

Petunjuk Pengembangan Kasir

Owo Sugiana <sugiana@rab.co.id>

Juli 2003

Daftar Isi

1 Platform	5
1.1 Database	5
1.2 Language	5
1.3 Environment	5
1.4 Library	5
1.5 Sistem Operasi	6
2 Strategi	7
2.1 Database	7
2.1.1 ANSI SQL 92	7
2.1.2 Struktur Tabel	7
2.2 Modularitas	7
2.2.1 Reusable Code	7
2.2.2 Object Oriented	7
2.2.3 Event Driven	8
2.2.4 Kemandirian	8
2.3 Toleransi Kesalahan	8
2.3.1 Skala Prioritas	8
2.3.2 Background Process	8
2.3.3 Recoverable	9
2.3.4 Backup	9
3 Sistem	11
3.1 Pembelian	11
3.2 Penjualan	11
3.3 Kartu Stok	11
3.4 Stock Opname	12
3.5 Backup	12
3.6 Multioutlet	12
4 Kemasan	15

5	Source	17
5.1	Konfigurasi	17
5.1.1	conf.py	17
5.2	Form	17
5.2.1	kasir.py	17
5.2.2	pelanggan.py	17
5.2.3	pemasok.py	17
5.2.4	barang.py	17
5.2.5	pembelian.py	17
5.2.6	penjualan.py	18
5.2.7	opname.py	18
5.3	Transaksi	18
5.3.1	transaksi.py	18
5.4	Report	18
5.4.1	rpt_struk.py	18
6	Release	19
6.1	Versi 1.5	19
6.2	Versi 1.1	19
6.3	Versi 1.0	19

Bab 1

Platform

1.1 Database

Produk database yang digunakan berbasis SQL yang mendukung DB API 2.0 pada Python. Banyak database terkemuka cocok dengan API ini.

1.2 Language

Bahasa utama yang digunakan adalah Python 2.2. Pada level database, standar ANSI SQL 92 digunakan, atau setidaknya semua database yang didukung Kasir dapat memahami SQL yang digunakan program ini. Jadi fungsi spesifik produk database tertentu dihindari.

Pemilihan Python bertujuan untuk mempercepat siklus pengembangan (development cycle).

1.3 Environment

Untuk menjalankan Kasir, dibutuhkan console environment atau textmode. Pemilihan ini bertujuan agar mesin-mesin lama masih bisa dimanfaatkan.

1.4 Library

Tentu, banyak pustaka (library) yang digunakan. Hal yang perlu diperhatikan adalah pustaka untuk setiap database yang memang belum disertakan dalam paket Python standar. Seiring dibuatnya Kasir versi 1.5, bersama-sama dibangun pula Qc, yaitu pustaka Python untuk urusan antarmuka (userinterface) yang menggunakan Curses.

1.5 Sistem Operasi

Program ditargetkan berjalan baik di sistem operasi Linux. Meski begitu semua platform yang diutarakan di atas sebenarnya dapat dijalankan di beberapa sistem operasi, dengan beberapa penyesuaian.

Bab 2

Strategi

2.1 Database

2.1.1 ANSI SQL 92

Secara umum dikatakan bahwa SQL yang digunakan harus mendukung standar ANSI SQL 92. Namun pada prakteknya - dan paling penting - SQL yang ditulis dalam source harus dapat diterima oleh database yang didukung Kasir, setidaknya oleh PostgreSQL dan MySQL.

2.1.2 Struktur Tabel

Struktur dirancang untuk mudah dikembangkan dan menghindari pembengkakan jumlah record yang terlalu cepat. Penyatuan seluruh transaksi dalam sebuah tabel seperti pada versi Kasir sebelumnya menimbulkan resiko tersebut.

2.2 Modularitas

2.2.1 Reusable Code

Seluruh kode sebisa mungkin dapat digunakan kembali pada program lainnya. Sebagai contoh, baris-baris pencatatan transaksi ditulis dalam sebuah class tersendiri yang tidak terkait dengan antarmuka (userinterface). Dengan demikian program Python lain (misalkan Zope) dapat menggunakannya untuk program berbasis web.

2.2.2 Object Oriented

Berkaitan dengan reusable code di atas maka pemrograman berorientas objek (object oriented programming) kental di sini. Pembuatan fungsi-fungsi umum tetap dilakukan bilamana perlu. Sebuah permasalahan dipecah sesuai dengan

inti permasalahannya. Class dirancang sedemikian rupa agar dapat berinteraksi dengan class lainnya.

