UJI KELAYAKAN
PROJECT SOURCE STANDART (*PS STANDART*)
SEBAGAI *TEMPLATE* MY EASY ACCOUNTING (MEA)
DI MB-LiF SOFTWARE LABORATORY

SKRIPSI

Diajukan kepada Fakultas Teknik
Universitas Negeri Yogyakarta
Untuk Memenuhi Sebagian Persyaratan
Guna Memperoleh Gelar Sarjana Pendidikan Teknik

Oleh:
Nugroho Joko Pramono
NIM 07520244023

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
SEPTEMBER 2011

LEMBAR PERSETUJUAN

SKRIPSI

UJI KELAYAKAN
PROJECT SOURCE STANDART (*PS STANDART*)
SEBAGAI *TEMPLATE* MY EASY ACCOUNTING (MEA)
DI MB-LiF SOFTWARE LABORATORY

Dipersiapkan dan disusun oleh:

Nugroho Joko Pramono

NIM. 07520244023

Telah diperiksa dan disetujui oleh pembimbing untuk diuji.

Pada tanggal 15 September 2011

Mengetahui

Ketua Program Studi

Pendidikan Teknik Informatika

Umi Rochayati, M.T
NIP. 19630528 198710 2 001

Pembimbing

Totok Sukardiyono, M.T
NIP. 19670930 199303 1 005

SURAT PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama	: Nugroho Joko Pramono
NIM	: 07520244023
Program Studi	: Pendidikan Teknik Informatika
Fakultas	: Fakultas Teknik

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya saya sendiri. Sepanjang pengetahuan saya tidak terdapat pernyataan, pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali sebagai acuan atau kutipan dengan mengikuti tata cara penulisan karya ilmiah yang berlaku.

Tanda tangan dosen penguji yang terdapat dalam lembar pengesahan adalah asli, apabila terbukti tanda tangan dosen penguji palsu, maka saya bersedia untuk memperbaiki dan mengikuti yudisium satu tahun lagi.

Yogyakarta, 15 September 2011
Yang menyatakan,

Nugroho Joko Pramono
NIM. 07520244023

LEMBAR PENGESAHAN

SKRIPSI

UJI KELAYAKAN
PROJECT SOURCE STANDART (*PS STANDART*)
SEBAGAI *TEMPLATE* MY EASY ACCOUNTING (MEA)
DI MB-LiF SOFTWARE LABORATORY

Dipersiapkan dan Disusun Oleh:

NUGROHO JOKO PRAMONO

NIM. 07520244023

Telah Dipertahankan di Depan Dewan Penguji

FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA

Pada tanggal 28 September 2011

Dan Dinyatakan Telah Memenuhi Syarat Guna Memperoleh Gelar

SARJANA PENDIDIKAN TEKNIK

SUSUNAN PANITIA PENGUJI

Jabatan	Nama Lengkap dan Gelar	Tanda Tangan
1. Ketua penguji	Totok Sukardiyono, M.T.	.....
2. Sekertaris	Aris Nasuha, M.T.	.....
3. Penguji Utama	Dr.Eko Marpanaji	.....

Yogyakarta, 2011
Dekan FT UNY

Dr. Moch. Bruri Triyono
NIP. 19560216 198603 1 003

MOTTO

1. Ojo Rumongso Biso Yen Durung Duwe Biso Rumongso (Sugito)
2. Aku Lihat Aku Rasa Aku Bisa (Sarbiran)
3. Kejar Yang Pasti Tapi Jangan Lupa Kemungkinan (Suharban)
4. Banyak kegagalan dalam hidup ini dikarenakan orang-orang tidak menyadari betapa dekatnya mereka dengan keberhasilan saat mereka menyerah (Thomas Alfa Edison).

PERSEMBAHAN

Skripsi ini ku persembahkan untuk:

1. Ayahanda dan Ibunda yang selalu memberikan dukungan dalam setiap jalan yang aku tempuh dan melantunkan doa disetiap nafasnya untuk kesuksesanku.
2. Kakaku Amreta Wijayanti dan Ony Setyo Nugroho yang telah banyak memberikan banyak bantuan.
3. Robbin Alfath Sauqi Nugroho dan Salsky Dzaki Arrafi Nugroho yang telah banyak memberikan semangat.
4. Lisna Nurrohmawati yang telah memberi banyak motivasi.
5. Lukman Rian Affandi, Sigit Purnama, Dri Rahmanto yang telah banyak memberi bantuan.
6. Kelas F Informatika Universitas Negeri Yogyakarta Angkatan 2007
7. Almamaterku
8. Nusa bangsa dan agama

UJI KELAYAKAN
***PROJECT SOURCE STANDART* (*PS STANDART*)**
SEBAGAI *TEMPLATE MY EASY ACCOUNTING* (MEA)
DI MB-LiF *SOFTWARE LABORATORY*

Oleh : NUGROHO JOKO PRAMONO

07520244023

ABSTRAK

Penelitian ini bertujuan untuk mengembangkan Perangkat Lunak *Project Source Standart* (PS Standart) sebagai template *My Easy Accounting* (MEA) di MB-LiF *Software Laboratory* dan menguji kelayakannya dari segi *Correctness, Usability, Reliability* dan *Integrity.*

Metode yang digunakan dalam penelitian ini terdiri dari beberapa tahap, yaitu (1) Analisis kebutuhan, (2) Desain, (3) Pengkodean, (4) Validasi perangkat lunak, (5) Revisi perangkat lunak, (6) Pengujian perangkat lunak. Pengujian dengan menggunakan *Black Box, Alpha Testing* dan *Beta Testing* yang diujikan kepada 10 pengguna. Validasi perangkat lunak dalam penelitian ini dilakukan oleh ahli rekayasa perangkat lunak.

Hasil penelitian menunjukan bahwa : 1) Perangkat Lunak *Project Source Standart* (PS *Standart*) sebagai *template My Easy Accounting* (MEA) di MB-LiF *Software Laboratory* telah berhasil dirancang, dibuat, dan diimplementasikan. 2) Unjuk kerja Perangkat Lunak *Project Source Standart* (PS *Standart*) sebagai *template My Easy Accounting* (MEA) di MB-LiF *Software Laboratory* memiliki unjuk kerja yang baik, semua sistem yang diujikan dapat berjalan dan bekerja sesuai dengan spesifikasi pada analisa kebutuhan. 3) Kelayakan perangkat lunak *Project Source Standart* (PS *Standart*) sebagai *template My Easy Accounting* (MEA) di MB-LiF Software Laboratory mempunyai tingkat kelayakan 100 % dari segi *correctness*, 100 % dari segi *usability*, 100 % dari segi *reliability,* dan 100 % dari segi *integrity.*

Kata kunci: *MB-LiF Software laboratory, PS Standart, My Easy Accounting*

FEASIBILITY TEST

PROJECT SOURCE STANDART (PS STANDART)

AS TEMPLATE MY EASY ACCOUNTING (MEA)

IN MB-LiF SOFTWARE LABORATORY

By : NUGROHO JOKO PRAMONO

07520244023

ABSTRACT

This research aims to develop a Software Project Source Standart (PS Standart) as template My Easy Accounting (MEA) in MB-LiF Software Laboratory and tests its feasibility in aspects of correctness, Usability, Reliability and Integrity.

The method used in this study consists of several stages, namely (1) Analysis of needs, (2) Design, (3) Encoding, (4) Validation of software, (5) Revision of the software, (6) Testing software. Using the Black Box Testing, Alpha Testing and Beta Testing tested to 10 users. Validation of software in this study carried out by expert software engineering.

The results showed that: (1) Software Project Source Standart (PS Standart) as template My Easy Accounting (MEA) in MB-LiF Software Laboratory has successfully designed, manufactured, and implemented. (2) The performance of the Software Evaluation Test on Crossword Model (SEMOTS) for Learning has a good performance, all systems which are tested can run and work in accordance with specifications on requirements analysis. (3) Feasibility of the Software Project Source Standart (PS Standart) as template My Easy Accounting (MEA) in MB-LiF Software Laboratory has a 100% level of feasibility in aspect of correctness, 100% in aspect of usability, 100% in aspect of reliability, and 100% in aspect of integrity.

Keywords: *MB-LiF Software laboratory, PS Standart, My Easy Accounting*

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Allah S.W.T yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Uji Kelayakan Project Source Standart (PS Standart) sebagai Template My Easy Accounting (MEA) di MB-LiF Software Laboratory".

Keberhasilan penulis dalam menyusun skripsi ini atas bantuan dan dorongan dari berbagai pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Prof. Dr. H Rochmat Wahab, M.Pd, MA selaku Rektor UNY.
2. Dr. Moch. Bruri Triyono, selaku Dekan Fakultas Teknik Universitas Negeri Yogyakarta yang telah memberikan ijin dan kesempatan kepada penulis untuk menyelesaikan skripsi ini.
3. Bapak Mazduki Zakaria, M.T, selaku Kepala Jurusan Teknik Elektronika.
4. Bapak Totok Sukardoyono, M.T, selaku Dosen Pembimbing Skripsi yang telah memberikan bimbingan dan dukungan dalam penyusunan skripsi ini.
5. Bapak Drs. Kadarisman Tejo Yuwono, selaku Dosen Pembimbing Akademik.
6. Clements Hof Bouner Joan Ricardo Malau, selaku pimpinan perusahaan MB-LiF Software Laboratory.
7. Tri Yoga Kuncara, S.Kom, selaku programer kepala di MB-LiF Software Laboratory.
8. Seluruh karyawan MB-Lif Software Laboratory.
9. Keluarga di Yogyakarta atas doa dan dukungannya.

10. Teman-teman kelas F yang selalu memberikan inspirasi, motivasi dan bantuanya dalam proses penyusunan skripsi.
11. Semua pihak yang telah membantu dalam penelitian untuk penulisan skripsi ini.

Atas segala bantuan yang telah diberikan kepada penulis dan penulis doakan semoga amal dan bantuan saudara mendapat berkah yang melimpah dari Allah S.W.T.

Akhirnya penulis berharap semoga skripsi ini bermanfaat bagi para pembaca semua.

Yogyakarta, 15 September 2011
Penulis

Nugroho Joko Pramono
NIM. 07520244023

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

MB-LiF *Software Laboratory* adalah sebuah perusahaan yang bergerak dalam bidang pengembangan dan pengadaan *software* (perangkat lunak) berbasis akuntansi untuk menjalankan kegiatan bisnis perusahaan. Selama empat tahun MB-LiF berusaha mengambangkan *software* sesuai dengan permintaan klien. Seiring berjalannya waktu, karena permintaan yang semakin meningkat maka diperlukan pembuatan sebuah *template* yang dapat dikembangkan sesuai dengan kebutuhan. *Template* tersebut nantinya harus memiliki fitur-fitur standar dari berbagai kebutuhan program yang akan dikembangkan nantinya, misalnya tampilan halaman utama program (*main program*), pemberian otomatisasi operator untuk pertama kali program dijalankan, otomatisasi pembuatan *database*, otomatisasi perubahan struktur *database*, pengaturan hak akses.

Selama ini MB-LiF belum menggunakan *template* tersebut. Proses pengembangan program dilakukan secara mengulang – ulang proses yang sama dan dikerjakan secara individu oleh masing-masing *programmer*. Selain pembuatan yang secara individu tersebut, seorang *programmer* hanya dapat mengerjakan satu macam *software* saja dari awal sampai selesai, tidak dapat saling membantu untuk memudahkan pakerjaannya dikarenakan perbedaan penggunaan komponen serta penamaan fungsi dan prosedur dalam

pengembangan. Selama ini di MB-LiF belum ada pendokumentasian program yang sudah ada sehingga diperlukan sebuah *template* bagi setiap *programmer*.

Berdasarkan permasalahan tersebut, penulis bermaksud mengembangkan dan menguji kelayakan *software* yang dikembangkan sebagai *template* yang dapat digunakan untuk pengembangan *software* - *software* selanjutnya sesuai kebutuhan. Program tersebut dinamakan *PS Standart* (*Project Source Standart*).

Program *PS Standart* tersebut diharapkan dapat menjadi pondasi dari setiap program yang akan dikembangkan sesuai kebutuhan. Dengan adanya *PS Standart* akan memudahkan pengembang program dalam melaksanakan tugasnya.

B. Identifikasi Masalah

Berdasarkan uraian pada latar belakang masalah diatas, beberapa permasalahan dapat diidentifikasi sebagai berikut :

1. Permintaan pengembangan *software* di MB-LiF semakin meningkat.
2. MB-LiF belum menggunakan program template dalam pengembangan *software*.
3. Pengembangan *software* dilakukan oleh masing-masing *programmer* secara terpisah.
4. Seorang *programmer* hanya dapat mengerjakan *software* yang dikerjakan olehnya dari awal dan tidak dapat saling membantu satu sama lain karena perbedaan penggunaan prosedur, fungsi dan komponen.

5. Pendokumentasian program di MB-LiF belum dilakukan secara maksimal sehingga sulit bagi *programmer* lain untuk melanjutkan *project* yang sudah ada.

C. Batasan Masalah

Luasnya pembahasan tentang pengujian *PS Standart* dan agar permasalahan yang dibahas tidak terlalu melebar maka penelitian ini dibatasi pada Pengembangan Program *PS Standart* yang akan dibangun dan diuji memiliki fitur sebagai berikut :

1. Otomatisasi pembuatan *database* untuk pertamakali atau bila *database* yang di tunjuk tidak di temukan.
2. Otomatisasi perubahan struktur *database*
3. Pengaturan *server database*
4. Pengelolaan hak hkses
5. Pengaturan hak akses
6. Login multi *user*
7. Pengelolaan pengguna
8. *Form* profil perusahaan milik klien

D. Rumusan Masalah

Berdasarkan uraian masalah di atas, maka rumusan masalah yang akan dibahas adalah :

1. Bagaimana mengembangkan *PS Standart* di MB-LiF *Software Laboratory*?
2. Bagaimana unjuk kerja PS *Standart*?

3. Bagaimana tingkat kelayakan PS *Standart*?

E. Tujuan Penelitian

Tujuan dari penelitian yang akan dilakukan yaitu :

1. Mengembangkan *PS Standart* di MB-LiF *Software Laboratory*.
2. Mengetahui unjuk kerja PS *Standart* .
3. Mengetahui tingkat kelayakan *PS Standart* bagi *programmer* MB-LiF *Software Laboratory*.

F. Manfaat Penelitian

1. Bagi Mahasiswa
   a. Dapat menambah pengetahuan tentang pengembangan *software.*
   b. Dapat menambah pengetahuan tentang kelayakan *software*.
   c. Menerapkan pengetahuan yang didapat selama menempuh perkuliahan di Universitas Negeri Yogyakarta untuk aplikasi di lapangan khususnya bidang Teknik Informatika.
2. Bagi MB-LiF *Software Laboratory*
   a. Dapat mengembangkan *PS Standart* yang digunakan sebagai *template* program di MB-LiF *Software Laboratory*.
   b. Dapat mengetahui kelayakan *PS Standart* sebagai *template* program di MB-LiF *Software Laboratory*.
   c. Meningkatkan kinerja programmer di MB-LiF *Software Laboratory*.
3. Bagi Universitas Negeri Yogyakarta (UNY)
   a. Manfaat praktis bagi Universitas Negeri Yogyakarta yaitu memberikan intisari ilmiah mengenai “Uji Kelayakan *Project Source Standart* (*PS*

Standart) Sebagai *Template My Easy Accounting* (MEA) di MB-LiF *Software Laboratory*".

BAB II

KAJIAN PUSTAKA

A. MB-LiF *SOFTWARE LABORATORY*

MB-LiF *Software Laboratory* adalah sebuah perusahaan yang bergerak dalam bidang pengembangan dan pengadaan *software* berbasis ilmu akuntansi yang berdiri pada 1 Oktober 2006 dengan produk bernama MEA (*My Easy Accounting*). *Software* merupakan faktor kunci dalam keberhasilan suatu usaha yang dapat membedakan satu perusahaan dari perusahan pesaingnya. Dengan dasar tersebut, MB-LiF *Software Laboratory* berusaha untuk selalu memberikan solusi termudah untuk sistem akuntansi dengan lebih mengkhususkan diri pada riset, desain dan pengembangan *software*.

Dalam perkembangannya MB-LiF *Software Laboratory* memberikan kemampuan IT dan infrastruktur yang dibutuhkan untuk menjalankan sistem akuntansi dari berbagai jenis bisnis agar berjalan lebih efektif, dengan hubungan antara kami dengan pelanggan yang dilihat sebagai suatu hubungan jangka panjang yang bertujuan sama. Selama 4 tahun berdiri, MB-LiF *Software Laboratory* selalu berusaha membangun reputasi dengan memberikan solusi yang efektif, stabil, fleksibel, dan terjangkau. Investasi yang dipercayakan pada MB-LiF *Software Laboratory* akan dijawab dengan solusi hemat biaya yang dibandingkan dengan kebutuhan perusahaan jangka panjang.

B. *MY EASY ACCOUTING* (MEA)

My Easy Accounting (MEA) adalah varian program yang dibagun oleh MB-LiF *Software laboratory*. Program ini merupakan program *accounting* bagi perusahaan dengan detail fitur sebagai berikut :

1. Menu
   a. Akses Keluar/ *Logout*

      Digunakan untuk keluar dari program dan masuk lagi kedalam program dengan operator yang berbeda.

   b. Keluar

      Digunakan untuk keluar sepenuhnya dari program.

2. Hak Akses
   a. Submenu kewenangan

      Digunakan untuk melakukan pengaturan wewenang *operator* terhadap fasilitas-fasilitas yang disediakan oleh program.

   b. Submenu operator

      Digunakan untuk mengiputkan data-data operator yang akan menggunakan program.

3. Data Dasar

   Data dasar berisi submenu data dasar sesuai dengan kebutuhan.

4. Transaksi

   Transaksi berisi submenu transaksi sesui kebutuhan.

5. Arsip

   Arsip berisi rekap transaksi yang sudah dilakukan.

6. Laporan

   Laporan berisi laporan setiap transaksi yang terjadi.

7. Alat Bantu

   Alat bantu berisi :

   a. Database Server

   b. Cek Struktur Database

   c. Reset Database

   d. Calculator

8. Pengaturan

   Pengaturan berisi :

   a. Penomoran Komputer

   b. Open Price

   c. Printer

   d. Wallpapper

   e. Skin

   f. Registrasi

9. Pertolongan

   Pertolongan berisi *form profil* dan submenu *help* sesuai dengan program.

Software ini dibangun di atas *platform Windows* dan menggunakan bahasa pemrograman *Borland Delphi Versi 7.*

C. PS *STANDART* SEBAGAI *TEMPLATE*

Template merupakan program yang berisikan model-model yang biasa digunakan programmer sehingga secara langsung dapat digunakan dan tidak perlu melakukan pengaturan-pengaturan yang sama berulang kali.

PS Standart adalah nama dari program yang akan dibangun oleh penyusun. PS *Standart* yang dibangun memuat fitur-fitur sebagai berikut :

1. Otomatisasi pembuatan *database*

   Otomatisasi pembuatan *database* yaitu secara otomatis program dapat membuat database apabila *database* yang dirujuk tidak ditemukan.

2. Otomatisasi perubahan struktur *database*

   Otomatisasi perubahan struktur *database* yaitu secara otomatis program dapat merubah baik struktur *database*, struktur tabel, maupun *index tabel*.

3. Pengaturan server *database*

   Pengaturan *server database* adalah fitur yang dapat digunakan untuk merubah maupun menunjuk database server yang diinginkan. *Server database* yang dapat digunakan diantaranya adalah *MS SQL Server.*

4. Pengelolaan kewenangan

   Pengelolaan kewenangan adalah fitur yang dapat dipergunakan untuk melakukan penambahan, mengubahan dan penghapusan *level* kewenangan.

5. Pengaturan hak akses

   Pengaturan hak akses digunakan untuk melakukan pengaturan wewenang operator terhadap fasilitas-fasilitas yang disediakan oleh program.

6. Login multi *user*

   Sistem yang akan dibangun memiliki banyak *user / operator* yang dapat melakukan *login* dengan hak akses yang berbeda.

7. Pengelolaan pengguna (*Operator*)

   Fasilitas pengelolaan pengguna pada program yang akan dibangun meliputi penambahan, pengubahan dan penghapusan *user atau operator*.

8. Form profil perusahaan milik klien

   Profil perusahaan digunakan untuk menginputkan profil perusahaan, nama perusahaan akan digunakan untuk *header* setiap cetak transaksi ataupun laporan.

D. PENGERTIAN SOFTWARE

Software (perangkat lunak) merupakan program komputer yang berfungsi sebagai sarana interaksi antara pengguna dan *hardware* (perangkat keras). Perangkat lunak dapat juga dikatakan sebagai "penerjemah" perintah-perintah yang dijalankan pengguna komputer untuk diteruskan ke atau diproses oleh perangkat keras. *Software* sifatnya pun berbeda dengan *hardware*, jika *hardware* adalah komponen yang nyata yang dapat dilihat dan disentuh oleh manusia, maka *software* tidak dapat disentuh dan dilihat secara fisik, *software* memang tidak tampak secara fisik dan tidak berwujud benda tapi kita bisa mengoperasikannya.

Menurut Jogiyanto (2005:358) mengatakan bahwa perangkat lunak (*software*) adalah: Teknologi yang canggih dari perangkat keras akan berfungsi apabila instruksi-instruksi tertentu telah di berikan kepada

perangkat keras tersebut. Instruksi-instruksi tersebut disebut dengan perangkat lunak (*software*).

Menurut Roger Pressman (2002:10) *Software* merupakan perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja yang diinginkan. Perangkat lunak ini merupakan catatan bagi mesin untuk menyimpan perintah, maupun dokumen serta arsip lainnya. Keberadaan *software* dalam sistem komputer merupakan hal yang penting. *Software* ini merupakan perintah-perintah yang disusun oleh programmer untuk dapat memberdayakan sumber daya yang dimiliki oleh komputer tersebut.

Dalam kegunaannya *software* berguna untuk menerjemahkan bahasa yang digunakan manusia untuk dimengerti oleh bahasa mesin komputer yaitu *true* atau *false*. Dapat disimpulkan bahwa *hardware* tidak dapat digunakan jika *software* tidak ada di dalam sistem komputer. *Software* adalah sebuah perangkat yang terdiri dari item-item / objek-objek yang merupakan konfigurasi dari :

1. **Program** yaitu perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan.
2. **Dokumen** yang menggambarkan operasi dan kegunaan program.
3. **Data** yaitu struktur data yang memungkinkan program memanipulasi informasi secara proporsional.

E. PENGERTIAN BASIS DATA

Menurut George Tsuder Chou “Basis data merupakan kumpulan informasi yang bermanfaat yang diorganisasikan ke dalam tata cara khusus”.

Selanjutnya terdapat pendapat lain mengenai pengertian basis data, yaitu Basis data adalah sistem berkas terpadu yang dirancang tertutama untuk meminimalkan pengulangan data. Terdapat pula sebuah pendapat mengenai basis data yang mengartikan bahwa Basis data merupakan sistem terkomputerisasi yang tujuan utamanya adalah memelihara informasi dan membuat informasi itu ada pada saat dibutuhkan.

Komponen utama sistem basis data yaitu : perangkat keras, data yang bersifat terpadu (saling terkait) dan berbagi (multi-user), perangkat lunak, dan pengguna.

DBMS (database management system) merupakan program komputer yang digunakan untuk memasukkan, mengubah, menghapus, memanipulasi, dan memperoleh data / informasi dengan praktis dan efisien. DBMS juga diartikan sebagai suatu perangkat lunak yang di desain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. Selain kedua definisi tersebut DBMS juga didefinisikan sebagai perangkat lunak yang berfungsi untuk mengelola database, mulai dari membuat database itu sendiri, sampai dengan proses-proses yang berlaku dalam database tersebut, baik berupa entity, edit, hapus, query terhadap data, membuat laporan dan lain sebagainya secara efektif dan efisien.

