational Database

Perancangan *VIEW*

View digunakan untuk mengurangi akses UI ke tabel secara langsung, sehingga algoritma dari relasi antar tabel yang cukup panjang akan di sederhanakan oleh *view*.

View **vbeli**, *view* ini digunakan untuk menampilkan data suplier, produk dan kapan terjadi pembelian yang di dapatkan dari tabel supplier, produk, beli dan dbeli.

```
create view vbeli as
select supplier.kodes,supplier.nama as namapemasok,produk.kode,produk.nama, beli.tgl,
dbeli.harsat,dbeli.qty
from supplier, produk, beli, dbeli
where supplier.kodes=beli.kodes and dbeli.kode=produk.kode and beli.id=dbeli.idb;
```

View ***vtotalbelipertgl,*** *view ini digunakan untuk menampilkan data rangkuman pembelian per tanggal yang didapat dari group sum tabel beli.*

```
create view vtotalbelipertgl as
select tgl,sum(jumlah) as total from beli group by tgl order by tgl;
```

Perancangan *Trigger* Pembelian

Untuk merancang *trigger* pembelian harus di buat *function* dulu seperti berikut ini. Pada function *f_beli_detail* terbagi menjadi 3 keadaan yaitu INSERT, UPDATE dan DELETE yang di handle oleh TG_OP. Pada saat INSERT, maka proses yang di lakukan adalah update tabel beli dan update produk.
sedangkan pada saat UPDATE maka proses yang dilakukan ada dua yaitu mengurangi data di tabel beli dengan data lama lalu mengupdate data di tabel beli dengan data yang baru saja di update, berikutnya mengurangi data di tabel produk dengan data lama dan mengupdate data di tabel produk dengan data yang baru saja di update.
sedangkan pada saat DELETE maka dilakukan update pada tabel beli dan tabel produk dengan data yang akan di deleted.

```
CREATE OR REPLACE FUNCTION f_beli_detail()
    RETURNS trigger AS
    $$
    BEGIN
    IF TG_OP = 'INSERT' THEN
        update beli set jumlah = jumlah + (NEW.qty * NEW.harsat), qty = qty + New.qty
                where id = NEW.idb;
         update produk set beli = beli + NEW.qty, stokonhand = stokonhand + NEW.qty
                where kode = NEW.kode;
        RETURN NEW;
    END IF;
    IF TG_OP = 'UPDATE' THEN
        update beli set jumlah = jumlah - (OLD.qty * OLD.harsat),
         qty = qty - OLD.qty where id = OLD.idb;
        update beli set jumlah = jumlah + (NEW.qty * NEW.harsat),
         qty = qty + New.qty where id = OLD.idb;
         update produk set beli = beli - OLD.qty, stokonhand = stokonhand - OLD.qty
                where kode = OLD.kode;
         update produk set beli = beli + NEW.qty, stokonhand = stokonhand + NEW.qty
                where kode = OLD.kode;
        RETURN NEW;
    END IF;
    IF TG_OP = 'DELETE' THEN
  update beli set jumlah = jumlah - (OLD.qty * OLD.harsat),
         qty = qty - OLD.qty where id = OLD.idb;
   update produk set beli = beli - OLD.qty, stokonhand = stokonhand - OLD.qty
                where kode = OLD.kode;
    RETURN OLD;
    END IF;
    END;
    $$
    language plpgsql ;
```

Setelah *function* terbentuk, maka *trigger* baru di buat

```
CREATE TRIGGER tr_beli_detail AFTER INSERT OR UPDATE OR DELETE ON dbeli FOR EACH ROW
EXECUTE PROCEDURE f_beli_detail();
```

Hasil dan Pembahasan

Sistem database yang telah diimplementasikan di RDBMS baik itu tabel, refference antar tabel, view dan trigger serta function perlu diuji coba untuk mengetahui validasi dari fungsi-fungsi yang telah di rencanakan. Pengujian di lakukan melalui UI terminal psql, dan membandingkan dengan cara tanpa *view, trigger* dan *function.*

Pengujian *View*

View vbeli, setelah di eksekusi dengan perintah select, maka akan di dapatkan list hasil select.

```
toko=# select * from vbeli;
 kodes | namapemasok | kode |        nama        |    tgl     | harsat | qty
-------+-------------+------+--------------------+------------+--------+-----
 s01   | Bejo        | 001  | Jahe Merah Kering  | 2016-10-01 |  20000 |   1
 s02   | Budiman     | 002  | Jahe Merah Basah   | 2016-10-01 |  15000 |   1
 s03   | Anwar       | 003  | Jahe Emprit Basah  | 2016-10-10 |  10000 |  10
 s04   | Irsyad      | 003  | Jahe Emprit Basah  | 2016-10-10 |  10000 |  10
 s04   | Irsyad      | 005  | Jahe Emprit Kering | 2016-10-10 |  10000 |   5
 s04   | Irsyad      | 005  | Jahe Emprit Kering | 2016-10-10 |  10000 |   2
 s01   | Bejo        | 002  | Jahe Merah Basah   | 2016-10-01 |  21000 |  10
 s01   | Bejo        | 002  | Jahe Merah Basah   | 2016-10-01 |  21000 |  10
(8 rows)
```