2.2.3 Event Driven

OOP juga erat kaitannya dengan event driven programming, yaitu memanfaatkan suatu “kejadian” pada saat sebuah proses berlangsung. Misalkan pada saat tahap akhir pengolahan halaman-halaman report, class lainnya dapat melakukan hal terkait, seperti: ikut membuat perubahan pada setiap halaman, atau sekedar menampilkan progress bar.

2.2.4 Kemandirian

Ada banyak modul dalam Kasir. Sebuah modul biasanya mewakili suatu proses. Misalkan modul penjualan (`penjualan.py`) yang dipanggil oleh program utama (`kasir.py`). Namun modul penjualan ini juga dapat dijalankan secara mandiri (tidak harus lewat menu pada program utama), sehingga mempercepat proses penelusuran kesalahan (debugging).

Class yang dibangun hanya menangani masalahnya saja, namun tetap dapat memberikan output tertentu agar dapat berinteraksi dengan class lainnya. Misalkan - seperti uraian sebelumnya - proses penulisan transaksi dipisahkan dengan urusan antarmuka. Namun apabila terjadi kesalahan pada saat penulisan, class tersebut dapat memberikan output berupa pesan kesalahan yang nantinya dapat ditampilkan oleh program yang memanggilnya.

2.3 Toleransi Kesalahan

Program yang baik adalah yang menjaga konsistensi sistem. Namun dari sudut pandang user, program yang baik adalah yang mampu menyelesaikan urusannya dengan cepat.

Misalkan, pada saat transaksi penjualan berlangsung seharusnya sistem memeriksa jumlah stok barang yang dijual. Pada kondisi pasar swalayan jelas hal tersebut tidak perlu, mengingat barang yang dibeli pelanggan sudah di depan mata. Oleh karena itu bisa saja terjadi catatan stok negatif, dan uraian di bawah ini berisi petunjuk umum untuk mengantisipasinya.

2.3.1 Skala Prioritas

Urutan sebuah proses transaksi berdasarkan yang terbaik buat user namun tetap dapat menjaga konsistensi dan ketahanan sistem (reliable).

2.3.2 Background Process

Sebuah proses bisa saja terjadi kemacetan yang seharusnya tidak perlu menjadi penghambat proses lainnya. Misalkan terjadi masalah saat mencetak struk belanja ke printer yang tidak akan menghentikan program utama untuk mencatat

transaksi berikutnya. Pencetakan dapat dilakukan di background dan program utama tidak perlu menunggunya hingga selesai.

2.3.3 Recoverable

Apabila printer memang benar-benar macet, file struk harus tetap tersimpan sehingga dapat dicetak di komputer lainnya. Setidaknya transaksi sudah tersimpan dalam database dan dapat diolah kembali agar struk yang dimaksud dapat tercetak.

Bila masalah terjadi pada database (misal: access denied), perlu dibuat sebuah log file berisi perintah-perintah SQL yang merupakan rangkaian transaksi.

2.3.4 Backup

Manajemen backup diperlukan, apalagi saat terjadi kekeliruan dalam pencatatan. Misalkan terjadi penghapusan record. Selain itu juga dapat mempercepat proses, ini yang penting buat user. Record transaksi lama tentu sudah tidak diperlukan di tingkat operator kasir. Maka perlu dibuat rutinitas backup secara harian, yaitu pemindahan record lama ke tabel atau database terpisah.

Bab 3

Sistem

3.1 Pembelian

Pembelian barang terkait dengan pemasok. Transaksi ini menjadi masukan sistem stok untuk **menambah** jumlah barang.

Transaksi ini dicatat pada tabel pembelian dan pembelian_brg.

3.2 Penjualan

Penjualan terkait dengan pelanggan. Transaksi ini menjadi masukan sistem stok untuk **mengurangi** jumlah barang. Diskon dapat diterapkan di sini, baik di setiap barang maupun di setiap transaksi.

Transaksi ini dicatat pada tabel penjualan dan penjualan_brg.