Dibandingkan dengan sistem yang berbasis kertas, DBMS memiliki 4 keunggulan, yaitu :

1. Kepraktisan
2. Kecepatan

3. Mengurangi kejemuan
4. Kekinian

Beberapa contoh DBMS yaitu :

1. Microsoft Access
2. MySQL
3. Oracle
4. PostgreSQL
5. MS-SQL Server

F. KELAYAKAN SOFTWARE

Kelayakan suatu software dapat diuji melalui indikator-indikator yang berlaku. Menurut Mc. Call (1977) yang lebih dikenal dengan sebutan Mc.Call's quality model, menyebutkan ada beberapa model yang dapat dijadikan indikator yaitu; *correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability,* dan *interoperability*.

Correctness terdiri dari *tracebility, compleness,* dan *consistency*. Kemudian pada *reliability* terdiri dari *consisency, accurancy,* dan *error tolerance*. Setelah itu, pada indikator *efficiency* terdiri dari *execution efficiency*, dan *storage efficiency*. Pada *integrity* terdiri dari *access control* dan *access audit*. Indikator selanjutnya adalah *usability*, yaitu terdiri dari *operability, training*, dan *communicativeness*. Setelah itu, *maintainability* terdiri dari 4 bagian yaitu *simplicity, conciseness, self descriptiveness,* dan *modularity*. Selanjutnya adalah mengenai indikator *testability* yang terdiri

dari *simplicity, instrumentation, self descriptiveness,* dan *modularity*. Selanjutnya, pada *flexibility* terdiri dari *self descriptiveness, expandability, generality*. *Portability* terdiri dari berbagai bagian, yaitu *self descriptiveness, software system independence,* dan *machine independence*. Selanjutnya adalah *reusability* yang terdiri dari *self descriptiveness, generality, modularity, software system independence,* dan *machine independence.* Kemudian indikator yang terakhir adalah *interoperability* yang terdiri dari *modularity, communication commonality*, dan *data commonality*.

Gambar 1.Faktor kualitas perangkat lunak McCall

Indikator yang digunakan dalam penelitian ini yaitu *usability, reliability, integrity* dan *correctness*, karena indikator tersebut cocok menguji kelayakan program *PS Standart* yang akan diuji.

G. UNIFIED MODELLING LANGUAGE (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan

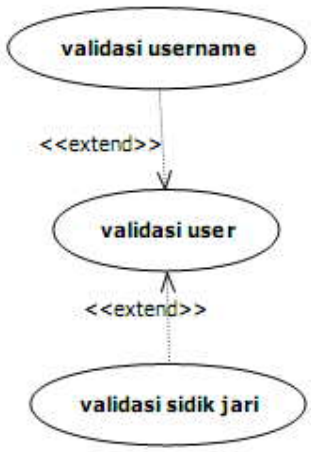
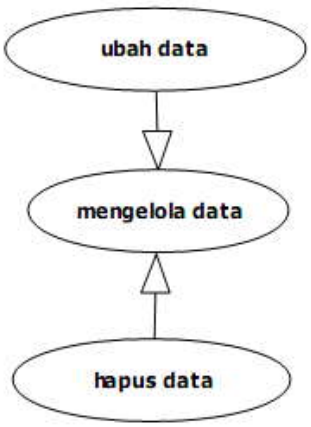
mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

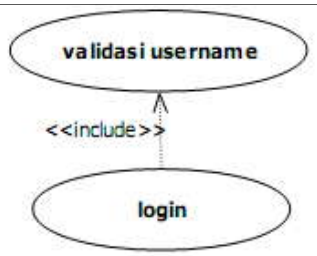
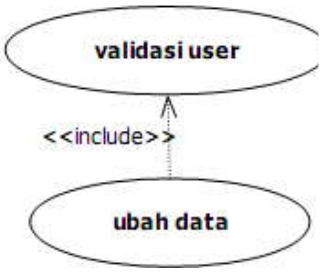
1. ***Use Case Diagram***

   Use case atau diagram *use case* merupakan pemodelan untuk kelakukan *(behavior)* suatu sistem. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah system dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 1. Simbol - simbol pada Use case

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan system sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor / *actor* 	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
Asosiasi / *association*	komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor
Ekstensi / *extend*	relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang

Simbol	Deskripsi
<<extend>>	ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  arah panah mengarah pada *use case* yang ditambahkan
Generalisasi / *generalization*	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: 

Simbol	Deskripsi
	arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
Menggunakan / *include* / *uses* <<include>> «uses»	 include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. arah panah include mengarah pada *use case* yang dipakai

Arah panah relasi pada *use case* mengarah pada *use case* yang lebih besar kontrolnya atau yang dipakai.

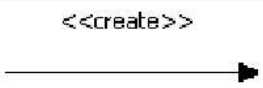
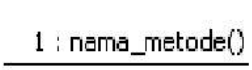
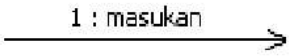
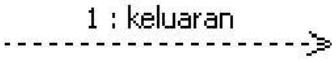
2. ***Sequence Diagram***

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang

dikirimkan dan diterima antar objek. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Tabel 2. Simbol - simbol pada *Sequence Diagram*

Simbol	Deskripsi
Aktor  atau  tanpa waktu aktif	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar system informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
Garis hidup / *lifeline* 	menyatakan kehidupan suatu objek
Waktu aktif 	menyatakan objek yang berinteraksi
Pesan tipe create	menyatakan suatu objek membuat objek yang lain, arah

Simbol	Deskripsi
	panah mengarah pada objek yang dibuat
Pesan tipe call 	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi / metode, karena ini memanggil operasi / metode maka operasi / metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

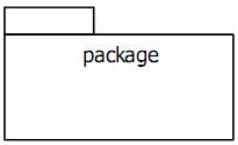
Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

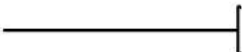
3. ***Class Diagram***

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas - kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Tabel 3. Simbol - simbol pada *Class Diagram*

Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih kelas
Kelas 	kelas pada struktur sistem
antarmuka / interface 	sama dengan konsep interface dalam pemrograman berorientasi objek

Simbol	Deskripsi
asosiasi / association	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
asosiasi berarah / directed association	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
kebergantungan / dependency	relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / aggregation 	relasi antar kelas dengan makna semua-bagian (whole-part)

Kelas – kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

a. Kelas *main*

Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem

Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

c. Kelas yang diambil dari pendefinisian *use case*

Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case.*

d. Kelas yang diambil dari pendefinisian data

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca file teks, dan lain sebagainya sesuai kebutuhan.

Metode yang didefinisikan di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah.

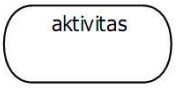
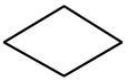
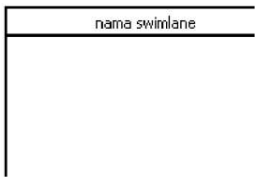
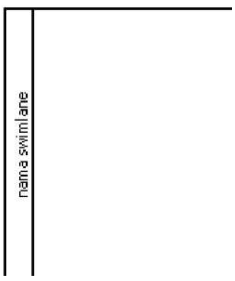
4. ***Activity Diagram***

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal – hal berikut:

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem / user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Tabel 4. Simbol - simbol pada *Activity Diagram*

Simbol	Deskripsi
status awal 	status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
percabangan / *decision* 	asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
penggabungan / *join* 	asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
status akhir 	status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane  atau 	memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

BAB III
METODE PENELITIAN

A. DESAIN PENELITIAN

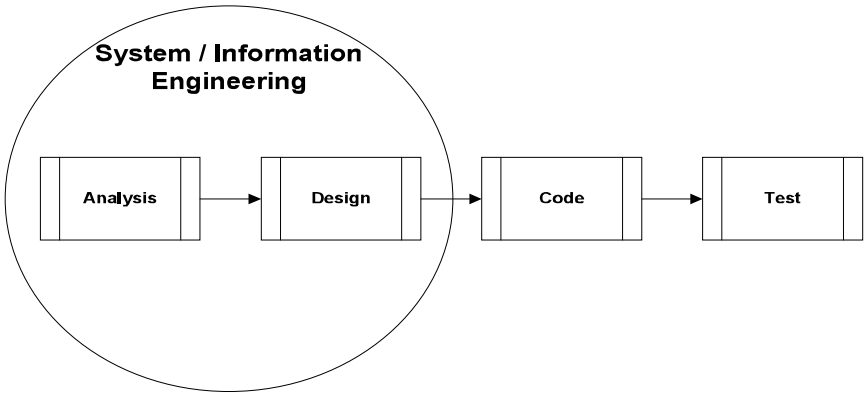
1. Metode Penelitian

Metode yang digunakan dalam penelitian ini menggunakan pendekatan penelitian pengembangan *(Research and Development)*. metode penelitian dan pengembangan adalah metode penelitian yang digunakan untuk menghasilkan produk tertentu, dan menguji keefektifan produk tersebut (Sugiyono, 2010:407).

Penelitian dan Pengembangan atau *Research and Development* (R&D) adalah suatu proses atau langkah-langkah untuk mengembangkan suatu produk baru, atau menyempurnakan produk yang telah ada, yang dapat dipertanggungjawabkan. Produk tersebut tidak selalu berbentuk benda atau perangkat keras *(hardware),* seperti buku, modul, alat bantu pembelajaran di kelas atau di laboratorium, tetapi bisa juga perangkat lunak *(software),* seperti program komputer untuk pengolahan data, pembelajaran di kelas, perpustakaan atau laboratorium, ataupun model-model pendidikan, pembelajaran, pelatihan, bimbingan, evaluasi, manajemen, dan lain – lain (Sujadi, 2003:164). Penelitian deskriptif ini hanya berusaha menggambarkan secara jelas dan sekuensial terhadap pertanyaan penelitian yang telah ditentukan sebelum para peneliti terjun ke lapangan dan mereka tidak menggunakan hipotesis sebagai petunjuk arah atau guide dalam penelitian (Sukardi, 2004:14). Tetapi perlu diketahui

bahwa tidak setiap penelitian harus merumuskan hipotesis. Penelitian yang bersifat eksploratif dan deskriptif sering tidak perlu merumuskan hipotesis(Sugiyono, 2010:96).

Salah satu metode dalam membangun perangkat lunak adalah metode waterfall. Metode waterfall adalah model klasik yang bersifat sistematis, berurutan dalam membangun suatu software. Langkah – langkah dalam *Waterfall Method* adalah (Pressman, 2001 : 29) :



Gambar 2. *Waterfall* menurut Pressman

a. Analisis kebutuhan *(Software requirements Analysis)*

Peneliti menganalisis kebutuhan dengan observasi terhadap aplikasi–aplikasi yang dikembangkan di MB-LiF *Software Laboratory* yang sudah ada. Aplikasi yang diobservasi adalah varian dari *My Easy Accounting* di MB-LiF *Software Laboratory*. Spesifikasi perangkat lunak dibahas pada tahap ini.

b. Desain *(design)*

Peneliti merancang desain dari analisis kebutuhan yang sudah dilakukan pada tahap pertama. Pemodelan desain sistem dari perangkat

lunak akan menggunakan metode UML. Spesifikasi perangkat lunak secara detail dibuat pada tahap ini.

c. Pengkodean *(Code Generation)*

Implementasi pengkodean dilakukan berdasarkan desain dan spesifikasi yang sudah diverifikasi. Peneliti akan langsung menguji dari tiap unit program pada saat implementasi pengkodean. Pengujian *white-box* dilakukan pada tahap ini.

d. Pengujian *(Testing)*

Pengujian terintegrasi merupakan pengujian secara keseluruhan program. Pengujian ini dilakukan peneliti untuk mendapatkan kelayakan dari aplikasi yang telah dibangun. Pengujian yang dilakukan peneliti dalam pengembangan perangkat lunak *Project Source Standart (PS Standart)* ini meliputi :

1) *Alpha testing*

Pengujian *alpha testing* dalam pengembangannya dilakukan oleh ahli rekayasa perangkat lunak. Penilaian ditinjau dari analisis kebutuhan dan desain. Ahli rekayasa perangkat lunak akan me-verifikasi dan me-validasi aplikasi yang sudah dibangun. Ahli rekayasa perangkat lunak menilai unjuk kerja dari perangkat lunak *PS Standart* dan memberikan masukkan– masukkan terhadap aplikasi yang sudah dibangun. Tindak lanjut dari *alpha testing* ini adalah revisi tahap awal untuk perangkat lunak *PS Standart.*

2) Beta testing

Beta testing merupakan tahap akhir pengujian penyempurnaan perangkat lunak *PS Standart* yang dikembangkan. Pengujian ini dilakukan oleh pengguna. Pengujian ini untuk mendapatkan kelayakan ditinjau dari segi *Correctness, Usability, Reliability* dan *Integrity* perangkat lunak yang telah di bangun. Pengguna akan memberikan umpan balik dari kesalahan yang terjadi pada perangkat lunak. Umpan balik tersebut digunakan untuk revisi tahap akhir dari perangkat lunak *PS Standart.*

2. Objek Penelitian

Objek penelitian adalah perangkat lunak *Project Source Standart (PS Standart)* sebagai perangkat lunak yang layak untuk digunakan sebagai *template* My Easy Accounting.

3. Tempat dan Waktu Penelitian

Penelitian dilaksanakan di MB-LiF Software Laboratory. Adapun pelaksanaan dimulai bulan Februari 2011 sampai selesai.

4. Responden

Responden dalam penelitian ini adalah sebanyak 10 orang. Penelitian ini juga memilih 5 (lima) orang ahli rekayasa perangkat lunak yang terdiri dari 3 (tiga) orang dosen Teknik Informatika Universitas negeri Yogyakarta dan 2 (dua) orang karyawan MB-LiF Software

Laboratory sebagai pengujian ahli *(Expert Judgment)* untuk mengukur unjuk kerja dari aplikasi yang telah dibangun.

5. Alat dan Bahan Penelitian

Fasilitas atau perangkat pendukung yang digunakan dalam penelitian ini yaitu:

a. Perangkat Komputer

Satu buah perangkat notebook Intel Pentium Core 2 Duo dengan prosesor 2,00 Ghz, memori DDR2 2 GB, Hardisk 250 GB, Soundcard, VGA, DVD/CD-RW, Keyboard, Mouse.

b. Printer

Printer yang digunakan adalah EPSON TX111. Printer ini digunakan untuk mencetak data berupa tulisan/teks, gambar dan laporan.

c. Jaringan Internet

Jaringan internet digunakan untuk mengakses internet.

d. Perangkat Lunak

Proses pengembangan Perangkat Lunak *Project Source Standart (PS Standart)* menggunakan beberapa perangkat lunak. Perangkat lunak tersebut adalah *Borland Delphi 7.0, Star UML* serta program perangkat lunak pendukung lainnya.

B. Pengembangan Perangkat Lunak

1. Analisa Kebutuhan

Analisis kebutuhan merupakan tahap awal dari sebuah tahapan rekayasa perangkat lunak pada penelitian ini. Peneliti menganalisis kebutuhan fungsional, *interface* perangkat lunak, dan membangun batasan yang harus dipenuhi oleh suatu perangkat lunak. Peneliti menganalisis kebutuhan dengan dua macam analisis, yaitu mencari informasi dengan observasi terhadap aplikasi – aplikasi farian *My Easy Accounting* yang sudah ada di MB-LiF Software Laboratory.

a. Hasil Observasi

Peneliti melakukan observasi langsung terhadap perangkat lunak *My Easy Accounting* yang sudah ada. Perangkat lunak tersebut adalah *My Easy Accounting For Tienz, My Easy Accounting For Retail V2, My Easy Accounting For Retail V2 Basic, My Easy Accounting For Retail V2 Advance, My Easy Accounting For Melia dan My Easy Accounting For UPN* . Peneliti mendapatkan kesimpulan bahwa:

1) Perangkat lunak harus mempunyai fasilitas untuk membuat database secara otomatis, merubah struktur database, dapat mengelola Operator, mengelola kewenangan, mengelola hak akses, penentuan server database, dan database yang di rujuk.
2) Semua operator dapat menggunakan fasilitas sesuai kewenangan yg di berikan.

3) Perangkat lunak harus dapat di jalankan secara *multi user*.

2. Perancangan Sistem

a. Perancangan Database

Database merupakan salah satu komponen yang sangat penting di dalam merancang sebuah sistem, hal pertama yang harus dilakukan adalah merancang database terlebih dahulu yang nantinya database tersebut digunakan sebagai bahan dalam menjalankan sistem. Database untuk PS Standart meliputi :

1) Tabel operator

Primary Key : operator_id

	Column Name	Data Type	Length	Allow Nulls
🔑	operator_id	varchar	17	
	privilege_id	varchar	17	✓
	operator_status	varchar	1	✓
	operator_user	varchar	50	✓
	operator_pass	varchar	50	✓
	operator_name	varchar	50	✓
	operator_logout	varchar	50	✓
	operator_last_login	datetime	8	✓

Gambar 3. Gambar Tabel operator

2) Tabel company

Primary Key : company_name

	Column Name	Data Type	Length	Allow Nulls
🔑	company_name	varchar	100	
	company_addr	varchar	100	✓
	company_telp	varchar	100	✓
	company_npwp	varchar	100	✓
	company_contact	varchar	100	✓

Gambar 4. Gambar Tabel Company

3) Tabel My_Index

Primary Key : mi_id

	Column Name	Data Type	Length	Allow Nulls
🔑	company_name	varchar	100	
	company_addr	varchar	100	✓
	company_telp	varchar	100	✓
	company_npwp	varchar	100	✓
	company_contact	varchar	100	✓

Gambar 5. Gambar Tabel my_index

4) Tabel op_permission

Primary Key : -

	Column Name	Data Type	Length	Allow Nulls
▶	privilege_id	varchar	17	
	permission_type	varchar	20	✓
	permission_value	char	1	✓

Gambar 6. Gambar Tabel op_permission

5) Tabel privilege

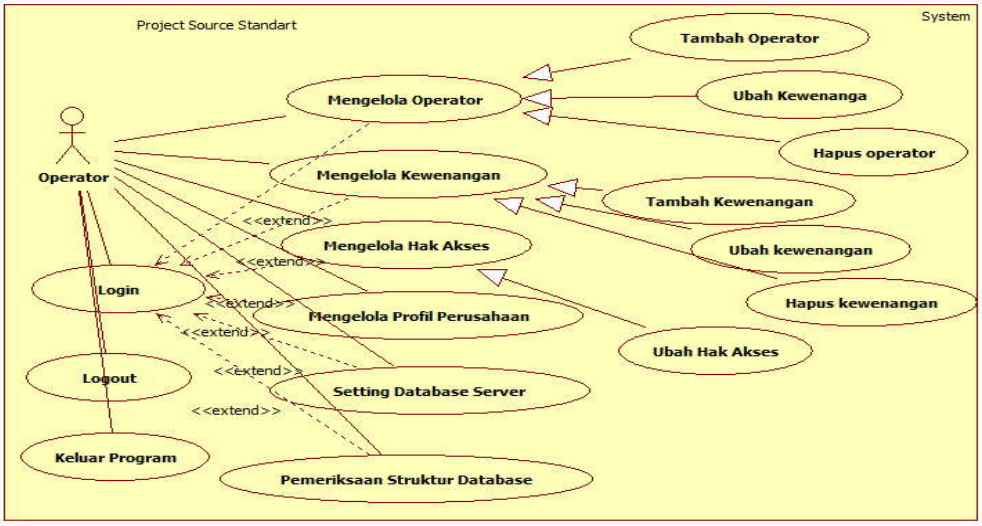
Primary Key : privilege_id

	Column Name	Data Type	Length	Allow Nulls
🔑	privilege_id	varchar	17	
	privilege_name	varchar	50	✓
	privilege_note	varchar	50	✓
	privilege_permission	varchar	1000	✓

Gambar 7. Gambar Tabel privilege

b. *Use Case Diagram*

Analisis kebutuhan yang sudah dilakukan akan dimodelkan dalam *use case* diagram. Kebutuhan fungsional yang ada dari analisis kebutuhan dimasukkan dalam *case – case* tersendiri Pemeran atau *actor* dalam hal ini adalah *Operator*.



Gambar 8. *Use Case Diagram*

1) Definisi Aktor

Berikut adalah deskripsi pendefinisian aktor pada perangkat lunak PS Standart :

Tabel 5. Definisi Aktor

No	Use case	Deskripsi
1	Operator	Orang yang bertugas dan memiliki hak akses untuk melakukan operasi pengelolaan data dan seluruh fasilitas menurut hak akses yang diberikan

2) Definisi *Use Case*

Berikut adalah deskripsi pendefinisian *use case* pada perangkat lunak PS Standart:

Tabel 6. Definisi *use case*

No	Use case	Deskripsi
1	Login	merupakan proses pengecekan validasi operator san passwordnya serta hak akses sapasaja yang dia peroleh dalam sistem
2	Logout	Merupakan proses yang dipergunaka oleh operator apabila ingin keluar dari hak aksesnya untuk dipergunakan oleh operator lain
3	Keluar	Merupakan fasilitas untuk keluar sepenuhnya dari program
4	Mengelola data operator	merupakan proses generalisasi yang meliputi tiga buah proses pengelolaan data operator yaitu menambah data operator, mengubah data operator, dan menghapus data operator
5	Menambah data operator	Merupakan proses memasukkan data operator yang baru
6	Mengubah data operator	Merupakan proses mengubah data operator yang sudah ada
7	Menghapus data operator	Merupakan proses menghapus data operator
8	Mengelola data kewenangan	merupakan proses generalisasi yang meliputi tiga buah proses pengelolaan data operator yaitu menambah data kewenangan, mengubah data kewenangan, dan menghapus data kewenangan
9	Menambah data kewenangan	Merupakan proses memasukkan data kewenangan yang baru
10	Mengubah data kewenangan	Merupakan proses mengubah data kewenangan yang sudah ada

No	Use case	Deskripsi
11	Menghapus data kewenangan	Merupakan proses menghapus data kewenangan
12	Mengubah data Akses	Merupakan proses mengubah hak akses setiap data kewenangan atas setiap menu
13	Memasukkan data profil perusahaan	Merupakan proses memasukkan data profil perusahaan dimana program nantinya akan di *install*
14	Mengubah data profil perusahaan	Merupakan proses mengubah data profil perusahaan
15	Melakukan Setting database server	Merupakan proses untuk menentukan nama database yang di rujuk
16	Melakukan *check* struktur database	Merupakan proses untuk melakukan pengecekan struktur database yang tidak konsisten maupun melakukan perubahan secara otomatis kepada struktur database apa bila dilakukan perubahan

3) Definisi Skenario

Berikut adalah skenario jalannya masing - masing *use case* yang telah didefinisikan sebelumnya :

a) Nama *use case* : *Login*

Skenario :

Tabel 7. Skenario *use case login*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memasukkan user dan password	
	2. Mengecek valid tidaknya data masukkan
	3. Masuk ke aplikasi *PS Standart*
	4. Kirim informasi aktifitas pengguna ke basis data
Skenario Alternatif	
1. Memasukkan user dan password yg tidak valid	
	2. Mengecek valid tidaknya data masukkan
	3. Menampilkan pesan login tidak valid
4. Memasukkan user dan password yang valid	
	5. Mengecek valid tidaknya data masukkan
	6. Masuk ke aplikasi *PS Standart*
	7. Kirim informasi aktifitas pengguna ke basis data

b) Nama *use case* : *Logout*

Skenario :

Tabel 8. Skenario *use case logout*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu logout dan memilih tombol "yes" pada jendela pertanyaan	
	2. Memberikan pertanyaan apakah akan benar – benar melakukan logout
	3. Program menutup fasilitas yang ada
	4. Jendela Login Muncul
Skenario Alternatif	
1. Memilih menu logout dan memilih tombol "no" pada jendela pertanyaan	
	2. Jendela pertanyaan menghilang
	3. Kembali ke menu utama

c) Nama *use case* : *Keluar*

Skenario :

Tabel 9. Skenario *use case keluar*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu keluar dan menekan tombol "yes" pada jendela pertanyaan	
	2. Memberikan informasi bahwa belum melakukan logput dan pertanyaan apakah akan benar – benar melakukan keluar
	3. Keluar program

Aksi Aktor	Reaksi Sistem
	4. Jendela Login Muncul
Skenario Alternatif	
1. Memilih menu keluar dan menekan tombol “no” pada jendela pertanyaan	
	2. Jendela pertanyaan menghilang
	3. Kembali ke menu utama

d) Nama *use case* : Memasukkan data pengguna

Skenario :