Akan menghasilkan output yang sama jika dibandingkan dengan perintah langsung yang cukup panjang dan komplek.

```
toko=# select supplier.kodes,supplier.nama as
       namapemasok,produk.kode,produk.nama,beli.tgl,dbeli.harsat,dbeli.qty
       from supplier, produk, beli, dbeli
       where supplier.kodes=beli.kodes and dbeli.kode=produk.kode and beli.id=dbeli.idb;

 kodes | namapemasok | kode |        nama        |    tgl     | harsat | qty
-------+-------------+------+--------------------+------------+--------+-----
 s01   | Bejo        | 001  | Jahe Merah Kering  | 2016-10-01 |  20000 |   1
 s02   | Budiman     | 002  | Jahe Merah Basah   | 2016-10-01 |  15000 |   1
 s03   | Anwar       | 003  | Jahe Emprit Basah  | 2016-10-10 |  10000 |  10
 s04   | Irsyad      | 003  | Jahe Emprit Basah  | 2016-10-10 |  10000 |  10
 s04   | Irsyad      | 005  | Jahe Emprit Kering | 2016-10-10 |  10000 |   5
 s04   | Irsyad      | 005  | Jahe Emprit Kering | 2016-10-10 |  10000 |   2
 s01   | Bejo        | 002  | Jahe Merah Basah   | 2016-10-01 |  21000 |  10
 s01   | Bejo        | 002  | Jahe Merah Basah   | 2016-10-01 |  21000 |  10
(8 rows)
```

Dengan menerapkan view vbeli, perintah SQL yang semula cukup panjang bisa di gantikan dengan perintah yang singkat dengan memanggil nama view yang di buat.. Dengan demikian proses select data bisa dilakukan dengan cukup memanggil nama *view.*

```
create view vbeli as
select supplier.kodes,supplier.nama as
namapemasok,produk.kode,produk.nama,beli.tgl,dbeli.harsat,dbeli.qty
from supplier, produk, beli, dbeli
where supplier.kodes=beli.kodes and dbeli.kode=produk.kode and beli.id=dbeli.idb;
```

View ***vtotalbelipertgl, V****iew ini digunakan* setelah di eksekusi dengan perintah select, maka akan di dapatkan list hasil select berupa jumlah pembelian pertanggal.

```
toko=# Select * from vtotalbelipertgl;
    tgl     | total
------------+--------
 2016-10-01 | 455000
 2016-10-10 | 270000
(2 rows)
```

Akan menghasilkan output yang sama jika dibandingkan dengan perintah langsung yang cukup panjang dan komplek.

```
toko=# select tgl,sum(jumlah) as total from beli group by tgl order by tgl;
    tgl     | total
------------+--------
 2016-10-01 | 455000
 2016-10-10 | 270000
(2 rows)
```

Dengan menerapkan view vtotalbelipertgl, perintah SQL yang semula cukup panjang bisa di gantikan dengan perintah yang singkat dengan memanggil nama view yang di buat.. Dengan demikian proses select data bisa dilakukan dengan cukup memanggil nama *view.*

```
create view vtotalbelipertgl as
select tgl,sum(jumlah) as total from beli group by tgl order by tgl;
```

Pengujian Trigger

Dalam pengujian trigger ini, akan diuji dari INSERT, UPDATE dan DELETE. Akan diuji pada produk dengan kode '**002**' dengan melakukan penambahan pembelian di untuk kode tersebut dengan beli.id = 2.

```
toko=# select * from produk;
 kode |        nama        | satuan | hargajual | jual | beli | stokonhand
------+--------------------+--------+-----------+------+------+------------
...
 005  | Jahe Emprit Kering | kg     |         0 |    0 |    7 |          7
 002  | Jahe Merah Basah   | kg     |         0 |    0 |   21 |         21  <-- Obyek
(4 rows)

toko=# select * from beli;
 id |    tgl     | qty | jumlah | kodes
```

```
----+------------+-----+--------+-------
  2 | 2016-10-01 |   1 |  15000 | s02 <-- obyek
  4 | 2016-10-10 |   3 | 170000 | s04
...
(4 rows)

toko=# select * from dbeli;
 id | qty | harsat | kode | idb
----+-----+--------+------+-----
  1 |   1 |  20000 | 001  |   1
...
  8 |  10 |  21000 | 002  |   1
  9 |  10 |  21000 | 002  |   1
(8 rows)
```

Perintah SQL

```
toko=# insert into dbeli(qty,harsat,kode,idb) values(5,15000,'002',2);
INSERT 0 1
```

Hasil setelah terjadi *INSERT*

Setelah di lakukan perintah INSERT pada table dbeli, maka terlihat pada tabel produk akan bertambah dari 21 menjadi 26 (bertambah 5), pada tabel beli untuk beli.id=2 qty menjadi 6 setelah terjadi penambahan 5 dan di tabel dbeli terjadi penambahan record pada id=10.