3.3 Kartu Stok

Sistem Pembelian dan Penjualan tidak melakukan perubahan terhadap stok barang. Ketiganya merupakan masukan bagi sistem Stok yang dilakukan secara terpisah, dengan alasan kecepatan. Kartu stok yang dimaksud adalah **catatan stok harian** dimana masukan berupa jumlah barang dari sistem Pembelian merupakan penambahan terhadap stok barang:

$$\text{stok} = \text{stok} + \text{jml}$$

sedangkan dari sistem Penjualan merupakan pengurangan:

$$\text{stok} = \text{stok} - \text{jml}$$

Kartu stok mirip stock opname, hanya saja ia tidak mencatat jumlah fisik barang.

Transaksi ini dicatat pada tabel stok dan stok_brg, sekaligus mengubah nilai stok pada tabel barang.

3.4 Stock Opname

Stock opname adalah pemeriksaan antara **stok fisik** yang tersedia (di gudang) dan stok yang tercatat (pada komputer). Biasanya dilakukan pada periode tertentu, misalnya sebulan sekali. Sistem ini bisa dikategorikan sebagai penjualan karena stok fisik cenderung lebih kecil dari stok yang tercatat, namun secara keuangan ini berarti pembiayaan, karena ada barang yang hilang namun tidak ada uangnya. Tahapan yang dilalui di sini adalah:

1. Perubahan stok terakhir dari hasil transaksi Pembelian dan Penjualan diperbaharui melalui sistem Stok.
2. Data seluruh barang disalin ke sebuah tabel sementara yang strukturnya mirip dengan tabel barang dengan sebuah kolom tambahan untuk mencatat stok fisik.
3. User cukup memasukkan stok fisiknya saja, karena selisihnya (yang menjadi biaya) akan diatur otomatis.
4. Setelah pencatatan rampung semuanya, kembali sistem Stok berperan kembali untuk memperbaharui catatan stok terakhir pada tabel barang.
5. Tabel sementara pada poin 2 dikosongkan.

Agar tidak terjadi kesalahan informasi stok, pada saat proses stock opname tidak boleh melakukan transaksi apapun yang melibatkan stok (pembelian maupun penjualan).

3.5 Backup

Untuk versi yang akan datang.

3.6 Multioutlet

Tentu Anda mengenal Indomaret, Alfa, Makro, atau Hero yang telah menebar banyak outlet di beberapa wilayah. Ini yang penulis maksud dengan sistem multioutlet dimana setiap outlet memiliki database sendiri yang nantinya digabung ke sebuah database besar di kantor pusat.

Penggabungan dilakukan di waktu tertentu saja dimana saat yang paling tepat adalah pada malam hari setelah transaksi ditutup.

Pada saat itu record transaksi terakhir dibackup ke dalam sebuah file terkompresi untuk kemudian dikirim ke kantor pusat, bisa melalui email atau hanya dengan disket. File ini juga berisi file `info` yang berisi:

- kode cabang tempat ia dibuat

- username dan password¹ untuk login ke database pusat

Tiba di kantor pusat file tersebut diextract dan disalin ke database induk untuk kemudian dibuat berbagai macam laporannya.

Penyalinan benar-benar dilakukan oleh user dari cabang berdasarkan username dan password pada file `info` guna lebih menjaga keabsahan data, meskipun bukan user bersangkutan yang menjalankan program penggabungan ini.

Sebaliknya, kantor pusat juga melakukan penyebaran daftar barang, misalnya untuk harga barang yang mengalami perubahan, atau ada barang baru.

Beberapa swalayan menerapkan harga yang berbeda-beda di setiap outlet untuk barang tertentu. Versi ini belum menerapkan sistem demikian.

Kode cabang di setiap outlet ditentukan pada tabel `setting` dengan field `name` “cabang” dan field `value` berisi kode cabangnya. Pada tabel ini juga terdapat “package” dengan `value`-nya berisi tanggal transaksi terakhir yang dibackup. File-file transaksi digabung ke sebuah file tar dan dibackup dengan kompresi zip, jadilah ia berakhiran `tgz`. Nama lengkap file ini memiliki rumusan:

`<kode-cabang><tgl-buat>.tgz`

kode-cabang harus terdaftar pada tabel `cabang` di kantor pusat. *tgl-buat* memiliki urutan tahun, bulan, dan tanggal, masing-masing dua angka. Di pusat kode cabang tidak diambil dari nama file backup, melainkan berasal dari file `info` yang ada di dalamnya. Semua itu dilakukan secara otomatis oleh `package.py`.