Tabel 10. Skenario *use case* Memasukkan data operator

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memasukkan data operator sesuai kolom yang ada	
	2. Mengecek valid tidaknya data masukkan
	3. Menyimpan data operator ke basis data
	4. Menampilkan pesan sukses disimpan
Skenario Alternatif	
1. Memasukkan data operator sesuai kolom yang ada	
	2. Mengecek valid tidaknya data masukkan
	3. Mengeluarkan pesan bahwa data masukkan tidak valid
1. Memperbaiki data masukkan yang tidak valid	
	2. Mengecek valid tidaknya data masukkan
	3. Menyimpan data operator ke basis data
	4. Menampilkan pesan sukses disimpan

e) *Nama use case* : Mengubah data operator

Skenario :

Tabel 11. Skenario *use case* Mengubah data operator

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih data operator yang akan diubah	
	2. Menampilkan semua data yang dipilih kedalam form yang ada
3. Mengubah data operator	
	4. Mengecek valid tidaknya data masukkan
	5. Menyimpan data Operator yang telah diubah ke basis data
	6. Menampilkan pesan bahwa data sukses disimpan
Skenario Alternatif	
1. Memilih data Operator yang akan diubah	
	2. Menampilkan semua data yang dipilih kedalam form yang ada
3. Mengubah data Operator	
	4. Mengecek valid tidaknya data masukkan
	5. Menampilkan pesan bahwa data masukkan tidak valid
6. Memilih data Operator yang akan diubah	
	7. Menampilkan semua data yang dipilih kedalam form yang ada
8. Mengubah data Operator	
	9. Mengecek valid tidaknya data masukkan
	10. Menyimpan data Operator yang telah diubah ke basis data
	11. Menampilkan pesan bahwa data sukses disimpan

f) Nama *use case* : Menghapus data Operator

Skenario :

Tabel 12. Skenario *use case* Menghapus data Operator

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih data Operator yang akan dihapus	
	2. Menampilkan pesan konfirmasi apakah data akan benar-benar dihapus
3. Mengeklik pilihan setuju data dihapus	
	4. Menghapus data Operator dari basis data
	5. Menampilkan pesan bahwa data sukses dihapus
Skenario Alternatif	
1. Memilih data Operator yang akan dihapus	
	2. Menampilkan pesan konfirmasi apakah data akan benar-benar dihapus
3. Mengeklik pilihan tidak setuju data dihapus	
	4. Kembali ke menu pengelolaan Operator

g) *Nama* use *case* : Memasukkan data kewenangan

Skenario :

Tabel 13. Skenario *use case* Memasukkan data kewenangan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
5. Memasukkan data kewenangan sesuai kolom yang ada	
	6. Mengecek valid tidaknya data masukkan
	7. Menyimpan data Kewenangan

Aksi Aktor	Reaksi Sistem
	ke basis data
	8. Menampilkan pesan sukses disimpan
Skenario Alternatif	
4. Memasukkan data Kewenangan sesuai kolom yang ada	
	5. Mengecek valid tidaknya data masukkan
	6. Mengeluarkan pesan bahwa data masukkan tidak valid
5. Memperbaiki data masukkan yang tidak valid	
	6. Mengecek valid tidaknya data masukkan
	7. Menyimpan data Kewenangan ke basis data
	8. Menampilkan pesan sukses disimpan

h) Nama *use case* : Mengubah data Kewenangan

Skenario :

Tabel 14. Skenario *use case* Mengubah data Kewenangan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
7. Memilih data Kewenangan yang akan diubah	
	8. Menampilkan semua data yang dipilih kedalam form yang ada
9. Mengubah data Kewenangan	
	10. Mengecek valid tidaknya data masukkan
	11. Menyimpan data Kewenangan yang telah diubah ke basis data
	12. Menampilkan pesan bahwa data sukses disimpan
Skenario Alternatif	
12. Memilih data Kewenangan	

Aksi Aktor	Reaksi Sistem
yang akan diubah	
	13. Menampilkan semua data yang dipilih kedalam form yang ada
14. Mengubah data Kewenangan	
	15. Mengecek valid tidaknya data masukkan
	16. Menampilkan pesan bahwa data masukkan tidak valid
17. Memilih data Kewenangan yang akan diubah	
	18. Menampilkan semua data yang dipilih kedalam form yang ada
19. Mengubah data Kewenangan	
	20. Mengecek valid tidaknya data masukkan
	21. Menyimpan data Kewenangan yang telah diubah ke basis data
	22. Menampilkan pesan bahwa data sukses disimpan

i) Nama *use case* : Menghapus data Kewenangan

Skenario :

Tabel 15. Skenario *use case* Menghapus data Kewenangan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
6. Memilih data Kewenangan yang akan dihapus	
	7. Menampilkan pesan konfirmasi apakah data akan benar-benar dihapus
8. Mengeklik pilihan setuju data dihapus	
	9. Menghapus data Kewenangan dari basis data
	10. Menampilkan pesan bahwa data sukses dihapus
Skenario Alternatif	

Aksi Aktor	Reaksi Sistem
5. Memilih data Kewenangan yang akan dihapus	
	6. Menampilkan pesan konfirmasi apakah data akan benar-benar dihapus
7. Mengeklik pilihan tidak setuju data dihapus	
	8. Kembali ke menu pengelolaan Kewenangan

j) Nama *use case* : Mengubah data Akses

Skenario :

Tabel 16. Skenario *use case* Mengubah hak akses

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Mengubah hak Akses	
	2. Mengecek hak Akses dari database
3. Klik tombol simpan	
	4. Menyimpan hak Akses ke basis data
	5. Menampilkan pesan sukses disimpan

k) Nama use *case* : Memasukkan data profil perusahaan

Skenario :

Tabel 17. Skenario *use case* Memasukkan data profil perusahaan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
9. Memasukkan data profil perusahaan sesuai kolom yang ada	
	10. Mengecek valid tidaknya data masukkan
	11. Menyimpan data profil perusahaan ke basis data

Aksi Aktor	Reaksi Sistem
	12. Menampilkan pesan sukses disimpan
Skenario Alternatif	
7. Memasukkan data profil perusahaan sesuai kolom yang ada	
	8. Mengecek valid tidaknya data masukkan
	9. Mengeluarkan pesan bahwa data masukkan tidak valid
9. Memperbaiki data masukkan yang tidak valid	
	10. Mengecek valid tidaknya data masukkan
	11. Menyimpan data profil perusahaan ke basis data
	12. Menampilkan pesan sukses disimpan

l) Nama *use case* : Mengubah data profil perusahaan

Skenario :

Tabel 18. Skenario *use case* Mengubah data profil perusahaan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
13. Memilih data profil perusahaan yang akan diubah	
	14. Menampilkan semua data yang dipilih kedalam form yang ada
15. Mengubah data profil perusahaan	
	16. Mengecek valid tidaknya data masukkan
	17. Menyimpan data profil perusahaan yang telah diubah ke basis data
	18. Menampilkan pesan bahwa data sukses disimpan

Aksi Aktor	Reaksi Sistem
Skenario Alternatif	
23. Memilih data profil perusahaan yang akan diubah	
	24. Menampilkan semua data yang dipilih kedalam form yang ada
25. Mengubah data profil perusahaan	
	26. Mengecek valid tidaknya data masukkan
	27. Menampilkan pesan bahwa data masukkan tidak valid
28. Memilih data profil perusahaan yang akan diubah	
	29. Menampilkan semua data yang dipilih kedalam form yang ada
30. Mengubah data profil perusahaan	
	31. Mengecek valid tidaknya data masukkan
	32. Menyimpan data profil perusahaan yang telah diubah ke basis data
	33. Menampilkan pesan bahwa data sukses disimpan

m) Nama *use case* : Melakukan Setting database server

Skenario :

Tabel 19. Skenario *use case* Melakukan setting database server

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih server, memasukkan username dan password server	
	2. Menampilkan semua database yang ada diserver tersebut
3. Memilih database	
	4. Mengecek valid tidaknya data masukkan
5. Klik menu simpan	

Aksi Aktor	Reaksi Sistem
	6. Menyimpan setting database server
	7. Menampilkan pesan bahwa data sukses disimpan
	8. Keluar dari aplikasi
Skenario Alternatif	
1. Memilih server, memasukkan username dan password server	
	2. Menampilkan semua database yang ada diserver tersebut
3. Memilih database	
	4. Mengecek valid tidaknya data masukkan
	5. Menampilkan pesan bahwa data masukkan tidak valid
	6. Menampilkan pesan apakan bahwa apakah akan membuat database yg di rujuk
7. Memilih membuat database baru	
	8. Melakukan proses membuat database baru
9. Klik menu simpan	
	10. Menyimpan setting database server
	11. Menampilkan pesan bahwa data sukses disimpan
	12. Keluar dari aplikasi
Skenario Alternatif	
13. Memilih server, memasukkan username dan password server	
	14. Menampilkan semua database yang ada diserver tersebut
15. Memilih database	
	16. Mengecek valid tidaknya data masukkan
	17. Menampilkan pesan bahwa data masukkan tidak valid
	18. Menampilkan pesan apakan bahwa apakah akan membuat database yg di rujuk
19. Memilih tidak membuat	

Aksi Aktor	Reaksi Sistem
database baru	
	20. Keluar dari aplikasi

n) Nama *use case* : Melakukan *check* struktur database

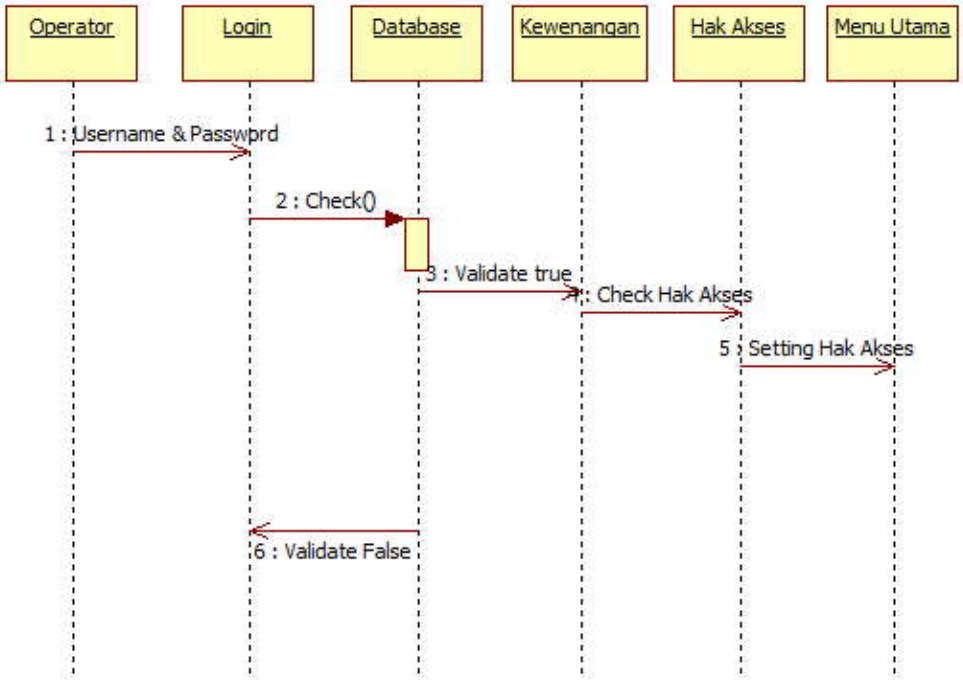
Skenario :

Tabel 20. Skenario *use case* Melakukan *check* struktur database

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu *check* struktur database	
	2. Menampilkan form ckeck struktur database
3. Menekan tombol Check	
	4. Melakukan proses check struktur database
	5. Menampilkan pesan bahwa check struktur database selesai di kerjakan
Skenario Alternatif	
1. Memilih menu *check* struktur database	
	2. Menampilkan form ckeck struktur database
3. Menekan tombol Check	
	4. Melakukan proses check struktur database
	5. Menampilkan pesan bahwa check struktur database gagal di kerjakan

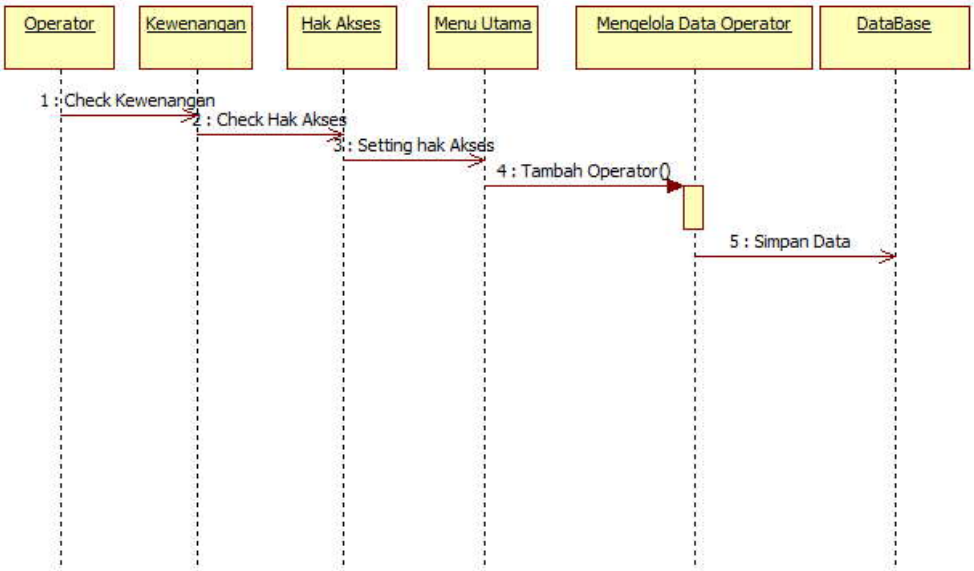
c. *Sequence* Diagram

Sequence diagram adalah tahapan selanjutnya setelah *use case diagram.* Di sini objek – objek dimodelkan dalam keterkaitannya. Tiap *use case* dimodelkan dalam satu *sequence diagram.*



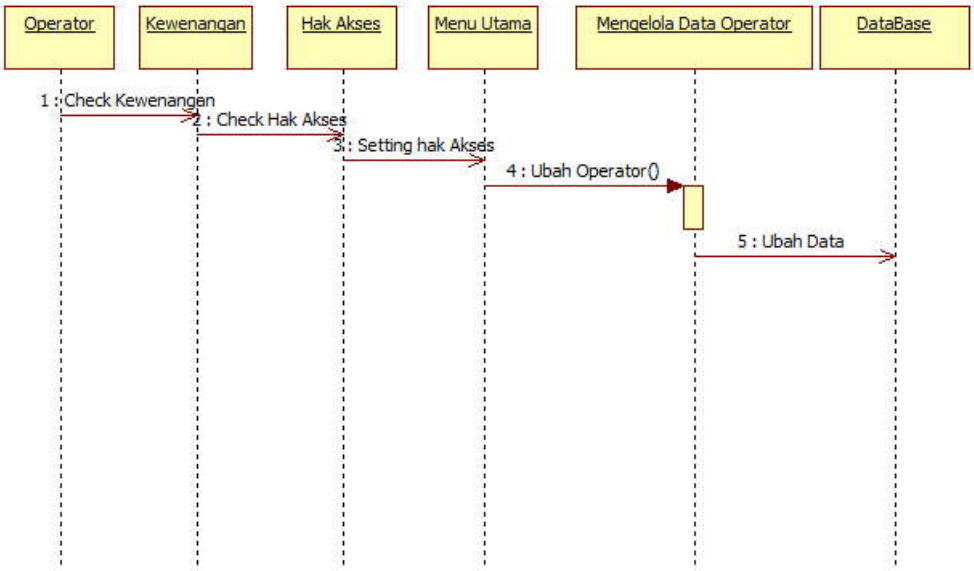
Gambar 9. *Sequence Diagram Login*

Sequence diagram di atas menjelaskan proses operator masuk ke dalam aplikasi. Operator mengakses menu *login* untuk dapat masuk ke menu utama. Setelah pengguna memasukkan *user name* dan *password* maka langkah selanjutnya sistem akan mengakses tabel Operator. Jika user dan password yang dimasukkan sesuai dengan tabel user maka Operator akan masuk ke menu utama. Sebaliknya jika user dan password yang dimasukkan tidak sesuai maka Operator tidak dapat masuk ke menu utama.

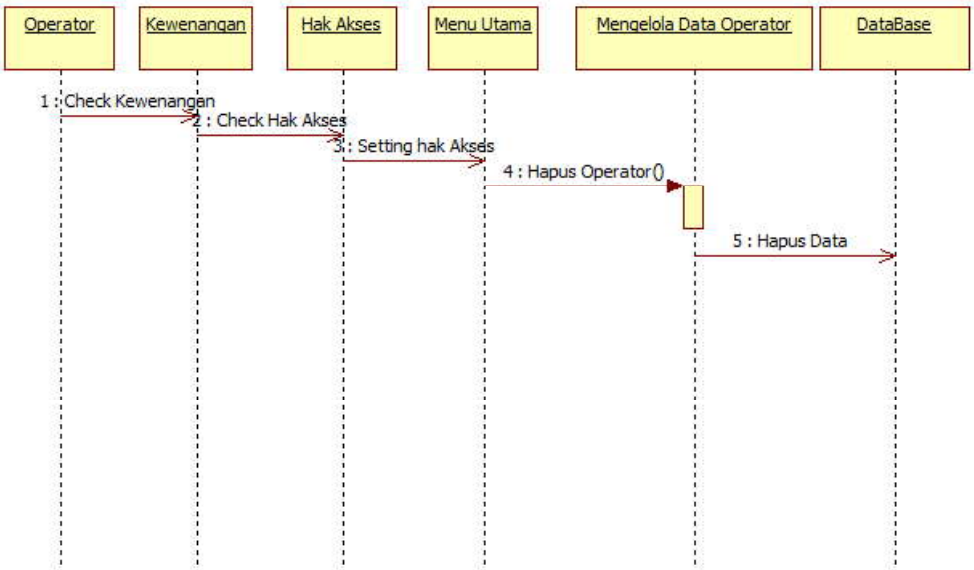


Gambar 10. *Sequence Diagram* Memasukkan data Operator

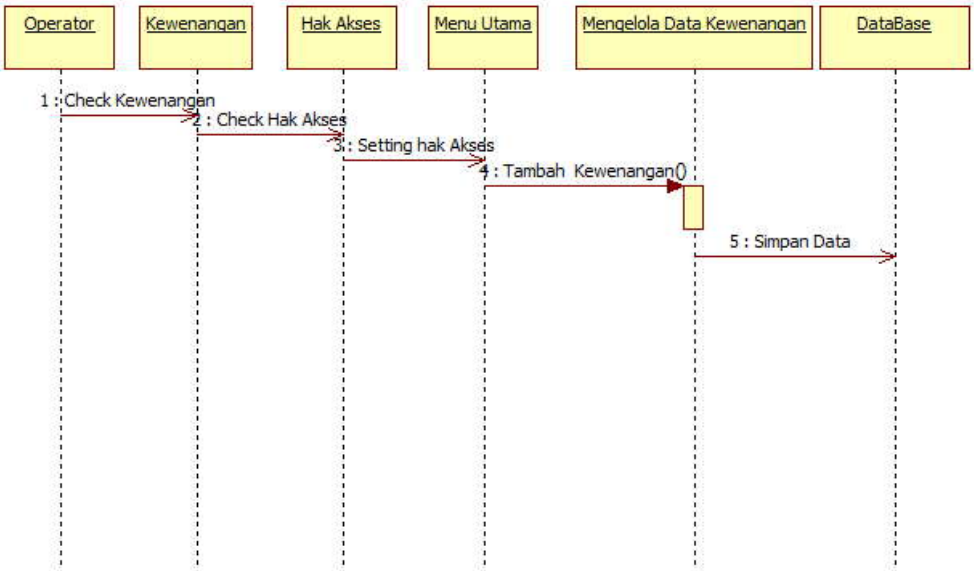
Sequence diagram di atas menjelaskan proses Operator *memasukkan* data Operator baru. Proses ini hanya bisa dilakukan oleh Operator yang memiliki level Kewenangan dengan Hak Akses tertentu. Operator memilih menu tambah pada menu mengelola data Operator. Perintah tambah Operator akan menambahkan data Opertor ke dalam tabel Operator. *Sequence diagram* pada gambar 6 dan 7 memiliki prosedur yang sama seperti proses pada gambar 5. Secara keseluruhan, perbedaan dari *sequence diagram* pada gambar 6 dan 7 terletak pada perintah mengubah dan menghapus.



Gambar 11. *Sequence Diagram* Mengubah data Operator

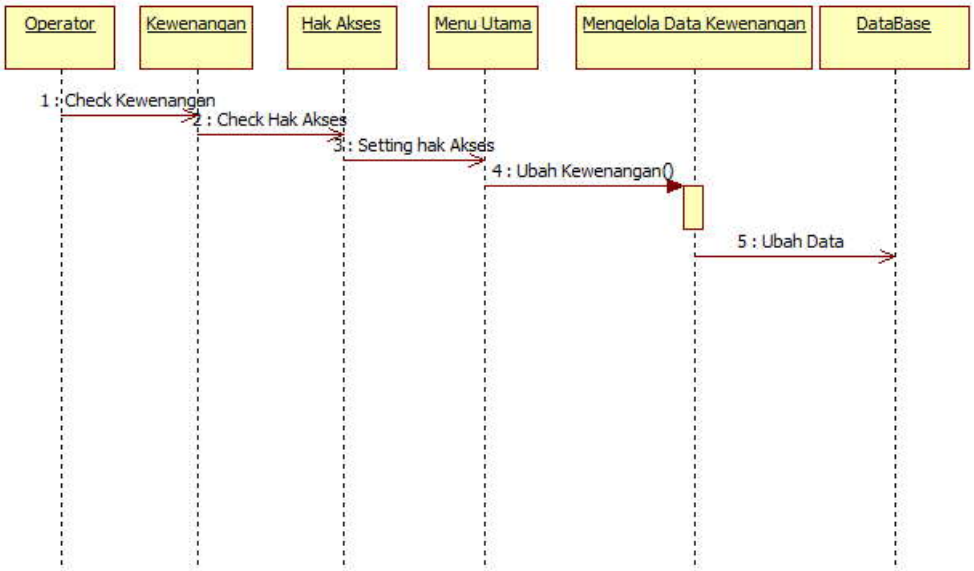


Gambar 12. *Sequence Diagram* Menghapus data Operator

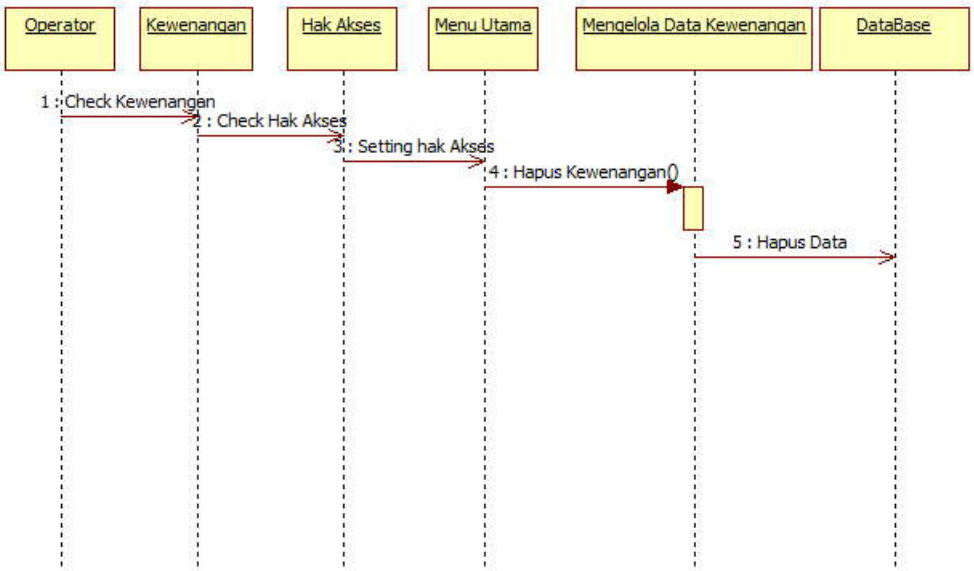


Gambar 13. *Sequence Diagram* Memasukkan data Kewenangan

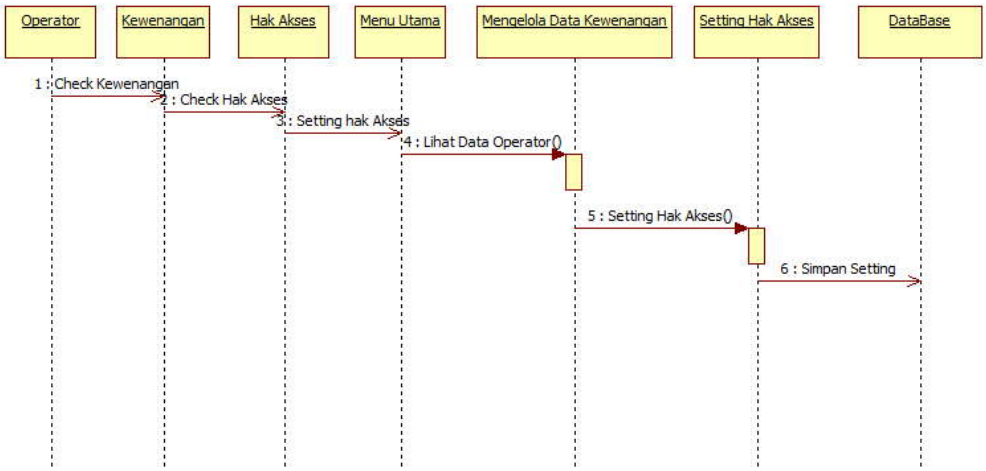
Sequence *diagram* di atas menjelaskan proses Operator memasukkan data Kewenangan baru. Proses ini hanya bisa dilakukan oleh Operator yang memiliki level Kewenangan dengan Hak Akses tertentu. Operator memilih menu tambah pada menu mengelola data Kewenangan. Perintah tambah Kewenangan akan menambahkan data kewenangan ke dalam tabel Previlege. *Sequence diagram* pada gambar 9 dan 10 memiliki prosedur yang sama seperti proses pada gambar 8. Secara keseluruhan, perbedaan dari *sequence diagram* pada gambar 9 dan 10 terletak pada perintah mengubah dan menghapus.