Dari percobaan ini terbukti bahwa *trigger* berjalan sempurna.

```
toko=# select * from produk;
 kode |        nama        | satuan | hargajual | jual | beli | stokonhand
------+--------------------+--------+-----------+------+------+------------
...
 005  | Jahe Emprit Kering | kg     |         0 |    0 |    7 |          7
 002  | Jahe Merah Basah   | kg     |         0 |    0 |   26 |         26  <-- obyek
(4 rows)

toko=# select * from beli;
 id |    tgl     | qty | jumlah | kodes
----+------------+-----+--------+-------
...
  1 | 2016-10-01 |  21 | 440000 | s01
  2 | 2016-10-01 |   6 |  90000 | s02  <-- obyek
(4 rows)

toko=# select * from dbeli;
 id | qty | harsat | kode | idb
----+-----+--------+------+-----
  1 |   1 |  20000 | 001  |   1
...
  8 |  10 |  21000 | 002  |   1
  9 |  10 |  21000 | 002  |   1
 10 |   5 |  15000 | 002  |   2  <-- obyek
(9 rows)
```

Pengujian *Update*

Dengan menggunakan data dan obyek yang sama diatas, akan dilakukan update data pembelian untuk dbeli.id=10 qty dari 5 menjadi 10. Dari hasil pengujian terlihat bahwa beli.id2 qty = 11, produk.beli = 31

```
toko=# update dbeli set qty=10 where id=10;
UPDATE 1
toko=# select * from dbeli;
 id | qty | harsat | kode | idb
----+-----+--------+------+-----
  1 |   1 |  20000 | 001  |   1
...
  8 |  10 |  21000 | 002  |   1
  9 |  10 |  21000 | 002  |   1
 10 |  10 |  15000 | 002  |   2 <-- obyek
(9 rows)

toko=# select * from beli;
 id |    tgl     | qty | jumlah | kodes
----+------------+-----+--------+-------
...
  1 | 2016-10-01 |  21 | 440000 | s01
```

```
  2 | 2016-10-01 |  11 | 165000 | s02 <-- obyek
(4 rows)

toko=# select * from produk;
 kode |        nama        | satuan | hargajual | jual | beli | stokonhand
------+--------------------+--------+-----------+------+------+------------
...
 005  | Jahe Emprit Kering | kg     |         0 |    0 |    7 |          7
 002  | Jahe Merah Basah   | kg     |         0 |    0 |   31 |         31 <-- obyek
(4 rows)
```

Pengujian *Delete*

Dengan menggunakan data dan obyek yang sama diatas, akan dilakukan *delete* data pembelian untuk dbeli.id=10. Setelah dilakukan delete maka record di dbeli dengan id=10 sudah tidak ada, beli.id=2 qty menjadi 1 dan produk.kode=002 beli menjadi 21.

```
toko=# delete from dbeli where id=10;
DELETE 1
toko=# select * from dbeli;
 id | qty | harsat | kode | idb
----+-----+--------+------+-----
  1 |   1 |  20000 | 001  |   1
...
  8 |  10 |  21000 | 002  |   1
  9 |  10 |  21000 | 002  |   1
(8 rows)

toko=# select * from beli;
 id |    tgl     | qty | jumlah | kodes
----+------------+-----+--------+-------
...
  1 | 2016-10-01 |  21 | 440000 | s01
  2 | 2016-10-01 |   1 |  15000 | s02 <--obyek
(4 rows)

toko=# select * from produk;
 kode |        nama        | satuan | hargajual | jual | beli | stokonhand
------+--------------------+--------+-----------+------+------+------------
...
 005  | Jahe Emprit Kering | kg     |         0 |    0 |    7 |          7
 002  | Jahe Merah Basah   | kg     |         0 |    0 |   21 |         21 <--obyek
(4 rows)
```

Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa, dengan melakukan optimalisasi database akan memungkinkan para pengembang sistem informasi tidak terlalu direpotkan dengan algoritma SQL, karena banyak pekerjaan yang bisa di bebankan ke RDBMS, baik itu untuk transaksional maupun untuk view. Dengan metode ini, apapun bahasa pemrograman yang digunakan tetap akan mengakses SQL yang sangat sederhana, sehingga akan memudahkan dalam pengembangan sistem informasi dan tetap akan mendapatkan algoritma database yang sama.

Saran untuk memperbaiki penelitian ini adalah seberapa besar beban yang di tanggung oleh server RDBMS jika beban transaksional dan view di letakkan di RDBMS, dari sisi keamanan transaksi juga perlu di lakukan kajian khusus.

Daftar Pustaka

Triyono, J, Fatkhiyah, E, (2014), "Penggunaan Jejaring Sosial *Twitter* untuk Mengelola Stok Simplisia di Assosiasi Biofarmaka As-Syifa Farma Tempuran Kecamatan Tempuran Kabupaten Magelang" *Jurnal Generic*, Vol. 9 (2) pp. 356-370.

Triyono, J. (2010), "Pelayanan KRS on-line Berbasis SMS" *Jurnal Teknologi*, Vol.3 No 1 pp. 33-38

Utami,E, Rahardjo, S (2006)," RDBMS dengan PostgeSQL di GNU/Linux" Andi Yogyakarta