Pada pusat, struktur tabel transaksi-nya mirip dengan yang terdapat pada cabang (outlet), namun ditambah identitas cabang-nya. File backup tadi diextract dan disalin ke setiap tabel terkait. Proses ini juga dilakukan secara otomatis oleh `unpack.py`.

¹password di-encrypt

Bab 4

Kemasan

Dengan alasan komersial, Kasir dikemas menjadi beberapa paket.

Kategori	Personal	Multiuser	Multioutlet
Pembelian	Ya	Ya	Ya
Penjualan	Ya	Ya	Ya
Stok harian	Ya	Ya	Ya
Stock opname	Ya	Ya	Ya
Discount per item	Ya	Ya	Ya
Discount per transaksi	Ya	Ya	Ya
Database pemasok	Ya	Ya	Ya
Database pelanggan	Ya	Ya	Ya
Report	Ya	Ya	Ya
Webbase report			Ya
Report designer	Ya	Ya	Ya
Import dari DBase	Ya	Ya	Ya
Kolaborasi data			Ya
Manajemen user		Ya	Ya
Mutasi antar gudang		Ya	Ya
Kolaborasi data			Ya
Thin terminal support (LTSP)		Ya	Ya
Barcode generator			Ya
Database PostgreSQL, MySQL	Ya	Ya	Ya
ODBC		Ya	Ya
Running on Linux console	Ya	Ya	Ya
Ready on CD	Ya	Ya	Ya
Opensource	Ya	Ya	Ya

Tabel 4.1: Kemasan Kasir

Bab 5

Source

5.1 Konfigurasi

5.1.1 `conf.py`

Berisi informasi mengenai database yang diperlukan sebelum login. Modul ini menyiapkan variabel db yang belum di-connect() karena menunggu username dan password.

5.2 Form

5.2.1 `kasir.py`

Merupakan program utama yang berisi class FormKasir yang merupakan form utama.

5.2.2 `pelanggan.py`

Terdapat class FormPelanggan untuk mendata pelanggan.

5.2.3 `pemasok.py`

Terdapat class FormPemasok untuk mendata pemasok.

5.2.4 `barang.py`

Terdapat class FormBarang untuk mendata barang.

5.2.5 `pembelian.py`

Terdapat class FormPembelian untuk menambah stok.

5.2.6 penjualan.py

Terdapat class FormPenjualan untuk mengurangi stok dan menambah pendapatan.

5.2.7 opname.py

Terdapat class FormOpname untuk memperbaiki stok yang tercatat dengan stok yang tersedia pada gudang.

5.3 Transaksi

5.3.1 transaksi.py

Terdapat class Penjualan untuk mencatat dalam database transaksi penjualan, class Pembelian untuk transaksi pembelian, class Opname untuk transaksi stock opname, dan class Stok untuk memperbaharui data stok berdasarkan tiga transaksi tersebut.

5.4 Report

5.4.1 rpt_struk.py

Terdapat class StrukReport untuk menghasilkan file struk belanja.

Bab 6

Release

6.1 Versi 1.5

Perombakan platform terjadi di sini, dimana environment tidak lagi menggunakan X tapi cukup di console. Tujuannya adalah membuat Kasir lebih bisa diterima mesin-mesin lama. Development tools juga sepenuhnya menggunakan Python dengan dukungan multidatabase. Dibangun seiring dengan paket Qc (Qt console).

Kontributor lainnya adalah Prihantoosa (NCS) dan Kasbullah.

6.2 Versi 1.1

Masih melanjutkan platform yang digunakan versi 1.0. Pada versi ini perombakan user interface secara besar-besaran. Tampilan hanya terdiri dari dua warna: hitam dan putih. Dibuat sedemikian rupa seolah-olah program ini dijalankan untuk console environment.

6.3 Versi 1.0

Dibangun dengan Borland Kylix 1 Open Edition dan PostgreSQL. Komponen database menggunakan Zeoslib. Beberapa report dibuat dengan pemanggilan script Python.

Seluruh jenis transaksi (pembelian dan penjualan) dicatat dalam sebuah tabel. Untuk membedakannya disediakan sebuah field jenis transaksi.

Versi ini dibangun di RAB dengan personil utama Dedes Setyo Adi pada bulan Desember 2001.