Gambar 14. *Sequence Diagram* Mengubah data Kewenangan

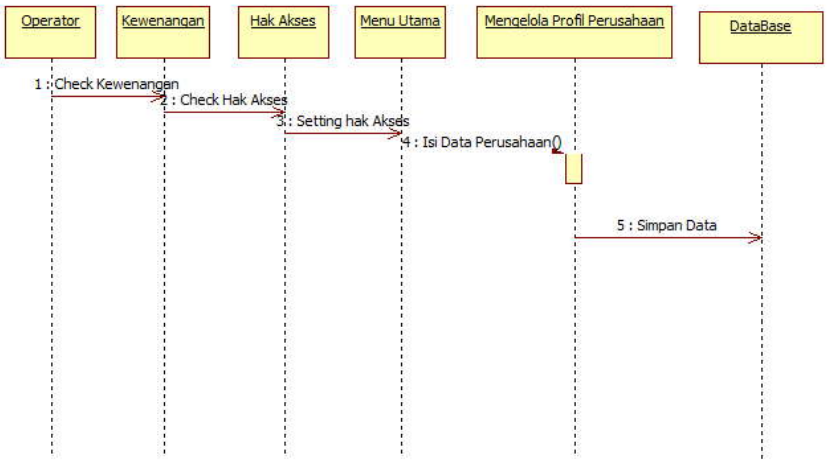


Gambar 15. *Sequence Diagram* Menghapus data Kewenangan



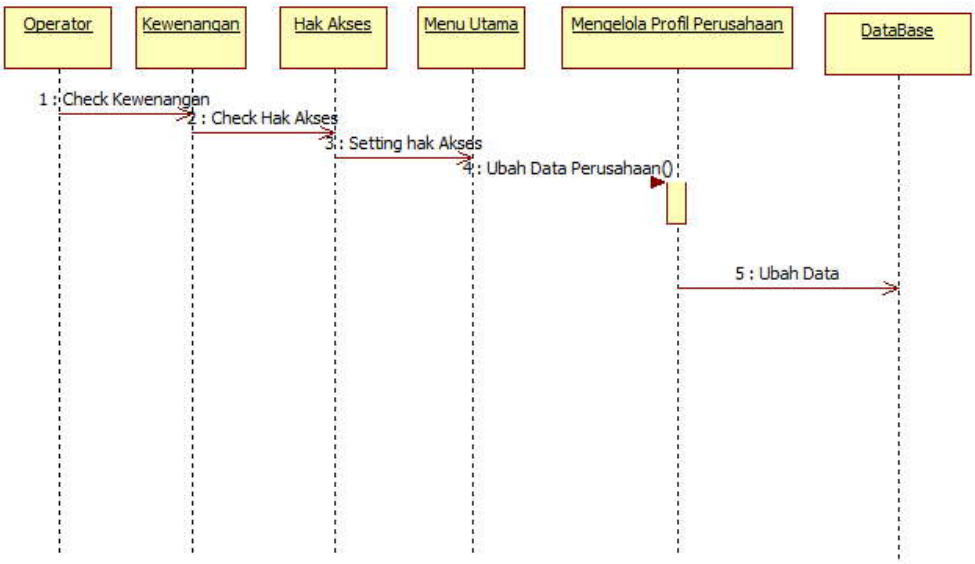
Gambar 16. *Sequence Diagram* Melakukan Setting Hak Akses

Sequence diagram di atas menjelaskan proses Operator melakukan setting Hak Akses pada *kewenangan* tiap operator. Hak Akses ini lah yang dipergunakan untuk mengantu apa saja yang dapat dilakukan oleh masing – masing operator.

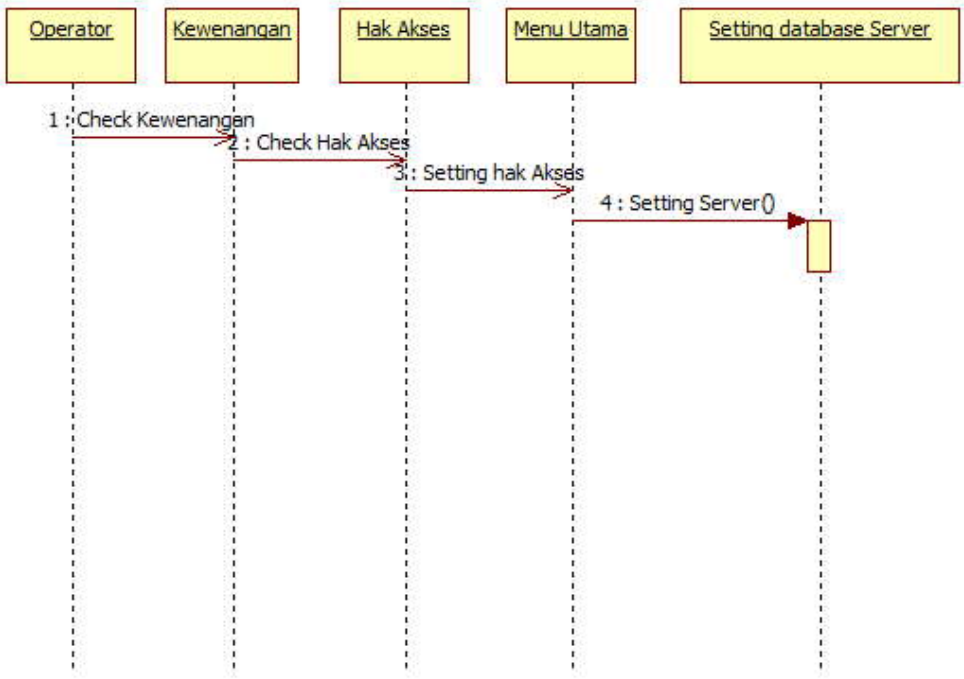


Gambar 17. *Sequence Diagram* Melakukan Pengisian Profil Perusahaan

Sequence diagram di atas menjelaskan proses Operator melakukan pengisian profil perusahaan dimana program nantinya akan di tanam. Profil Perusahaan tidak dapat dihapus melainkan hanya dapat diisi maupun dilakukan erbahan atas isinya seperti yang ditunjukan pada *sequence diagram* pada gambar 13.

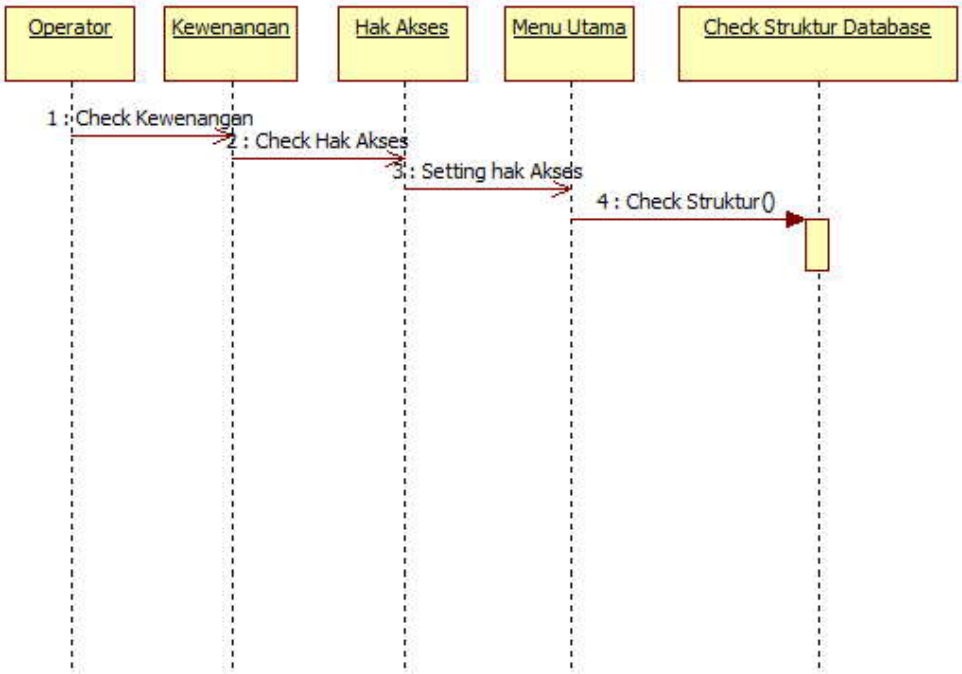


Gambar 18. *Sequence Diagram* Mengubah Profil Perusahaan



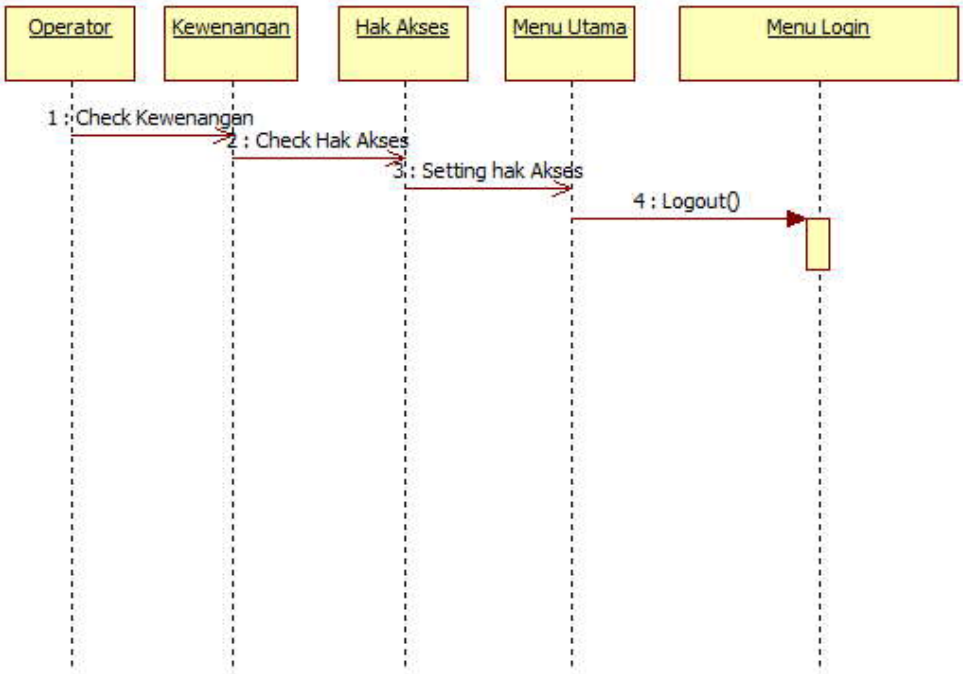
Gambar 19. *Sequence Diagram* Melakukan Setting database server

Sequence diagram di atas menjelaskan proses pengguna melakukan Setting Database Server. Proses ini hanya bisa *dilakukan* oleh pengguna yang memiliki level sebagai Hak Akses sesuai kewenangannya.Proses ini akan menunjuk server dimana database yang akan dijadikan server dengan memasukkan nama komputer server atau ip komputer server dilanjutkan dengan mangisi username dan password database, apabila sudah terjadi koneksi dan database tidak ditemukan maka program akan secara otomatis akan membuat database tersebut.



Gambar 20. *Sequence Diagram* Melakukan *Check* Struktur Database

Sequence diagram di atas menjelaskan proses pengguna melakukan Check Struktur Database. Proses ini hanya bisa dilakukan oleh pengguna yang memiliki level sebagai Hak Akses sesuai kewenangannya. Proses ini akan merubah struktur database atau memperbaiki struktur database yang tidak konsisten.



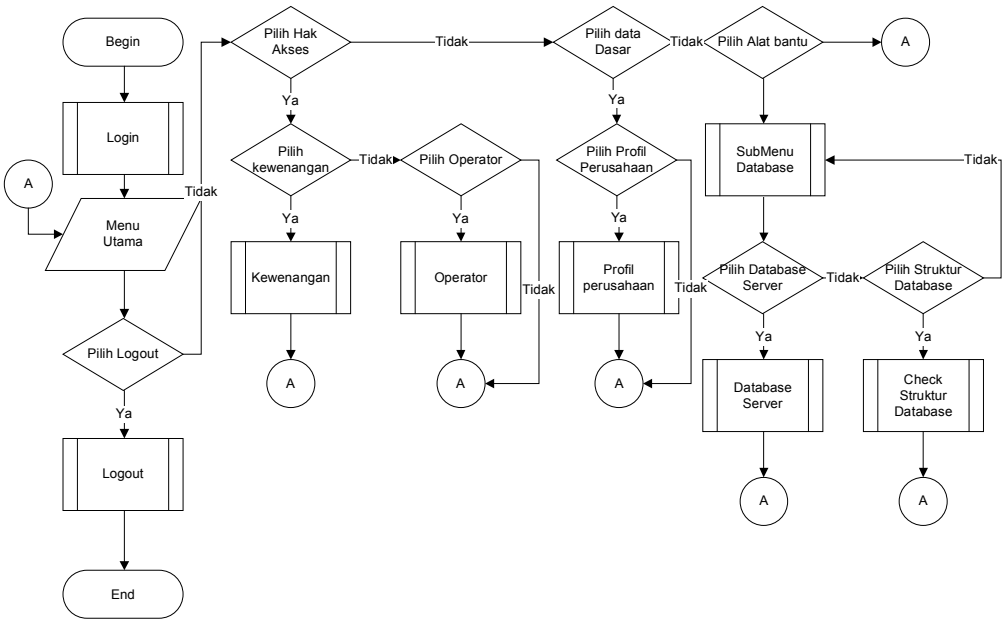
Gambar 21. *Sequence Diagram Logout*

Sequence diagram *Logout* bisa dilakukan oleh Operator dengan memilih keluar pada menu utama.

d. Desain Diagram Alir (Flowchart)

1) Menu Utama

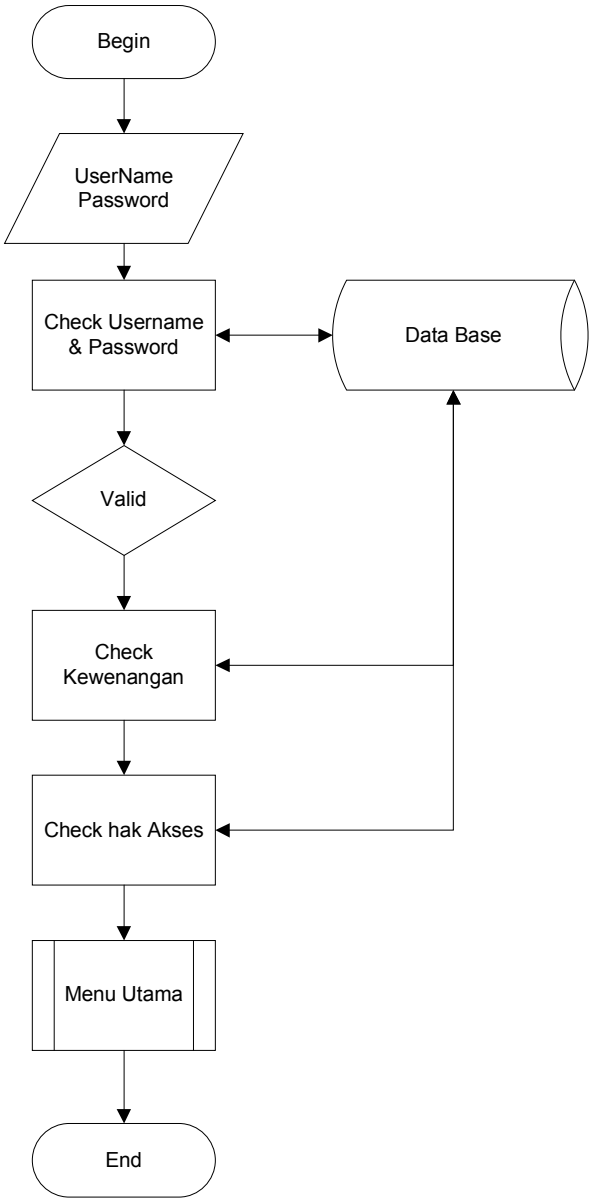
Proses pada Menu Utama memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 22. *Flowchart* Menu Utama

2) Login

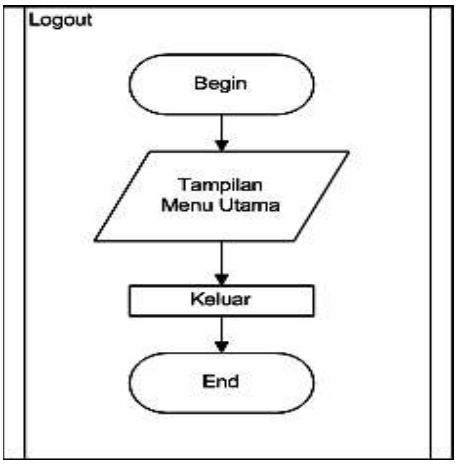
Proses pada Login memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 23. *Flowchart Login*

3) Logout

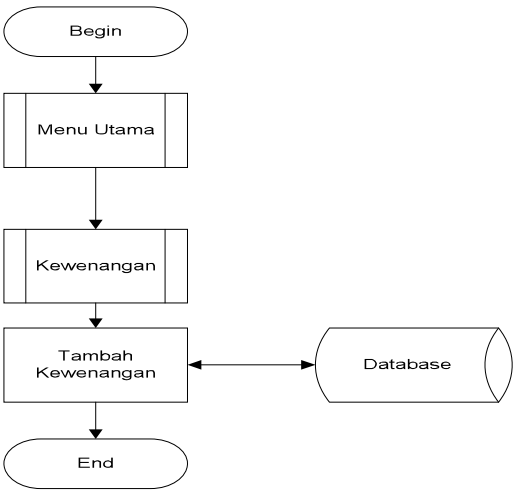
Proses pada Logout memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 24. *Flowchart Logout*

4) Menambah Kewenangan

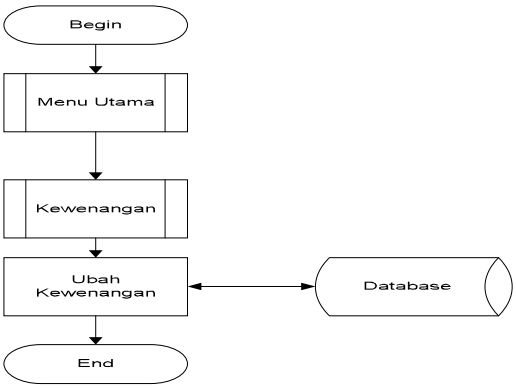
Proses pada Menambah data Kewenangan memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 25. *Flowchart* Menambah Data Kewenangan

5) Mengubah Kewenangan

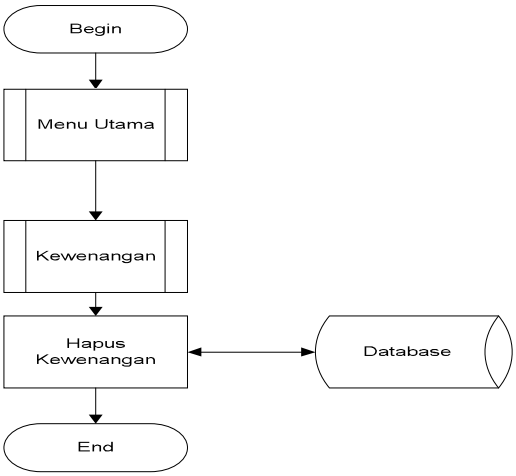
Proses pada Mengubah data Kewenangan memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 26. *Flowchart* Mengubah Data Kewenangan

6) Menghapus Kewenangan

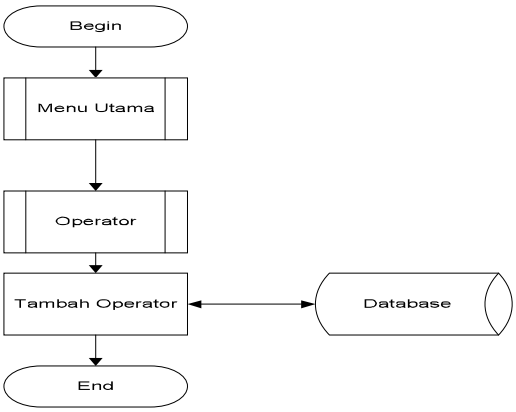
Proses pada Menghapus data Kewenangan memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 27. *Flowchart* Menghapus Data Kewenangan

7) Menambah Operator

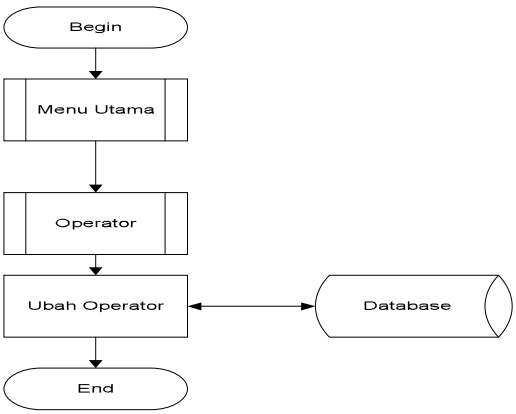
Proses pada Menambah data Operator memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 28. *Flowchart* Menambah Data Operator

8) Mengubah Operator

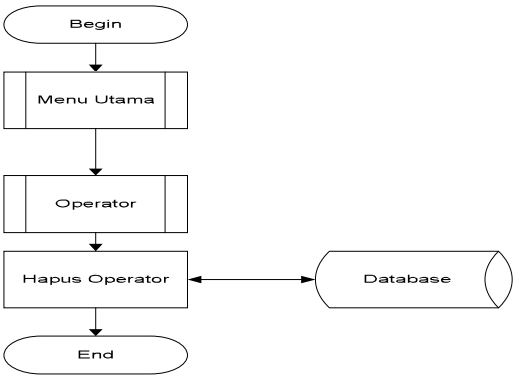
Proses pada Mengubah data Operator memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 29. *Flowchart* Mengubah Data Operator

9) Menghapus Operator

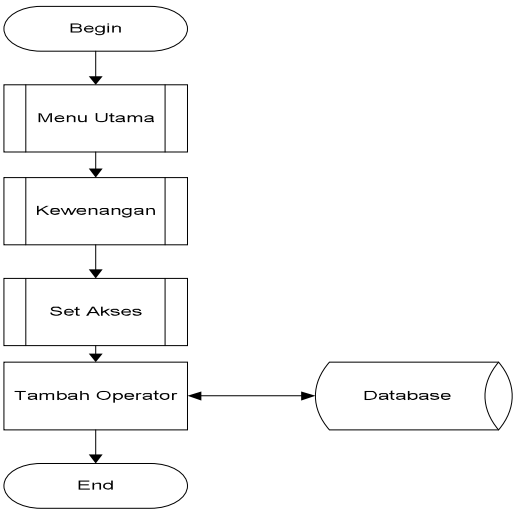
Proses pada Menghapus data Operator memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 30. *Flowchart* Menghapus Data Operator

10) Mengubah hak Akses

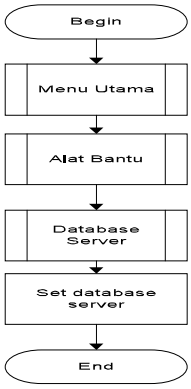
Proses pada Mengubah hak akses Operator memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 31. *Flowchart* Mengubah hak akses

11) Melakukan setting database server

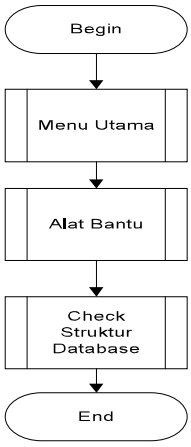
Proses pada Setting database server memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 32. *Flowchart* Set database Server

12) Melakukan Check Struktur Database

Proses pada check struktur database memiliki alur seperti pada gambar flowchart di bawah ini :



Gambar 33. *Flowchart* Check struktur database

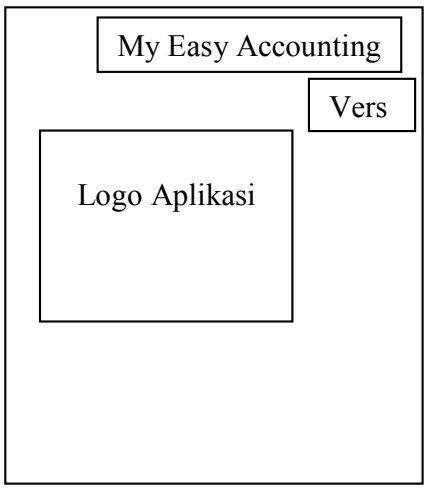
e. Perancangan Antarmuka

Desain dialog layar merupakan rancangan bangun komunikasi antara user dengan komputer. Proses komunikasi yang sering terjadi antara sistem dengan user dibentuk sedemikian rupa sehingga instruksi-instruksi yang ditempatkan pada layar dapat dipahami dengan baik oleh *user*.

Rancangan interface untuk perangkat lunak *PS Standart* terdiri dari 11 menu form untuk *form splash screen*, *form login*, *form* menu utama, *form* kewenangan, *form* tambah/ubah kewenangan, *form* set akses, *form operator*, *form* tambah/ubah *operator*, *form* profil perusahaan, *form database server* dan *form check struktur database*.

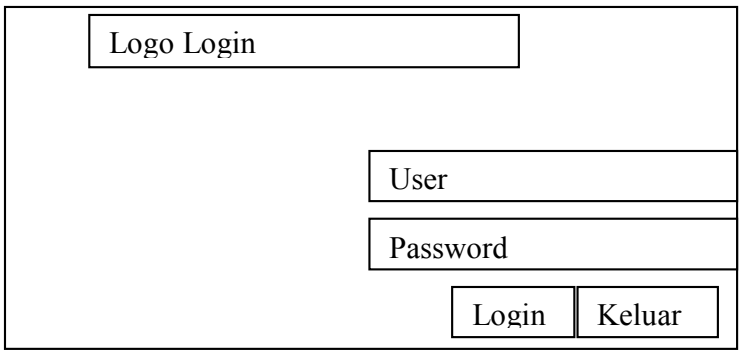
1) *Form Splash screen*

Form Splash screen adalah tampilan awal bagi user ketika perangkat lunak PS *Standart* dibuka untuk pertama kali**.** *Splash screen* akan tertampil selama 10 detik kemudian akan muncul *form* berikutnya.



Gambar 34. Rancangan Form Splash Screen

2) Form Login



Gambar 35. Rancangan Form Login

Form login berfungsi untuk pintu masuk pengguna ke perangkat lunak *PS Standart*. Untuk masuk ke perangkat lunak *PS Standart* dibutuhkan kode akses berupa user name dan password.

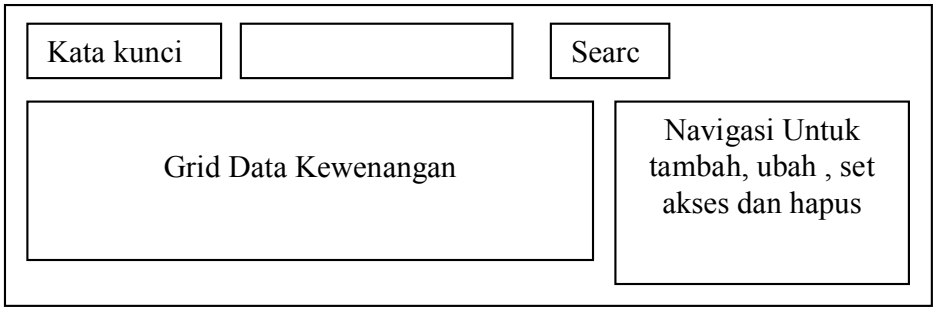
3) *Form* Menu Utama

Menu utama ini akan muncul setelah *user login* ke sistem.

Gambar 36. Rancangan Form Menu Utama

4) *Form* kewenangan

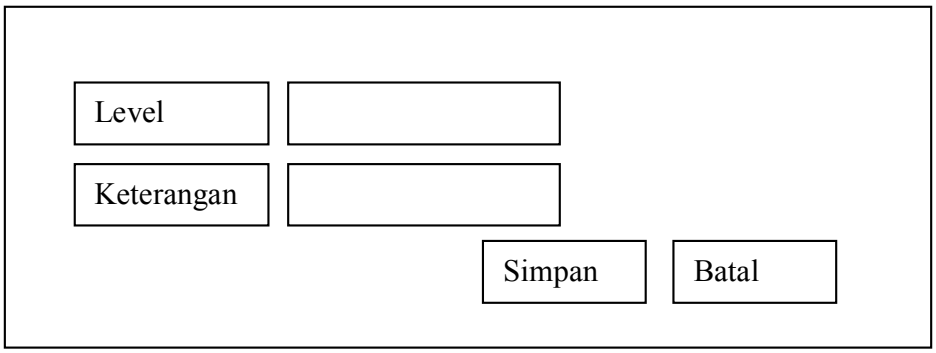
Pada *form* kewenangan ini *user* akan mengelola data kewenangan.



Gambar 37. Rancangan Form Kewenangan

5) *Form* tambah / ubah kewenangan

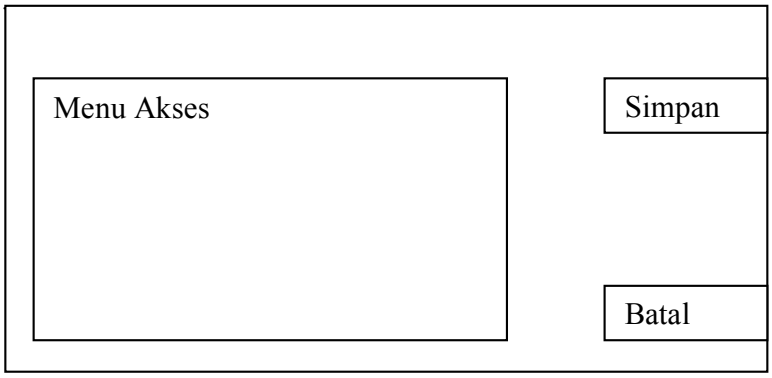
Pada *form* ini *user* akan dapat menambah/mengubah data kewenangan.



Gambar 38. Rancangan *Form* tambah / ubah kewenangan

6) *Form* set akses

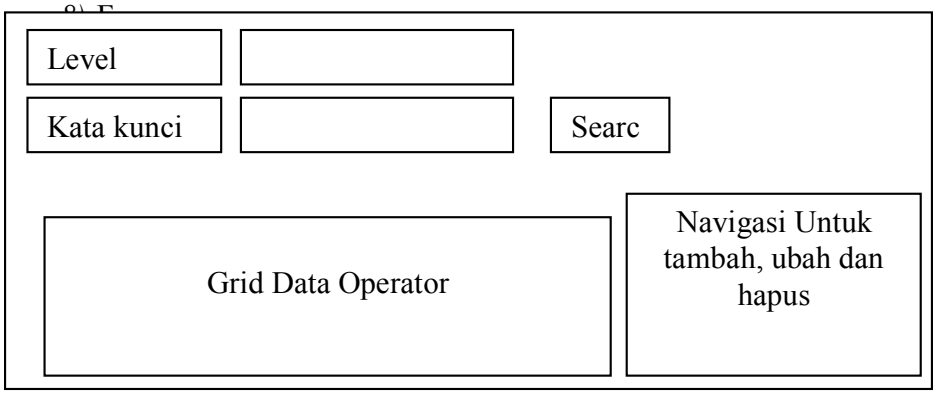
Jika *form* set akses di buka maka akan ditampilkan *tree* yang berisikan menu yang ada pada manu utama, untuk memberikan hak akses pada salah satu *operator* beri tanda centang pada cek box tree menu lalu klik simpan



Gambar 39. Rancangan *form* set akses

7) Form operator

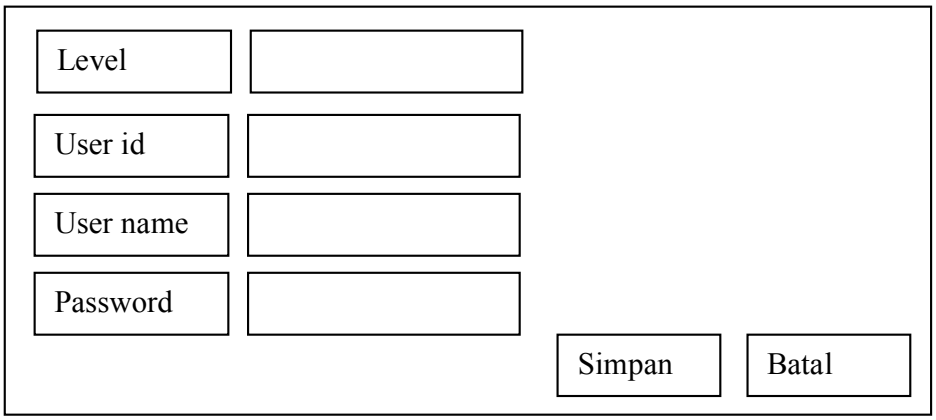
Pada *form operator* ini *user/ operator* akan mengelola data operator.



Gambar 40. Rancangan Form Operator

ambah / ubah *operator*

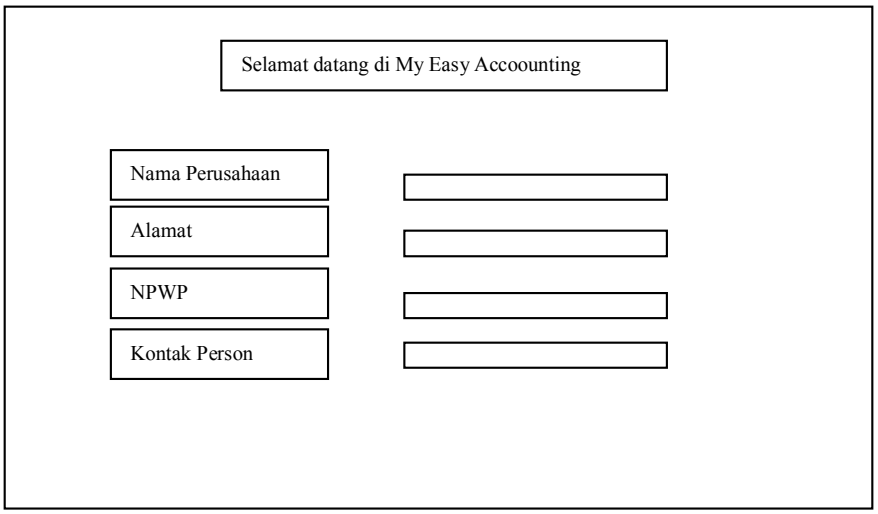
Pada *form* ini user akan menambah/mengubah data operator.



Gambar 41. Rancangan *Form* tambah / ubah *operator*

9) *Form* profil perusahaan

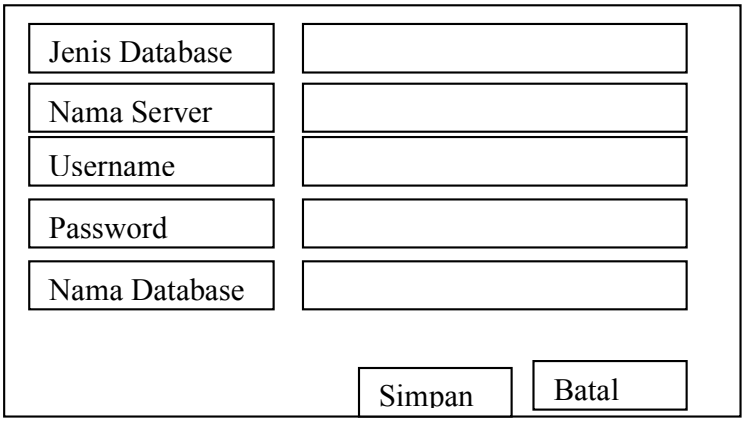
Pada *form* profil perusahaan ini *user / operator* akan memasukkan/mengubah profil perusahaan.



Gambar 42. Rancangan *form* profil perusahaan

10) Form database server

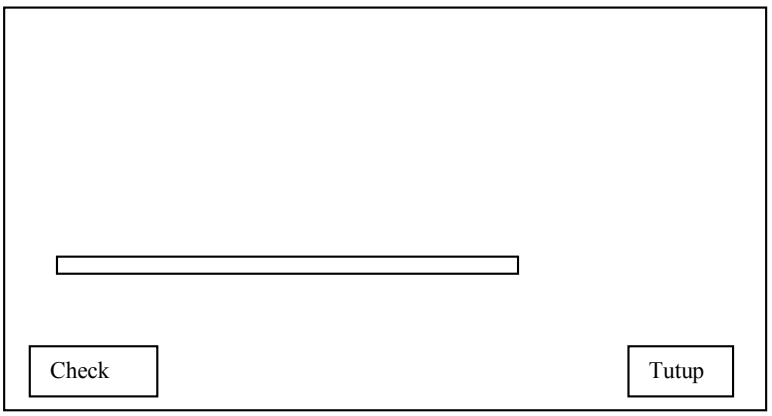
Form berikut adalah rancangan *form* untuk melakukan setting database server.



Gambar 43. Rancangan *form database server*

11) Form check struktur database

Form ini berfungsi untuk melakukan pengecekan struktur database dan memperbaiki struktur yang tidak konsisten.



Gambar 44. Rancangan Form Check Struktur Database

C. Teknik Pengumpulan Data

Ada beberapa teknik pengumpulan data yang digunakan untuk suatu penelitian, antara lain: (1) Metode Tes, (2) Metode Kuesioner atau Angket, (3) Metode Wawancara, (4) Metode Observasi (5) Metode Dokumentasi

(Suharsimi Arikunto, 2002:198-206). Teknik pengumpulan data dalam penelitian ini adalah metode angket atau kuesioner.

Kuesioner merupakan teknik pengumpulan data yang dilakukan dengan cara memberi seperangkat pertanyaan atau pertanyaan tertulis kepada responden untuk dijawab (Sugiyono, 2010:199). Jenis – jenis kuesioner dibagi dalam (Suharsimi Arikunto, 2002:128-129) :

1. Berdasarkan dari cara menjawab
   a. Kuesioner terbuka, yang member kesempatan kepada responden untuk menjawab dengan kalimatnya sendiri.
   b. Kuesioner tertutup, yang sudah disediakan jawabannya sehingga responden tinggal memilih.
2. Berdasarkan dari jawaban yang diberikan
   a. Kuesioner langsung, yaitu responden menjawab tentang dirinya.
   b. Kuesioner tidak langsung, yaitu jika responden menjawab tentang orang lain.
3. Berdasarkan dari bentuknya
   a. Kuesioner pilihan ganda yang dimaksud adalah sama dengan kuesioner tertutup
   b. Kuesioner isian, yang dimaksud adalah kuesioner terbuka
   c. Check list, sebuah daftar, dimana responden tinggal membubuhkan tanda check (√) pada kolom yang sesuai

d. Skala bertingkat, yaitu sebuah pernyataan diikuti oleh kolom-kolom yang menunjukan tingkat-tingkatan misalnya mulai dari sangat setuju sampai sangat tidak setuju.

Ditinjau dari beberapa jenis angket di atas, maka dalam penelitian ini jika dilihat dari cara menjawabnya menggunakan kuesioner tertutup, jika dilihat dari jawaban yang diberikan penelitian ini menggunakan kuesioner langsung, dan jika dilihat dari bentuknya penelitian ini menggunakan kuesioner check list.

D. Instrumen Penelitian

Instrumen penelitian adalah suatu alat yang digunakan mengukur fenomena alam maupun sosial yang diamati. Secara spesifik semua fenomena ini disebut variable penelitian (Sugiyono, 2010:148).

Instrumen yang dipakai peneliti adalah apa yang menjadi spesifikasi awal pada analisis kebutuhan menjadi acuan pembanding apakah semua spesifikasi yang direncanakan awal sudah terpenuhi dari aplikasi yang sudah dibuat. Penelitian ini akan menggunakan skala Guttman. Skala pengukur dengan tipe ini, akan didapat jawaban yang tegas, yaitu "ya-tidak", "benar-salah", "pernah-tidak pernah", "positif-negatif" dan lain-lain (Sugiyono, 2010: 139). Suharsimi Arikunto (2009: 107) mengemukakan bahwa pemilihan alternatif jawaban dapat disesuaikan pada keinginan dan kepentingan peneliti yang menciptakan instrumen. Berdasarkan pengertian tersebut, maka dalam penelitian ini akan menggunakan kriteria jawaban "ya-

tidak". Untuk keperluan analisis kuantitatif, maka jawaban itu dapat diberi skor. Rincian skor adalah sebagai berikut:

Ya : 1

Tidak : 0

Instrumen ini dipakai pada tahap implementasi dan pengujian. Dimana pengujian perangkat lunak ada beberapa macam, diantaranya adalah:

1. *Alpha Testing*

Alpha testing merupakan bagian dari validasi perangkat lunak yang sudah dibangun. Pengujian ini dilakukan oleh ahli *(expert judgment)* untuk mendapatkan penilaian unjuk kerja dari perangkat lunak *PS Standart.* Peneliti membuat instrumen untuk pengujian *Alpha* berdasarkan kesimpulan dari kebutuhan fungsional.

Tabel 21. Pengujian Aplikasi bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login*	1. Pengguna dapat masuk ke aplikasi dengan memasukkan *username* dan memasukkan *password* yang sesuai. 2. Jendela *Menu Utama* terbuka.		

Tabel 22. Pengujian Aplikasi bagian *Menu Utama*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Menu*	1. User dapat Logout dari menu utama. 2. User dapat login dengan user yang berbeda deng menu login dari menu utama.		

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
2.	*Data Datar* - *Kewenangan* - *Operator*	1. User dapat melihat data kewenangan dari menu kewenangan. 2. User dapat melihat data operator dari menuperator.		
3.	*Alat Bantu*	**1.** User dapat menjalankan menu database server. **2.** User dapat menjalankan menu check struktur database.		

Tabel 23. Pengujian Aplikasi bagian *menu Kewenangan*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Kewenangan*	1. Menampilkan data Kewenangan. 2. Mencari data kewenangan. 3. Menambah data kewenangan. 4. Mengubah data kewenanga. 5. Menghapus data kewenangan 6. Setting hak akses		

Tabel 24. Pengujian Aplikasi bagian *menu Operator*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Operator*	1. Menampilkan data Operator. 2. Mencari data Operator. 3. Menambah data Operator. 4. Mengubah data Operator. 5. Menghapus data Operator		

Tabel 25. Pengujian Aplikasi bagian *Database Server*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Database Server*	1. Dapat merubah komputer server. 2. Dapat memilih database.		

Tabel 26. Pengujian Aplikasi bagian Check Struktur *Database*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Check Struktur Database*	1. Dapat menjalankan check struktur database.		

Tabel 27. Pengujian Aplikasi menurut indikator *Correctness, Usability, Reliability* dan *Integrity*

No.	Hasil yang diharapkan	Tercapai	
		Ya	Tidak
1.	Aplikasi dapat dijalankan pada Sistem Operasi Windows		
2.	*User* / operator dapat menganses menu utama dengan login terlebih dahulu		
3.	Dibutuhkan waktu singkat untuk mempelajari fasilitas – fasilitas menu yang ada pada PS Standart		
4.	Melalui *menu Kewenangan* Operator dapat mengelola data kewenangan dan mengatur hak akses		
5.	Melalui *menu Operator,* Operrator dapat melihat informasi operator dan mengelolanya.		
6.	*PS Standart* mampu menangani kesalahan proses dengan menampilkan pesan kesalahan pada menu – menu tertentu yang mudah dimengerti pengguna		

2. Beta Testing

Beta testing merupakan bagian dari validasi tingkat akhir perangkat lunak yang sudah dibangun. Pengujian ini dilakukan oleh pengguna umum yang menjadi pengguna atau *operator*.

Peneliti membuat instrumen untuk pengujian *Beta* berdasarkan sudut pandang pengguna mengenai fasilitas yang ada dengan tujuan mendapatkan penilaian kelayakan dari perangkat lunak PS Standart untuk digunakan sebagai *template My Easy Accounting*.

Tabel 28. Kisi - kisi penilaian dari segi *Correctness, Usability, Reliability* dan *Integrity* menurut Mc Call untuk pengujian oleh Pengguna biasa

No	Aspek	Indikator	Butir
1	*Correctness*	Tingkat pemenuhan program terhadap kebutuhan yang dispesifikasikan dan memenuhi tujuan/ misi konsumer.	1, 2, 5
2	*Usability*	Usaha yang diperlukan untuk mempelajari, mengoperasikan, menyiapkan masukkan dan mengartikan keluaran oleh program.	3
3	*Reliability*	Tingkat kemampuan program yang diharapkan dapat menampilkan fungsi yang dimaksud dengan presisi yang ditetapkan.	6
4	*Integrity*	Tingkat kemampuan pengawasan akses terhadap data atau software oleh orang-orang tertentu.	4

Tabel 29. Pengujian Aplikasi untuk Operator

No.	Hasil yang diharapkan	Tercapai	
		Ya	Tidak
1.	Aplikasi dapat dijalankan pada Sistem Operasi Windows		
2.	*User* / operator dapat menganses menu utama dengan login terlebih dahulu		
3.	Dibutuhkan waktu kurang dari 1 jam untuk mempelajari fasilitas – fasilitas menu yang ada pada PS Standart		
4.	Melalui *menu Kewenangan* Operator dapat mengelola data kewenangan dan mengatur hak akses		
5.	Melalui *menu Operator,* Operator dapat melihat informasi operator dan mengelolanya.		
6.	*PS Standart* mampu menangani kesalahan proses dengan menampilkan pesan kesalahan pada menu – menu tertentu yang mudah dimengerti pengguna		

E. Teknik Analisis Data

Teknik analisis data yang digunakan dalam penelitian ini adalah teknik analisis data deskriptif kualitatif dan kuantitatif. Teknik analisis data deskriptif kualitatif digunakan untuk menganalisa data yang diperoleh dari hasil tabel pengujian alpha kemudian didiskrepsikan untuk dapat menentukan apakah sistem memiliki unjuk kerja yang baik atau buruk. Teknik analisis data deskriptif kuantitatif digunakan untuk menganalisa data yang diperoleh dari hasil tabel pengujian beta. Apabila data telah terkumpul, maka diklasifikasikan menjadi dua kelompok, yaitu data kuantitatif dan data kualitatif (Suharsimi Arikunto, 2002:213). Data kuantitatif dinyatakan dalam angka-angka dan data kualitatif dinyatakan dalam kata-kata dan simbol. Data yang bersifat kuantitatif yang berwujud angka – angka hasil perhitungan diproses dengan cara:

1. *Dijumlahkan*, dibandingkan dengan jumlah yang diharapkan dan diperoleh persentase;
2. *Dijumlahkan*, diklasifikasikan sehingga merupakan susunan urutan data (*array*) untuk selanjutnya dibuat tabel, maupun diproses lebih lanjut menjadi perhitungan pengambilan kesimpulan ataupun untuk kepentingan visualisasi datanya.

Berdasarkan data yang akan dikumpulkan melalui kuesioner yang bertujuan untuk mengetahui apakah sistem dapat berjalan dengan baik atau tidak, analisa data dilakukan dengan perhitungan teknik deskriptif kuantitatif dengan persentase dengan rumus sebagai berikut.

$$DP = \frac{n}{N} \times 100\ \%$$

Keterangan :

DP= Deskriptif Persentase (%)

n = Skor empirik (Skor yang diperoleh)

N = Skor ideal (Ditentukan berdasarkan spesifikasi sistem)

Apabila telah diperoleh persentase, maka dapat diketahui apakah perangkat lunak PS Standart memiliki kelayakan ditinjau dari segi *Correctness, Usability, Reliability* dan *Integrity*.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

A. HASIL PENELITIAN

1. Implementasi Pengkodean

Implementasi pengkodean adalah kelanjutan proses setelah desain. Fungsi atau *method* yang diperlukan diubah ke dalam bahasa pemrograman kemudian tiap unit langsung diuji. Ini dimaksudkan untuk menghindari kesalahan – kesalahan yang semakin besar. Pengkodean dilakukan secara bertahap sesuai dengan kebutuhan sistem.

Implementasi dilakukan setelah proses analisis kebutuhan sampai dengan desain perancangan sistem diverifikasi. Verifikasi bertujuan untuk menganalisa apakah kebutuhan dan desain sudah sesuai dengan konsep maupun teori – teori.

Kelas – kelas yang dibuat berdasarkan masing – masing objek, dan juga fungsi atau method. Kelas – kelas ini dikelompokkan menjadi beberapa menu. Menu utama adalah "*UMain"*, dan di dalamnya terdapat menu *"akses keluat (logout)","Keluar" "Kewenangan", "Operator", "Profil Perusahaan","Database Server", "Struktur Database", "Ufunc"* dan *"UDM"*. Penjelasan di bawah ini akan membahas pengkodean yang ada pada perangkat lunak *PS Standart*

.

a. Modul *UDM*

Modul UDM merupakan *modul*e pada perangkat lunak *PS Standart* untuk menyimpan kode – kode, variable global dan komponen – komponen pendukung yang dibutuhkan oleh perangkat lunak *PS Standart*.

Tabel 30. Fungsi dan *procedure* pada UDM

Fungsi dan Procedure
procedure DataSetFormat(DataSet: TDataSet);
function fnCheckDefaultData: Boolean;
procedure DataModuleDestroy(Sender: TObject);
procedure DataModuleCreate(Sender: TObject);
function fnSetIndex(TableName: string): TResultDB;
function fnSetTable(TableName: string): TResultDB;
function fnCheckDBStructure(ProgressBar: TProgressBar = nil; ACaption: TLabel = nil): Boolean;

b. Ufunc

Ufunc merupakan *form* pada perangkat lunak *PS Standart* untuk menyimpan kode – kode dan variable global yang dibutuhkan oleh perangkat lunak *PS Standart*

Tabel 31. Fungsi dan *procedure* pada UFunc

Fungsi dan Procedure
function fnCheckDB(ConStr: string; DBName: string): TConnectionResult;
function fnConStr: string;
function fnGetConnectionString(HostName, Catalog, UserName, Password: string): string;
function fnMessage(Text: string; MsgType: string): integer;
procedure prCreateForm(InstanceClass: TComponentClass; var Reference; ShowMode: string = 'SHOW');
procedure fnResetMenu(Permission: string; Menu: TMainMenu);

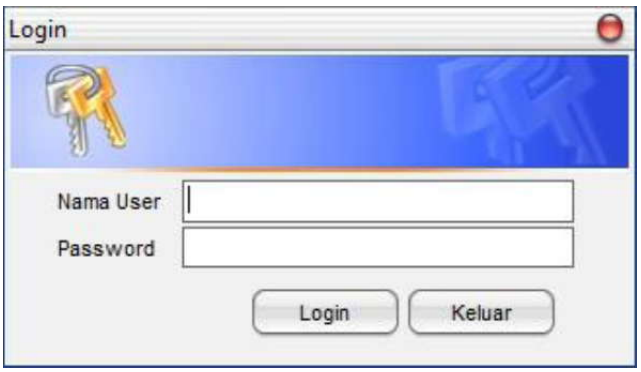
Fungsi dan Procedure
function fnBlowFishEncrypt(Src: string): string;
function fnBlowFishDecrypt(Src: string): string;
function fnRC2Encrypt(Str: string): string;
function fnRC2Decrypt(Str: string): string;
function fnGenerateId: string;
function fnFindValue(TableName, FieldValue, FieldCondition, Condition: string): string;
procedure prExplodeStr(SourceStr: string; Delimiter: char; var List: TStringList);
function fnCheckTableStructure(TableList: TStringList; ArrayTable: array of string; TableName: string): boolean;
function fnCheckIndexStructure(IndexList: array of string; TableName: string; TableList: TStringList; IndexName: string): boolean;
procedure fnSQLAdd(Query: TADOQuery; SQL: string; ClearPrior: boolean = False); overload;
procedure fnSQLAdd(Query: TADOCommand; SQL: string); overload;
procedure fnSQLAdd(Query: TADODataSet; SQL: string); overload;
procedure fnSQLOpen(Query: TADOQuery); overload;
procedure fnSQLOpen(Query: TADODataSet); overload;
procedure fnExecSQL(Query: TADOQuery); overload;
procedure fnExecSQL(Query: TADOCommand); overload;
procedure fnSQLParamByName(Query: TADOQuery; ParamStr: string; Value: Variant); overload;
procedure fnSQLParamByName(Query: TADOCommand; ParamStr: string; Value: Variant); overload;
procedure fnSQLParamByName(Query: TADODataSet; ParamStr: string; Value: Variant); overload;
procedure fnClearGrid(Stg: TAdvStringGrid; InitializeRow: integer = 2);
procedure fnCleanLog;
procedure fnStartTransaction;
procedure fnCommit;
procedure fnRollBack;
function fnInTransaction: boolean;
procedure fnRefreshDB(Qry: TADOQuery);
function fnGetServerDate: TDateTime;
function fnInputDateDB(Value: TDateTime; FromServer: boolean = True): string;

Fungsi dan Procedure
procedure fnCheckNetwork(E: Exception; FormSender: TForm = nil);
procedure fnWriteCrashLog(ErrorType: string);
procedure fnLogoutOperator;
function SQLConfigDataSource(HwndParent: HWND; FRequest: WORD; Driver: PChar; Attributes: Pchar): boolean; Stdcall;
function SQLGetInstalledDrivers(Size: string; Buff: WORD; BuffOut: WORD): boolean; Stdcall;
function FindWindowX(ACaption, AClass: string): THandle;
function IsCompressed: boolean;

c. Menu *Login*

Menu *login* akan muncul setelah *form splash* muncul selam 2 detik. Menu ini berfungsi untuk pintu masuk pengguna ke perangkat lunak *PS Standart*. Untuk masuk ke perangkat lunak *PS Standart* dibutuhkan hak akses berupa *user name* dan *password*. Tampilan pada gambar 41 dan 42 merupakan tapilan *form splash* dan menu *login*.

Gambar 45. *Splash Screen*



Gambar 46. Menu *login*

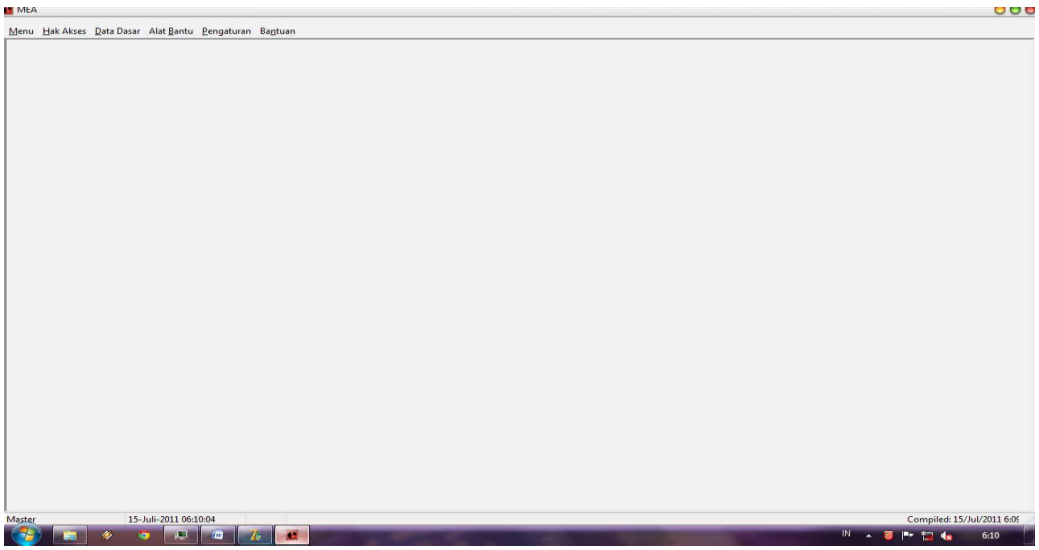
Penjelasan untuk objek dan method yang ada pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 32. Fungsi dan *procedure* pada menu *login*

Fungsi dan Procedure
procedure suitempbtnExitClick(Sender: TObject);
procedure suitempedtUserKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
procedure suitempbtnLoginClick(Sender: TObject);
procedure suitempbtnLoginKeyDown(Sender: TObject; var Key: Word;Shift: TShiftState);
procedure FormCreate(Sender: TObject);
procedure GantiWarna1Click(Sender: TObject);

d. Menu Utama

Menu utama ini akan muncul setelah user login ke sistem. Menu yang ditampilkan adalah menu sesuai kewenangan maupun hak akses setiap operator.



Gambar 47. Menu Utama

Penjelasan untuk method yang ada pada menu ini dapat dilihat pada tabel di bawah ini.

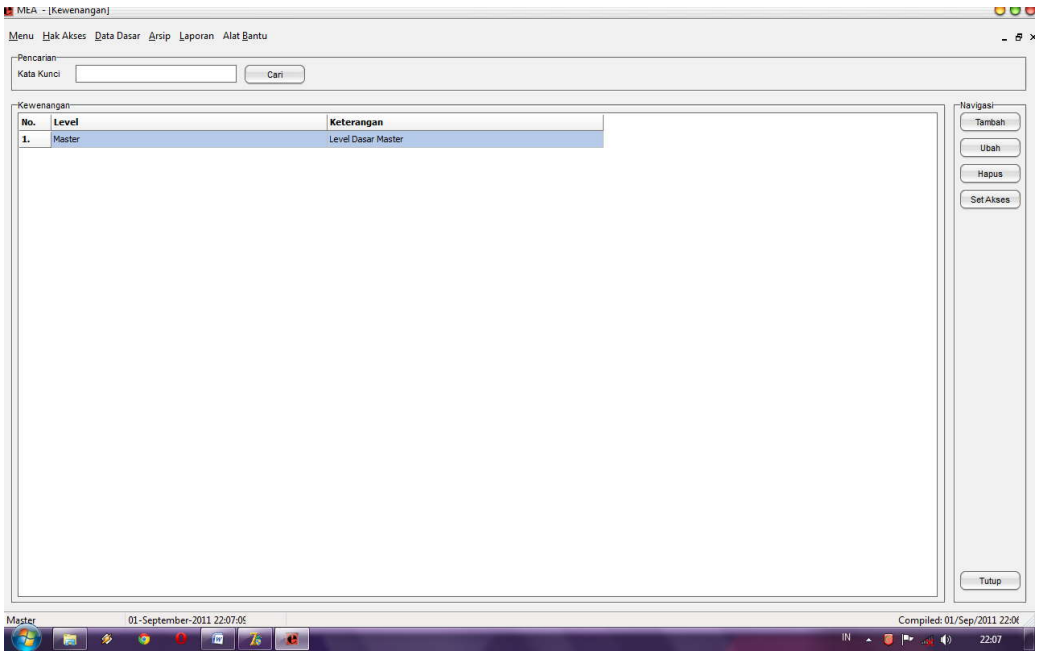
Tabel 33. Fungsi dan *procedure* pada menu utama

Fungsi dan Procedure
procedure FormCreate(Sender: TObject);
procedure AksesKeluar1Click(Sender: TObject);
procedure Keluar1Click(Sender: TObject);
procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
procedure Kewenangan1Click(Sender: TObject);
procedure DatabaseServer1Click(Sender: TObject);
procedure StrukturDatabase1Click(Sender: TObject);
procedure Operator1Click(Sender: TObject);
procedure Registrasi1Click(Sender: TObject);
procedure NomorKomputer1Click(Sender: TObject);
procedure ProfilPerusahaan1Click(Sender: TObject);
procedure Calculator1Click(Sender: TObject);
procedure Printer1Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);

```
procedure GeneralSetting1Click(Sender: TObject);
procedure General1Click(Sender: TObject);
procedure Profil1Click(Sender: TObject);
procedure ShowLogin;
procedure CloseForms;
```

e. Menu Kewenangan

Menu kewenangan ini *user/operator* akan mengelola data kewenangan yang nantinya dipakai untuk menentukan hak akses *user/operator* terhadap fasilitas yang ada dalam program.



Gambar 48. Menu Kewenangan

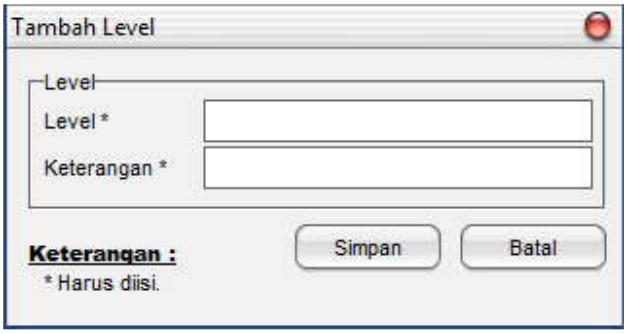
Penggunaan fungsi dan *procedure* pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 34. Fungsi dan *procedure* pada menu kewenangan

Fungsi dan Procedure
procedure Setting;
procedure ClearGrid;
procedure SearchData;
procedure FormCreate(Sender: TObject);
procedure suitempBtnCloseClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormResize(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure suitempbtnSearchClick(Sender: TObject);
procedure suitempButton1Click(Sender: TObject);
procedure suitempButton2Click(Sender: TObject);
procedure suitempButton3Click(Sender: TObject);
procedure suitempButton4Click(Sender: TObject);
procedure suitempButton5Click(Sender: TObject);
procedure suitempESearchKeyDown(Sender: TObject; var Key: Word;Shift: TShiftState);
procedure CmbLocChange(Sender: TObject);

Menu kewenangan terdapat tombol tambah dan ubah yang akan memanggil *form* UprivilegeAdd, dimana di dalam nya terdapat fungsi untuk membedakan ketika salah satu tombol tambah/ubah yang dipilih oleh *operator* sehingga dapat melakukan proses secara benar untuk melakukan panambahan data *level* kewenangan ataukan perubahan data dan juga terdapat tombol hapus untuk menghapus data

level kewenangan yang ada. Berikut adalah tampilan tambah/ubah level kewenangan.



Gambar 49. Tambah *level* kewenangan

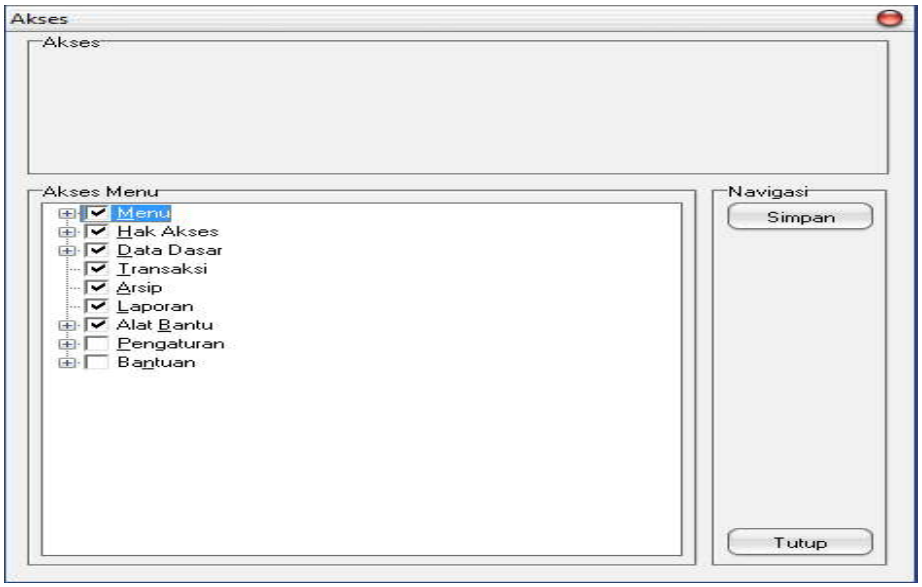
Gambar 50. Ubah *level* kewenangan

Penggunaan fungsi dan *procedure* pada form UprivilegeAdd ini dapat dilihat pada tabel di bawah ini.

Tabel 35. Fungsi dan *procedure* pada Tambah / Ubah Kewenangan

Fungsi dan Procedure
procedure suitempBtnCancelClick(Sender: TObject);
procedure suitempBtnSaveClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormActivate(Sender: TObject);
procedure suitempEdit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

Menu kewenangan terdapat tombol Set Akses yang akan memanggil *form* UPermission, dimana didalam nya terdapat fungsi untuk untuk merubah hak akses setiap *level* kewenangan sesuai dengan keperluan terhadap fasilitas yang ada dalam progam. Berikut adalah tampilan akses.



Gambar 51. Set Akses

Penggunaan fungsi dan *procedure* pada form UPermission ini dapat dilihat pada tabel di bawah ini.

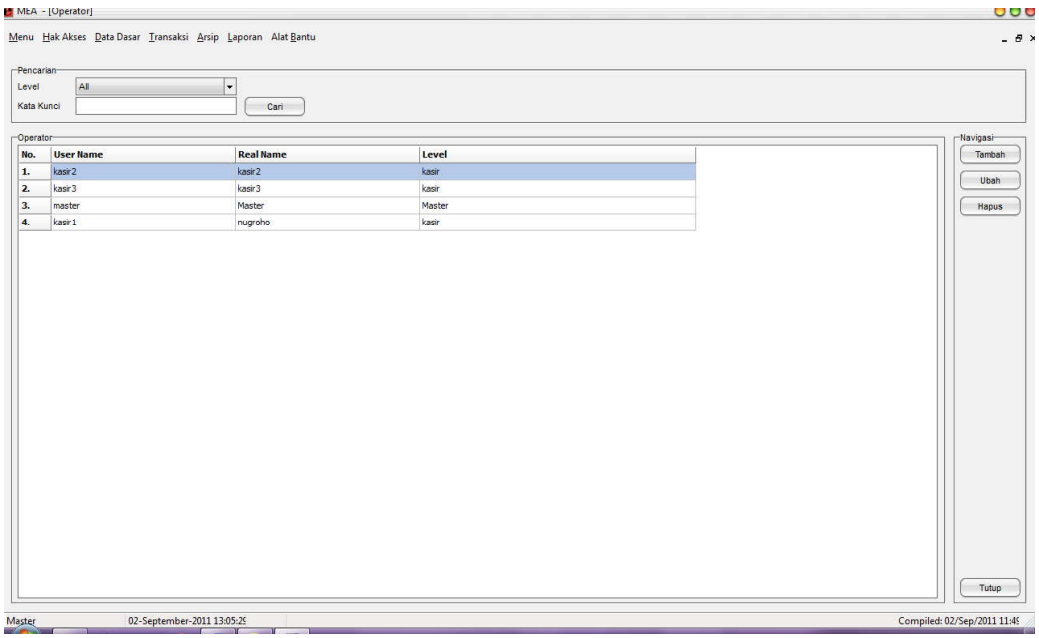
Tabel 36. Fungsi dan *procedure* pada set Akses

Fungsi dan Procedure
procedure CreateMenu(Item: TMenuItem; Parent: TTreeNode);
function CheckPermission(Treeview: THTMLTreeview): string;
procedure CheckMenu(Treeview: THTMLTreeview; StrPerm: string);
procedure FormCreate(Sender: TObject);
procedure suitempBtnCloseClick(Sender: TObject);
procedure FormClose(Sender: TObject; var

Fungsi dan Procedure
Action: TCloseAction);
procedure suitempButton1Click(Sender: TObject);
procedure HTMLTreeview1CheckBoxClick(Sender: TObject; Node: TTreeNode;Check: Boolean);
procedure FormActivate(Sender: TObject);

f. Menu *operator*

Menu *operator* ini *user/operator* akan mengelola data *operator* yang nantinya dipakai untuk melakukan *login*, sehinggan program dapat melakukan tindakan sesuai dengan level kewenangan dan hak akses yang diberikan atas fasilitas yang ada dalam program. Menu *operator* ini juga terdapat tombol tambah, ubah dan hapus yang berguna untuk mengolah data *operator*.



Gambar 52. Menu Operator

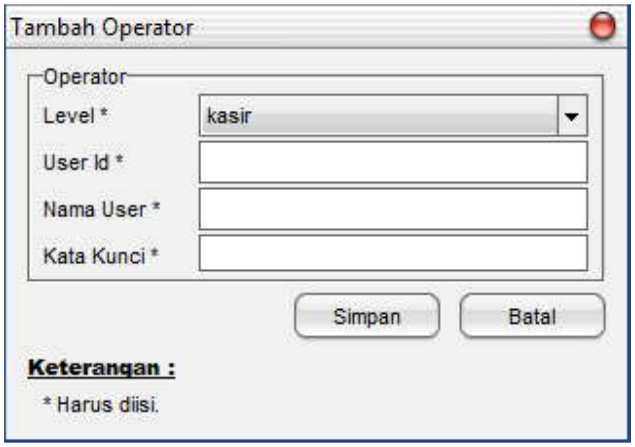
Penggunaan fungsi dan procedure pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 37. Fungsi dan *procedure* pada menu Operator

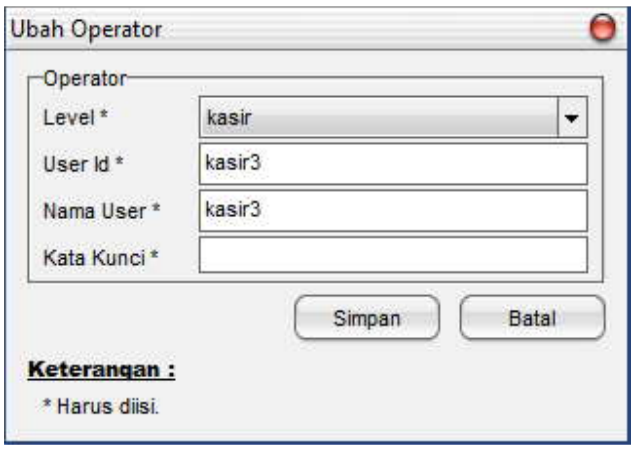
Fungsi dan Procedure
procedure Setting;
procedure SearchData;
procedure FillLevel;
procedure FormCreate(Sender: TObject);
procedure suitempBtnCloseClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormResize(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure suitempbtnSearchClick(Sender: TObject);
procedure suitempButton1Click(Sender: TObject);
procedure suitempButton2Click(Sender: TObject);
procedure suitempButton3Click(Sender: TObject);
procedure suitempComboBox1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure suitempESearchKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure suitempButton4Click(Sender: TObject);
procedure suitempGrbMainResize(Sender: TObject);
procedure CmbLocChange(Sender: TObject);
procedure suitempComboBox1Change(Sender: TObject);

Menu kewenangan terdapat tombol tambah dan ubah yang akan memanggil form UOperatorAdd, dimana didalam nya terdapat fungsi untuk membedakan ketika salah satu tombol tambah/ubah yang dipilih oleh operator sehingga dapat melakukan proses secara benar untuk

melakukan panambahan data *level* kewenangan ataukan perubahan data. Berikut adalah tampilan tambah/ubah *level* kewenangan.



Gambar 53. Tambah *operator*



Gambar 54. Ubah *operator*

Penggunaan fungsi dan *procedure* pada form UOperatorAdd ini dapat dilihat pada tabel di bawah ini.

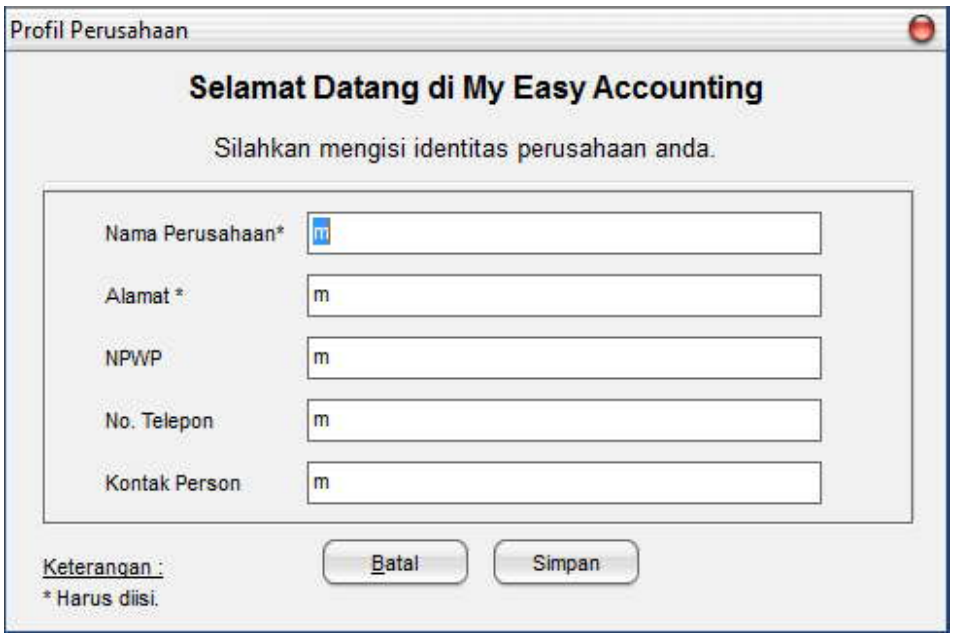
Tabel 38. Fungsi dan *procedure* pada Tambah / Ubah Operator

Fungsi dan Procedure
procedure Setting;
procedure suitempBtnCancelClick(Sender: TObject);
procedure suitempBtnSaveClick(Sender:

Fungsi dan Procedure
TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormActivate(Sender: TObject);
procedure suitempEdit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure FormResize(Sender: TObject);

g. Menu profil perusahaan

Menu profil perusahaan ini berfungsi untuk menyimpan data profil dari perusahaan atau tempat dimana program hasil pengembangan program PS *Standart* ini dipergunakan.



Gambar 55. Menu Profil Perusahaan

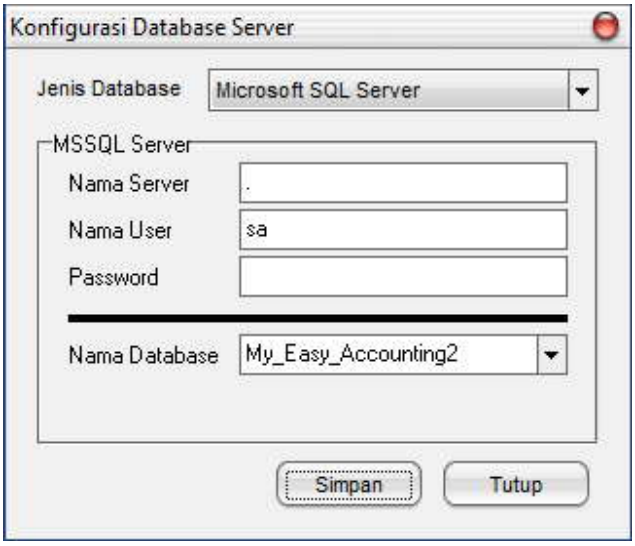
Penggunaan fungsi dan *procedure* pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 39. Fungsi dan *procedure* pada menu Profil Perusahaan

Fungsi dan Procedure
procedure FormCreate(Sender: TObject);
procedure suitempENameKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure suitempBtnCancelClick(Sender: TObject);
procedure suitempBtnNextClick(Sender: TObject);
procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure CleanUp;

h. Menu *database server*

Menu *database server* ini berfungsi untuk melakukan konfigurasi *database* untuk penyimpanan data, pada menu ini terdapat fasilitas untuk menentukan nama *database* maupun *server* yang dirujuk oleh program nantinya.



Gambar 56. Menu Konfigurasi Database Server

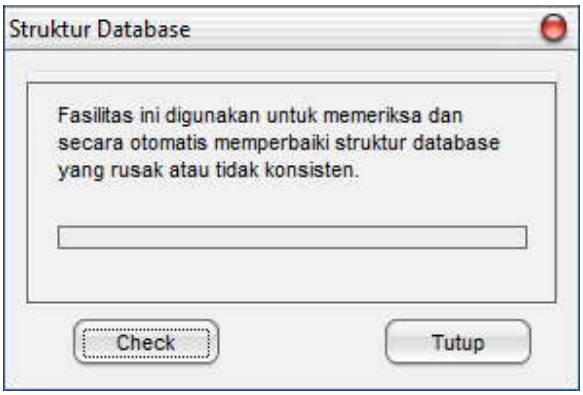
Penggunaan fungsi dan p*rocedure* pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 40. Fungsi dan *procedure* pada menu Konfigurasi Database Server

Fungsi dan Procedure
procedure Setting;
procedure suitempButton2Click(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure suitempEdtHostKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure suitempBtnSaveClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure suitempEDBNameEnter(Sender: TObject);
procedure suitempCmbDBTypeChange(Sender: TObject);
procedure suitempCmbServerTypeChange(Sender: TObject);
procedure suitempsuiButton1Click(Sender: TObject);

i. Menu struktur *database*

Menu struktur *database* ini berfungsi untuk melakukan pemeriksaan terhadap struktur *database* dan secara otomatis memperbaiki struktur database yang rusak atau tidak konsisten.



Gambar 57. Menu Struktur Database

Penggunaan fungsi dan *procedure* pada menu ini dapat dilihat pada tabel di bawah ini.

Tabel 41. Fungsi dan *procedure* pada menu Konfigurasi Database Server

Fungsi dan Procedure
procedure suitempButton2Click(Sender: TObject);
procedure suitempButton1Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);

2. Hasil Pengujian Terintegrasi

a. Black-Box Testing

Pengujian *black-box* adalah pengujian terintegrasi yang dilakukan oleh peneliti untuk memastikan bahwa sistem sudah siap

untuk diuji *alpha.* Pengujian *black-box* wajib dilakukan sebelum pengujian *alpha.* Peneliti melakukan pengujian *black-box software* PS *Standart* dengan membagi menjadi 13 bagian sebagaimana *use case* pada analisis kebutuhan dan perancangan sistem.

Tabel 42. Hasil pengujian *black-box*

No	Use Case	Aplikasi Bagian	Hasil Pengujian
1.	*Login*	Otentikasi	Sesuai
2.	*Logout*	Otentikasi	Sesuai
3.	Memasukkan data kewenanga	Menu kewenangan	Sesuai
4.	Mengubah data kewenangan	Menu kewenangan	Sesuai
5.	Menghapus data kewenanga	Menu kewenangan	Sesuai
6.	Mengubah akses kewenangan	Menu kewenangan	Sesuai
7.	Memasukkan data pengguna	Menu penguna	Sesuai
8.	Mengubah data pengguna	Menu penguna	Sesuai
9.	Menghapus data pengguna	Menu penguna	Sesuai
10.	Memasukkan dan mengubah data profil perusahaan	Menu profil Perusahaan	Sesuai
11.	Mengkonfigurasi database server	Menu Database Server	Sesuai
12.	Pemerikaan Struktur database	Menu Struktur database	Sesuai
13.	Keluar	Menu Keluar	Sesuai

b. Alpha Testing

Uji *alpha* adalah memvalidasi produk yang dilakukan oleh ahli. Ahli melakukan validasi dengan mengoreksi kesalahan – kesalahan dan kekurangan yang ada dalam produk, memberikan saran dan komentar serta rekomendasi untuk perbaikan. Hasil dari koreksi tersebut menjadi data yang akan digunakan untuk merevisi produk *software* PS *Standart.* Berikut adalah hasil pengujian yang dilakukan peneliti dengan beberapa ahli rekayasa *software*:

Tabel 43. Pengujian Aplikasi bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login*	1. Pengguna dapat masuk ke aplikasi dengan memasukkan username dan memasukkan password yang sesuai. 2. Jendela Menu Utama terbuka.	√	

Tabel 44. Pengujian Aplikasi bagian Menu Utama

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Menu*	1. User dapat Logout dari menu utama. 2. User dapat login dengan username yang berbeda dengan menu login dari menu utama.	√	
2.	*Data Dasar* - *Kewenangan* - *Operator*	1. User dapat melihat data kewenangan dari menu kewenangan. 2. User dapat melihat data operator dari menuperator.	√	
3.	*Alat Bantu*	**1.** User dapat melihat tampilan form menu database server. **2.** User dapat melihat tampilan form menu check struktur database.	√	

Tabel 45. Pengujian Aplikasi bagian *menu Kewenangan*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Kewenangan*	1. Menampilkan data Kewenangan. 2. Mencari data kewenangan. 3. Menambah data kewenangan. 4. Mengubah data kewenanga. 5. Menghapus data kewenangan 6. Setting hak akses	√	

Tabel 46. Pengujian Aplikasi bagian *menu Operator*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Operator*	1. Menampilkan data Operator. 2. Mencari data Operator. 3. Menambah data Operator. 4. Mengubah data Operator. 5. Menghapus data Operator	√	

Tabel 47. Pengujian Aplikasi bagian *Database Server*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Database Server*	1. Dapat merubah komputer server. 2. Dapat memilih database.	√	

Tabel 48. Pengujian Aplikasi bagian Check Struktur *Database*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Check Struktur Database*	1. Dapat menjalankan check struktur database.	√	

Tabel 49. Pengujian Aplikasi menurut indikator *Correctness, Usability, Reliability* dan *Integrity*

No.	Hasil yang diharapkan	Tercapai	
		Ya	Tidak
1.	Aplikasi dapat dijalankan pada Sistem Operasi Windows	√	
2.	*User* / operator dapat mengakses menu utama dengan login terlebih dahulu	√	
3.	Dibutuhkan waktu kurang dari 1 jam untuk mempelajari fasilitas – fasilitas menu yang ada pada PS Standart	√	
4.	Melalui *menu Kewenangan* Operator dapat mengelola data kewenangan dan mengatur hak akses	√	

No.	Hasil yang diharapkan	Tercapai	
		Ya	Tidak
5.	Melalui *menu Operator,* Operator dapat melihat informasi operator dan mengelolanya.	√	
6.	*PS Standart* mampu menangani kesalahan proses dengan menampilkan pesan kesalahan pada menu – menu tertentu yang mudah dimengerti pengguna	√	

Ahli rekayasa perangkat lunak memberikan kesimpulan bahwa perangkat lunak PS *Standart* yang telah peneliti bangun adalah mempunyai unjuk kerja yang baik.

c. *Beta Testing*

Pengujian *beta* dilakukan dengan melibatkan 10 operator yang telah ditentukan sebelumnya. Berikut adalah hasil pengujian *beta.*

Tabel 50. Hasil Uji Beta

Penguji	Aspek yang di uji						Jumlah
	1	2	3	4	5	6	
	Correctness		*Usability*	*Integrity*	*Correctness*	*Reliability*	
1	1	1	1	1	1	1	6
2	1	1	1	1	1	1	6
3	1	1	1	1	1	1	6
4	1	1	1	1	1	1	6
5	1	1	1	1	1	1	6
6	1	1	1	1	1	1	6
7	1	1	1	1	1	1	6
8	1	1	1	1	1	1	6
9	1	1	1	1	1	1	6
10	1	1	1	1	1	1	6
Jumlah	10	10	10	10	10	10	60

Keterangan :

Ya : 1

Tidak : 0

B. DESKRIPSI PROGRAM

S*oftware* PS *Standart* merupakan program yang nantinya akan dikembangkan lebih lanjut menjadi produk yang sesuai dengan kebutuhan pengembangan program di MB-LiF *Software laboratory*. S*oftware* ini menyediakan fasilitas – fasilitas umum yang dibutuhkan dalam pengembangan perangkat lunak di MB-LiF *Software Laboratory*.

1. Otentikasi

   Aplikasi bagian otentikasi merupakan bagian untuk mengatur keamanan sistem. Otentikasi ini meliputi *login*, *logout*. *Login* aplikasi berguna untuk membatasi operator atau pengguna dari pihah – pihak yang tidak memiliki kewenangan terhadap aplikasi.

2. Mengelola data Kewenangan

   Fasilitas ini dapat menambah, mengubah, memberi set akses dan menghapus *level* Kewenangan. Set akses merupakan fasilitas untuk memberikan *level* kewenangan atau mengurangi *level* kewenangan terhadap fasilitas yang ada dalam program. Fasilitas yang ada di dalam bagian ini antara lain :

   a). Menambah data level kewenangan baru

   b). Mengubah data kewenangan

   c). Menghapus data kewenangan

   d). Mencari data kewenangan

   e). Mengubah Set Akses

3. Mengelola data Operator

Fasilitas ini dapat menambah, mengubah, dan menghapus pengguna. Fasilitas yang ada di dalam bagian ini antara lain :

a). Menambah data *operator* baru

b). Mengubah data *operator*

c). Menghapus data *operator*

d). Mencari data *operator*

4. Mengelola data profil perusahaan

Fasitilas ini berguna untuk mencatat profil dari perusahaan atau instansi dimana program hasil pengembangan dari PS Standart ini tantinya dipergunakan. Fasilitas yang ada di dalam bagian ini antara lain :

a). Mengisi data profil perusahaan

b). Mengubah data profil perusahaan

5. Konfigurasi *database server*

Fasilitas ini operator yang memiliki kewenangan terhadap fasilitas ini, dapat melakukan konfigurasi dengan memilih dimana komputer *server* dan *database* yang dituju.

6. Struktur *database*

Fasilitas struktur *database* ini berfungsi untuk melakukan pemeriksaan terhadap struktur *database* dan secara otomatis memperbaiki struktur *database* yang rusak atau tidak konsisten .

C. Pembahasan

1 . Alpha Testing

Uji *alpha* yang dilakukan oleh ahli rekayasa *software* adalah untuk mendapatkan unjuk kerja dari *software* yang telah dibangun. Unjuk kerja didapat dengan menganalisa perangkat lunak dari spesifikasi yang diharapkan saat analisis kebutuhan.

Unjuk kerja yang diharapkan dari *software* PS *Standart* ini meliputi 8 bagian, yakni bagian (1) otentikasi, (2) bagian menu utama, (3)bagian menu Kewenangan, (4) bagian menu Operator, (5)bagian menu Profil perusahaan, (6)bagian menu database Server, (7)bagian menu Struktur database dan (8) pengujian menurut menurut indikator *correctness*, *usability, reliability* dan *integrity*. Keseluruhan dari hasil pengujian oleh ahli *software* menjadi bahan pertimbangan dan masukan yang berharga bagi peneliti.

Semua bagian secara keseluruhan sudah sesuai dengan spesifikasi harapan. Semua spesifikasi harapan tercapai dan ahli sudah setuju dengan unjuk kerja pada perangkat lunak ini. Namun ada beberapa masukan untuk bisa ditindak lanjuti oleh peneliti. Berdasarkan pengujian yang sudah dilakukan ahli, maka secara teoritis perangkat lunak PS Standart ini mempunyai unjuk kerja yang baik.

a. Revisi untuk *alpha*

Revisi awal dilakukan sesuai dengan saran ahli rekayasa perangkat lunak. Sesuai dengan hasil pengujian ahli rekayasa perangkat lunak, peneliti merevisi *software* PS *Standart* ini dengan memperbaiki pada masalah *logout* serta *counting* kesalahan *login*, dengan memperbaiki pada pengkodean.

Revisi kedua yang dilakukan oleh peneliti berupa perbaikan atas penanganan *progressbar* yang salah dengan memperbaiki pada pengkodean programnya

2*.** *Beta* ***Testing

Uji *beta* merupakan pengujian lanjutan dari uji *alpha.* Revisi awal dari uji *alpha* dilanjutkan dengan uji *beta.* Uji *beta* perangkat lunak PS Standart dilakukan oleh 10 pengguna. Uji *beta* ini untuk mendapatkan kelayakan ditinjau dari segi *Correctness, Usability, Reliability* dan *Integrity.*

Tabel 51. Skor kelayakan oleh Operator

Pengguna	Correctnes	Reliability	Usability	Integrity
1	3	1	1	1
2	3	1	1	1
3	3	1	1	1
4	3	1	1	1
5	3	1	1	1
6	3	1	1	1
7	3	1	1	1
8	3	1	1	1
9	3	1	1	1
10	3	1	1	1
n	30	10	10	10
N	30	10	10	10
DP(%)	**100**	**100**	**100**	**100**

$$DP = \frac{n}{N} \times 100\ \%$$

Keterangan :

DP= Deskriptif Persentase (%)

n = Skor empirik (Skor yang diperoleh)

N = Skor ideal (Ditentukan berdasarkan spesifikasi sistem)

Dari hasil pengujian oleh pengguna/ operator didapat hasil seperti dari tabel di bawah ini.

Tabel 52. Persentase skor kelayakan oleh Pengguna / Operator

	Correctnes	**Reliability**	**Usability**	**Integrity**
Operator (%)	100	100	100	100

Berdasarkan hasil perhitungan, maka dapat di gambarkan dengan diagram batang sebagai berikut.

a. Revisi Akhir

Software Project Source Standart (PS *Standart*) dikembangkan dengan menggunakan program utama *Borland Delphi 7*. *Software* PS *Standart* ini telah mengikuti tahap – tahap dalam pengembangan sesuai dengan tahap pengembangan *software* menurut Pressman. *Software* PS *Standart* ini telah selesai divalidasi oleh ahli rekayasa *software*.

Aplikasi divalidasi oleh ahli rekayasa *software*, kemudian dilakukan berbagai revisi terhadap perangkat lunak PS *Standart* sesuai dengan saran ahli yang bersangkutan sampai diperoleh hasil *software* PS *Standart* yang diharapkan. Penilaian dilakukan oleh pengguna dari berbagai kalangan yang terdiri dari 10 pengguna atau operator.

Software PS *Standart* yang dihasilkan dari penelitian pengembangan ini memiliki kelebihan dan kelemahan. Kelebihan dari *software* PS *Standart* adalah sebagai berikut:

1). Perangkat lunak ini dapat dijalankan oleh banyak pengguna dalam satu jaringan yang sama *(Multi User).*
2). Cukup satu *database server* untuk dimanfaatkan oleh semua pengguna dalam jaringan.
3). Terdapat fasilitas pengelolaan pengguna.
4). Perangkat lunak ini dapat memberikan level kewenangan kepada penggunanya.
5). Dapat melakukan *check Struktur database* dan otomatis memperbaikinya.
6). Perangkat lunak ini dapat secara otomatis membuat *database* yang datanya masih kosong apabila database yang ditunjuk tidak di ketemukan

Kelemahan aplikasi perangkat lunak PS Standart :

1). *Software* ini hanya bisa dijalankan pada komputer dengan sistem operasi Microsoft Windows.
2). Belum tersedianya fasilitas driver database server selain MS SQLServer.
3). Perintah SQL di dalam perangkat lunak ini hanya bekerja secara baik pada MS SQLServer.

BAB V
KESIMPULAN DAN SARAN

A. KESIMPULAN

Berdasarkan hasil penelitian pengembangan yang dilakukan pada pengembangan perangkat lunak yang dibangun, tentang "Perangakat Lunak *Project Source Standart* (PS *Standart*) di MB-LiF *Software Laboratory*" maka dapat disimpulkan sebagai berikut.

1. Pengembangan *software* PS *Standart* melalui tahapan analisis, desain dan implementasi menggunakan skrip pemrograman *Borland Delphi 7* sebagai template My Easy *Accounting* di MB-LiF *Software Laboratory*. Berdasarkan hasil pengujian dengan menggunakan *Black Box, Alpha Tesingt* dan *Beta Testing*, program dapat bekerja sesuai dengan spesifikasi pada analisa kebutuhan.
2. Penilaian ahli rekayasa perangkat lunak terhadap unjuk kerja *software* PS *Standart* menunjukan bahwa sistem memiliki unjuk kerja yang baik, semua sistem yang diujikan dapat berjalan dan bekerja sesuai dengan spesifikasi pada analisa kebutuhan.
3. Penilaian pengguna terhadap tingkat kelayakan *software* PS *Standart* mempunyai tingkat kelayakan 100 % dari segi *correctness*, 100 % dari segi *usability*, 100 % dari segi *reliability* dan 100 % dari segi *integrity*.

B. SARAN

Berdasarkan analisis perangkat lunak yang dibangun terdapat keterbatasan pada penggunaan *driver database* pada *software* PS *Standart*, karena *software* PS *Standart* yang dikembangkan masih terdapat banyak kekurangan sehingga peneliti menyarankan untuk menambahkan *driver* untuk *database* lain dan pengembangan lebih lanjut untuk memperoleh template program yang lebih baik lagi bagi MB-LiF *Software laboratory.*

DAFTAR PUSTAKA

Arikunto, Suharsimi. (2002). *Prosedur Penelitian, Suatu Pendekatan Praktek*. Jakarta: PT Rineka Cipta.

Arikunto, Suharsimi. (2009). *Manajemen Penelitian*. Jakarta: PT Rineka Cipta

Bertha Sidik, Ir . (2005). *MySQL Untuk Pengguna, Administrator, dan Pengembangan Aplikasi Web*. Bandung : Informatika.

Darsono, dkk. (2000). *Belajar dan Pembelajaran*. Semarang: IKIP Semarang

Pressman, S Roger, (2002). *Rekayasa Perangkat Lunak: pendekatan praktisi (Buku I) / Roger S. Pressman; Diterjemahkan oleh: LN Harnaningrum, Ed. II* – Yogyakarta : Andi.

Pressman, S Roger, (2002). *Rekayasa Perangkat Lunak: pendekatan praktisi (Buku II) / Roger S. Pressman; Diterjemahkan oleh: LN Harnaningrum, Ed. II* – Yogyakarta : Andi.

Shalahudin, M., A, S, Rosa. (2008). *Analisis dan Desain Sistem Informasi.* Bandung: Politeknik Telkom.

Sudjana, H. D. 2000. *Metode dan Tehnik Pembelajaran Partisipatif*. Bandung: Sinar Baru.

Sugihartono, dkk. (2007). *Psikologi Pendidikan*. Yogyakarta: UNY Press.

Sugiyono. (2010). *Metode Penelitian Pendidikan*. Bandung: Alfabeta.

Sujadi, 2002. *Metodologi Penelitian Pendidikan*. Jakarta: Rineka cipta.

Sukardi. (2004). *Metodologi Penelitian Pendidikan*. Jakarta: Bumi Aksara.

Universitas Negeri Yogyakarta. (2003). *Pedoman Tugas Akhir UNY.* Yogyakarta: Universitas Negeri Yogyakarta.

LAMPIRAN 1

PSStandart.dpr

```
program PSStandart;

uses
  Forms,
  Controls,
  UDBServer in 'UDBServer.pas' {FDBS erver},
  UFunc in 'UFunc.pas',
  UMain in 'UMain.pas' {FMain},
  UDM in 'UDM.pas' {dm: TDataModule},
  USplash in 'USplash.pas' {FSplash},
  ULogIn in 'ULogIn.pas' {FLogin},
  URegister in 'URegister.pas' {FRegister},
  UPrivillege in 'UPrivillege.pas' {FPrivillege},
  UPrivillegeAdd in 'UPrivillegeAdd.pas' {FPrivillegeAdd},
  UPermission in 'UPermission.pas' {FPermission},
  UCheckDB in 'UCheckDB.pas' {FCheckDB},
  UOperator in 'UOperator.pas' {FOperator},
  UOperatorAdd in 'UOperatorAdd.pas' {FOperatorAdd},
  UCashierSetting in 'UCashierSetting.pas' {FCashierSetting},
  UCompany in 'UCompany.pas' {FCompany},
  UPrinterSetting in 'UPrinterSetting.pas' {FPrinterSetting},
  UReconnectDB in 'UReconnectDB.pas' {FReconnectDB},
  UAbout in 'UAbout.pas' {FAbout},
  UFirstCash in 'UFirstCash.pas' {FFirstCash},
  UOSettingBScanTest in 'UOSettingBScanTest.pas' {FOBScanTest},
  UOSettingGeneral in 'UOSettingGeneral.pas' {FOSettingGeneral},
  USaveExcel in 'USaveExcel.pas' {FSaveExcel};

{$R *.res}

var Modal: integer;

begin
  Application.Initialize;
  vrKeyLocation := fnRC2Decrypt('72Q9IgY='); //'Tienz';
  vrDBName := fnRC2Decrypt('72Q9IgY='); //'Tienz';
  DBName := vrDBName;

  vrDSNNameFB := '/9k4+iRB9+vsdYxWlssETRTA';
  vrDriverNameFB := '/WY/v1OeH0AaI/8uiLgX6kVYWRSspABjcAS7bQ==';
  vrDSNNameMy := '/9k4+iRB9+vsdYxWlssETRTA5w==';
  vrDriverNameMy := '9g2KiHtmpWDajFyLL7cO9YUeuxR7jw==';

  if not IsCompressed then
  begin
    if FindWindowX(fnRC2Decrypt('/+8yhAjW2MI='),
fnRC2Decrypt('70yChnTz0Lz/ySLK')) = 0 then
    begin
```

```
fnMessage(fnRC2Decrypt('72X+Zn7qdxChN3aJt3W0U1ejVOUMUZsr2++GGXFZ8Xfz
Km5/IF5CxUzf5O9gFcT5SO+qAb8T4psprcxk'), mERROR);
      IsTerminated := True;
      Application.Terminate;
    end
  end;

  if not Isterminated then
  begin
    prCreateForm(TDM, DM, '');
    prCreateForm(TFMain, FMain, '');
  end;

  if not Isterminated then
  begin
    prCreateForm(TFSplash, FSplash, 'Show');
    FSplash.Update;

    ConStr := fnConStr;

    case fnCheckDB(ConStr, DBName) of
      crSuccess:
        begin
          DM.DB.Close;

          if ConnectionType = scMySQL then
          begin
            DM.DB.ConnectionString := ConStr;
            DM.DB.Open;

            try
              fnSQLAdd(DM.Cmd1, 'USE ' + DBName);
              fnExecSQL(DM.Cmd1);
            except
              IsTerminated := True;
            end
          end
          else
          begin
            DM.DB.ConnectionString := ConStr + ';Initial Catalog=' + DBName + ';';
            DM.DB.Open;
          end;
        end;
      crServerNotFound:
        begin
          fnMessage('Server Database tidak terkoneksi.'#13'Silakan mengkonfigurasi Server
Database.', mWARN);
          prCreateForm(TFDBServer, FDBServer, 'ShowModal');
        end;
```

```
      crDBNotFound:
        begin
          fnMessage('Server Database terhubung, tetapi Database ' + vrDBName + ' tidak
ditemukan.', mINFO);
          prCreateForm(TFDBServer, FDBServer, 'ShowModal');
        end;
    end;

    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM company', True);
    fnSQLOpen(dm.QryTemp1);

    if dm.QryTemp1.IsEmpty then
    begin
      prCreateForm(TFCompany, FCompany, '');
      Modal := FCompany.ShowModal;
      if Modal <> mrOK then
      begin
        IsTerminated := True;
        Application.Terminate
      end
    end
    else
    begin
      CompName := DM.qryTemp1.FieldByName('company_name').AsString;
      CompAddr := DM.qryTemp1.FieldByName('company_addr').AsString;
      CompTelp := DM.qryTemp1.FieldByName('company_telp').AsString;
      CompNPWP := DM.qryTemp1.FieldByName('company_npwp').AsString;
    end;
  end;

  if not IsTerminated then
  begin
    IsTerminated := not dm.fnCheckDefaultData;
    if IsTerminated then
      IsTerminated := not DM.fnCheckDBStructure;
  end;

  if not IsTerminated then
  begin
    prCreateForm(TFLogin, FLogin, 'ShowModal');
    if FLogin.LoginOK then
      FMain.Logined := True;
    FLogin.Cleanup;
  end;
  fnLoadLocalSetting;
  FSplash.Free;

  if IsTerminated then
  begin
    dm.Free;
    Application.Terminate
  end
  else
  begin
    Application.Run;
  end
end.
```

UDBServer.pas

```
unit UDBServer;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, DB, ADODB, ExtCtrls;

type
  TFDBServer = class(TForm)
    ADOConnection1: TADOConnection;
    Label9: TLabel;
    Label5: TLabel;
    Label8: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Shape1: TShape;
    OD: TOpenDialog;
    lblServerName: TLabel;
    BtnSave: TButton;
    BtnClose: TButton;
    CmbDBType: TComboBox;
    GroupBox2: TGroupBox;
    suiProgressBar1: TProgressBar;
    EDBNameFB: TEdit;
    CmbServerType: TComboBox;
    suiButton1: TButton;
    EServerName: TEdit;
    GroupBox1: TGroupBox;
    EdtHost: TEdit;
    EdtUserName: TEdit;
    EdtPassword: TEdit;
    EDBName: TComboBox;
    ProgressBar: TProgressBar;
    EPassword: TEdit;
    Label7: TLabel;
    ADOConnection2: TADOConnection;
    Label6: TLabel;
    EUsername: TEdit;
```

```
    GrbMySQL: TGroupBox;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Shape2: TShape;
    EMyServerName: TEdit;
    EMyUserName: TEdit;
    EMyPassword: TEdit;
    EMyDBName: TComboBox;
    ProgressBar1: TProgressBar;
    procedure Setting;
    procedure suitempButton2Click(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure suitempEdtHostKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure suitempBtnSaveClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure suitempEDBNameEnter(Sender: TObject);
    procedure suitempCmbDBTypeChange(Sender: TObject);
    procedure suitempCmbServerTypeChange(Sender: TObject);
    procedure suitempsuiButton1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FDBServer: TFDBServer;

implementation

uses UFUnc, UDM, UMain, StrUtils;

{$R *.dfm}

procedure TFDBServer.Setting;
begin
//  GroupBox1.Top := CmbDBType.Top + CmbDBType.Height + 5;
  BtnSave.Top := GroupBox1.Top + GroupBox1.Height + 8;
  BtnClose.Top := BtnSave.Top;
  ProgressBar.Top := BtnSave.Top;
  ClientWidth := GroupBox1.Width + (GroupBox1.Left * 2);
  ClientHeight := BtnClose.Top + BtnClose.Height + GroupBox1.Left;
end;

procedure TFDBServer.suitempButton2Click(Sender: TObject);
begin
  IsTerminated := True;
  Close
end;

procedure TFDBServer.FormResize(Sender: TObject);
begin
  Setting;
end;

procedure TFDBServer.FormCreate(Sender: TObject);
var Host: string;
  MyList: TStringList;
begin
  case ConnectionType of
    scSQLServer:
      begin
        EdtHost.Text := fnRC2Decrypt(fnReadReg('HostName', 'String'));
        CmbDBType.ItemIndex := 0;

        if EdtHost.Text = '' then
        begin
          EdtHost.Text := '(local)';
        end;

        EdtUserName.Text := fnRC2Decrypt(fnReadReg('UserName', 'String'));
        if EdtUserName.Text = '' then
        begin
          EdtUserName.Text := 'sa';
        end;

        GroupBox2.Hide;
        GroupBox1.Show;
        GrbMySQL.Hide;
      end;
    scFirebird:
      begin
        Host := fnRC2Decrypt(fnReadReg('HostName', 'String'));
        CmbDBType.ItemIndex := 1;
        if (Pos('Local', Host) > 0) or (Host = '') then
        begin
          CmbServerType.ItemIndex := CmbServerType.Items.IndexOf('Local');
          EServerName.Clear;
          EServerName.Enabled := False;
        end
        else
        begin
          CmbServerType.ItemIndex := CmbServerType.Items.IndexOf('Remote');

          MyList := TStringList.Create;
          prExplodeStr(Host, ';', MyList);
```

```
          EServerName.Text := MyList[1];
          EServerName.Enabled := True;

          FreeAndNil(MyList)
        end;

        GroupBox2.Show;
        GroupBox1.Hide;
        GrbMySQL.Hide;

        CmbDBType.ItemIndex := 1;
      end;
    scMySQL:
      begin
        EMyServerName.Text := fnRC2Decrypt(fnReadReg('HostName', 'String'));
        CmbDBType.ItemIndex := 2;

        if EMyUserName.Text = '' then
        begin
          EMyUserName.Text := 'root';
        end;

        GroupBox2.Hide;
        GroupBox1.Hide;
        GrbMySQL.Show;
      end;
  end;

  EDBName.Text := DBName;
  EDBNameFB.Text := DBName;
  EMyDBName.Text := DBName;
end;

procedure TFDBServer.suitempEdtHostKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then Perform(WM_NEXTDLGCTL, 0, 0)
end;

procedure TFDBServer.suitempBtnSaveClick(Sender: TObject);
var
  S: TCustomMemoryStream;
  TryConStr: string;
  TempConType: TSQLConnection;
begin
  if CmbDBType.Text = 'Firebird' then
  begin
    ConnectionType := scFirebird;
```

```
    if EDBNameFB.Text = '' then
    begin
      fnMessage('Nama Database tidak boleh kosong.', mINFO);
      EDBNameFB.SetFocus;
      Exit;
    end;
  end
  else if CmbDBType.Text = 'Microsoft SQL Server' then
  begin
    ConnectionType := scSQLServer;

    if EDBName.Text = '' then
    begin
      fnMessage('Nama Database tidak boleh kosong.', mINFO);
      EDBName.SetFocus;
      Exit;
    end;
  end
  else if CmbDBType.Text = 'MySQL' then
  begin
    ConnectionType := scMySQL;

    if EMyDBName.Text = '' then
    begin
      fnMessage('Nama Database tidak boleh kosong.', mINFO);
      EMyDBName.SetFocus;
      Exit;
    end;
  end;

  btnSave.Enabled := false;
  TempConType := ConnectionType;

  case ConnectionType of
    scSQLServer:
      begin
        TryConStr := fnGetConnectionString(Trim(EdtHost.Text), '',
Trim(EdtUserName.Text),
          Trim(EdtPassword.Text));

        DBName := Trim(EDBName.Text);
      end;
    scFirebird:
      begin
        try
          SQLConfigDataSource(0, ODBC_ADD_SYS_DSN,
            PChar(fnRc2Decrypt(vrDriverNameFB)),
            PChar(fnRC2Decrypt(vrDSNNameFB)))
        except
        end;
```

```
      DBName := Trim(ChangeFileExt(EDBNameFB.Text, '.FDB'));
      if ExtractFilePath(DBName) = '' then
        DBName := ExtractFilePath(Application.ExeName) + DBName;

      if CmbServerType.Text = 'Local' then
      begin
        if not FileExists(DBName) then
        begin
          if fnMessage('File database tidak ditemukan. Buat baru ?', mCONFIRM) =
IDYES then
          begin
            try
              S := TResourceStream.Create(HInstance, fnRC2Decrypt('5JB2jQ==',
PChar(fnRC2Decrypt('6aqBqq0='))));
              S.SaveToFile(DBName);

              FreeAndNil(S);
            except
              on E: Exception do
              begin
                fnMessage('Database Firebird gagal dibuat '#13'Error: ' + E.Message,
mERROR);
                Exit;
              end;
            end;
          end
        end;
      end
      else
      begin
        DBName := Format('%s:%s', [EServerName.Text, DBName]);
      end;

      TryConStr := fnGetConnectionString(fnRc2Decrypt('19cZA88lVPodnDBv'),
        DBName, Trim(EUserName.Text), Trim(EPassword.Text));
    end;
  scMySQL:
    begin
      try
        SQLConfigDataSource(0, ODBC_ADD_SYS_DSN,
          PChar(fnRc2Decrypt(vrDriverNameMy)),
          PChar(fnRC2Decrypt(vrDSNNameMy)))
      except
      end;

      DBName := Trim(EMyDBName.Text);
      TryConStr := fnGetConnectionString(Trim(EMyServerName.Text),
        '', Trim(EMyUserName.Text), Trim(EMyPassword.Text));
    end;
```

```
    end;

  case fnCheckDB(TryConStr, DBName) of
    crSuccess:
      begin
        case ConnectionType of
          scSQLServer:
            begin
              fnWriteReg('HostName', 'String', fnRC2Encrypt(Trim(EdtHost.Text)));
              fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EdtUserName.Text)));
              fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EdtPassword.Text)));
              fnWriteReg('DBName', 'String', fnRC2Encrypt(DBName));
              fnWriteReg('Database', 'String', fnRC2Encrypt(CmbDBType.Text));
            end;
          scFirebird:
            begin
              if CmbServerType.Text = 'Local' then
                fnWriteReg('HostName', 'String', fnRc2Decrypt(CmbServerType.Text))
              else
                fnWriteReg('HostName', 'String', fnRc2Decrypt(CmbServerType.Text + ';' +
EServerName.Text));

              fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EUserName.Text)));
              fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EPassword.Text)));
              fnWriteReg('DBName', 'String', fnRC2Encrypt(DBName));
              fnWriteReg('Database', 'String', fnRC2Encrypt(CmbDBType.Text));

              DM.DB.Close;
              DM.DB.ConnectionString := TryConStr;
              DM.DB.Open;
              if not DM.fnCheckDBStructure(suiProgressBar1) then
              begin
                fnMessage('Konfigurasi Database gagal disimpan.', mERROR);
                IsTerminated := True;
                Application.Terminate;
                Exit
              end;
            end;
          scMySQL:
            begin
              fnWriteReg('HostName', 'String', fnRC2Encrypt(Trim(EMyServerName.Text)));
              fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EMyUserName.Text)));
              fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EMyPassword.Text)));
              fnWriteReg('DBName', 'String', fnRC2Encrypt(DBName));
              fnWriteReg('Database', 'String', fnRC2Encrypt(CmbDBType.Text));
            end;
        end;
        fnMessage('Konfigurasi Database telah disimpan.'#13'Jalankan kembali aplikasi ini.',
mINFO);
        IsTerminated := True;
```

```
      Application.Terminate;
      Exit
    end;
  crServerNotFound: fnMessage('Server Database tidak terkoneksi.'#13'Silakan
mengkonfigurasi Server Database.', mWARN);
  crDBNotFound:
    begin
      if fnMessage('Server Database terhubung, tetapi Database "' + DBName + '" tidak
ditemukan.'#13'Buat Database baru ?', mCONFIRM) = IDYES then
      begin
        DM.DB.Close;
        DM.DB.ConnectionString := TryConStr;
        DM.DB.Open;
        case ConnectionType of
          scSQLServer: DBName := EDBName.Text;
          scMySQL: DBName := EMyDBName.Text;
          scFirebird: DBName := EDBNameFB.Text;
        end;

        if DM.fnCheckDBStructure(ProgressBar) then
        begin
          case ConnectionType of
            scSQLServer:
              begin
                fnWriteReg('HostName', 'String', fnRC2Encrypt(Trim(EdtHost.Text)));
                fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EdtUserName.Text)));
                fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EdtPassword.Text)));
                fnWriteReg('DBName', 'String', fnRC2Encrypt(Trim(EDBName.Text)));
              end;
            scFirebird:
              begin
                fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EUserName.Text)));
                fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EPassword.Text)));

                if CmbServerType.Text = 'Local' then
                  fnWriteReg('HostName', 'String', fnRc2Decrypt(CmbServerType.Text))
                else
                  fnWriteReg('HostName', 'String', fnRc2Decrypt(CmbServerType.Text + ';' +
EServerName.Text));

                fnWriteReg('DBName', 'String', fnRC2Encrypt(Trim(EDBNameFB.Text)));
              end;
            scMySQL:
              begin
                fnWriteReg('UserName', 'String', fnRC2Encrypt(Trim(EMyUserName.Text)));
                fnWriteReg('Password', 'String', fnRC2Encrypt(Trim(EMyPassword.Text)));
                fnWriteReg('HostName', 'String',
fnRC2Encrypt(Trim(EMyServerName.Text)));
                fnWriteReg('DBName', 'String', fnRC2Encrypt(Trim(EMyDBName.Text)));
              end;
```

```
            end,
            fnWriteReg('Database', 'String', fnRC2Encrypt(CmbDBType.Text));
            fnMessage('Konfigurasi Database telah disimpan.'#13'Jalankan kembali aplikasi
ini.', mINFO);
//          ShellExecute(handle, 'open', Pchar(Application.ExeName), nil, nil,
SW_SHOWMAXIMIZED);
            IsTerminated := True;
            Application.Terminate;
            Exit
          end
          else
            fnMessage('Konfigurasi Database gagal disimpan.', mERROR);
        end
        else
        begin
          dm.DB.Close;
          IsTerminated := True;
          Exit
        end;
      end,
  end;

  ConnectionType := TempConType; // kembalikan tipe database lama
  btnSave.Enabled := true;
end;

procedure TFDBServer.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FDBServer := nil
end;

procedure TFDBServer.suitempEDBNameEnter(Sender: TObject);
var
  TryConStr: string;
  TryDB: TADOConnection;
  TempName: string;
  Qry: TADOQuery;
  TempConType: TSQLConnection;
begin
  TempConType := ConnectionType;

  if UpperCase(CmbDBType.Text) = 'FIREBIRD' then
    ConnectionType := scFirebird
  else if UpperCase(CmbDBType.Text) = 'MICROSOFT SQL SERVER' then
    ConnectionType := scSQLServer
  else if UpperCase(CmbDBType.Text) = 'MYSQL' then
    ConnectionType := scMySQL;

  TryConStr := fnGetConnectionString(Trim(EdtHost.Text), '', Trim(EdtUserName.Text),
```

```
    Trim(EdtPassword.Text));
  case fnCheckDB(TryConStr, '') of
    crSuccess: begin
        TryDB := TADOConnection.Create(Self);
        TryDB.ConnectionString := TryConStr;
        TryDB.LoginPrompt := False;
        TryDB.Open;

        Qry := TADOQuery.Create(Self);
        Qry.Connection := TryDB;

        case ConnectionType of
          scSQLServer:
            begin
              TempName := IfThen(EDBName.Text = '', EDBName.Text, DBName);

              TryDB.GetTableNames(EDBName.Items);

              fnSQLAdd(Qry, 'SELECT * FROM sysdatabases WHERE name NOT IN
("master",' +
                '"model", "msdb", "Northwind", "pubs", "tempdb") ', True);
              fnSQLOpen(Qry);

              while not Qry.Eof do
              begin
                EDBName.Items.Add(Qry.FieldByName('name').AsString);
                Qry.Next
              end;

              EDBName.Text := TempName;
            end;
          scFirebird:
            begin
              TempName := IfThen(EDBNameFB.Text = '', EDBNameFB.Text, DBName);
            end;
          scMySQL:
            begin
              TempName := IfThen(EMyDBName.Text = '', EMyDBName.Text, DBName);

              TryDB.GetTableNames(EMyDBName.Items);
            end;
        end;

        FreeAndNil(TryDB);
        FreeAndNil(Qry);
      end;
    else
      fnMessage('Error:'#13'Pastikan Nama Server, Name User dan Password sudah benar !',
mERROR);
    end;
```

```
    ConnectionType := TempConType;
  end;

  procedure TFDBServer.suitempCmbDBTypeChange(Sender: TObject);
  begin
    if CmbDBType.Text = 'Firebird' then
    begin
      GroupBox2.Show;
      GroupBox1.Hide;
      GrbMySQL.Hide;
    end
    else if CmbDBType.Text = 'Microsoft SQL Server' then
    begin
      GroupBox1.Show;
      GroupBox2.Hide;
      GrbMySQL.Hide;
    end
    else if CmbDBType.Text = 'MySQL' then
    begin
      GrbMySQL.Show;
      GroupBox1.Hide;
      GroupBox1.Show;
    end
  end;

  procedure TFDBServer.suitempCmbServerTypeChange(Sender: TObject);
  begin
    if CmbServerType.Text = 'Local' then
    begin
      suiButton1.Show;
      lblServerName.Enabled := False;
      EServerName.Enabled := False;
    end
    else
    begin
      suiButton1.Hide;
      lblServerName.Enabled := True;
      EServerName.Enabled := True;
    end
  end;

  procedure TFDBServer.suitempsuiButton1Click(Sender: TObject);
  begin
    if OD.Execute then
    begin
      EDBNameFB.Text := OD.FileName
    end
  end;

  end
```

UfunctLpas

```
unit UFunc;

interface

uses SysUtils, ADODB, Forms, Windows, Classes, StdCtrls, AdvGrid, StrUtils,
  DateUtils, Registry, Controls, ComCtrls, Menus, WinTypes, IniFiles, VaComm,
  RpDevice, RpDefine, Grids, RvClass, RVCsRpt, RvData, WinSpool, DBGrids,
  DBAdvGrid, DB, TypInfo, DBGridEh, dbgridEhImpExp, AdvGridExcel;

type
  TConnectionResult = (crSuccess, crServerNotFound, crDBNotFound);
  TInputType = (itNumeric, itDiscount, itString);
  TSQLConnection = (scSQLServer, scFirebird, scMySQL);
  TAccPostItem = (apDebet, apCredit);

  TBuiltInSkin = (WinXPGreen, Vista1, Vista2, RealOne,
    MediaPlayer10, Office2007, DiamondBlue, CarImness);

  TSkinnedForm = class(TForm)
    procedure RemoveSkin(Sender: TObject);
  end;

// koneksi database
function fnCheckDB(ConStr: string; DBName: string): TConnectionResult;
function fnConStr: string;
function fnGetConnectionString(HostName, Catalog, UserName, Password: string):
string;

// form control
function fnMessage(Text: string; MsgType: string): integer;
procedure prCreateForm(InstanceClass: TComponentClass; var Reference; ShowMode:
string = 'SHOW');

procedure fnResetMenu(Permission: string; Menu: TMainMenu);

// conversion function
function fnRound(total: Double; Default: Integer = 100): double;
function fnStrToBool(S: string; DefaultTrue: boolean = False): boolean;
function fnStrToCurr(Value: string): Currency;
function fnFormatCurr(const Value: Currency; const isInput: boolean = True): string;
overload;
function fnFormatCurr(const Value: string; const isInput: boolean = True): string;
overload;
function fnReplaceStr(strSource, strReplaceFrom, strReplaceWith: string;
  goTrim: boolean = true): string;
function fnInputDisc(Value: string): string;
function fnUpperFirstLetter(Value: string): string;
```

```
// input
function fnInput(Key: char; Text: string; MaxChar: integer; InputType: TInputType;
  DecimalAllowed: boolean = False; MinusAllowed: boolean = False): string;
procedure fnInputBox(Sender: Tobject; var Key: Char; MaxChar: Integer;
  InputType: TInputType; DecimalAllowed: Boolean = False; MinusAllowed: boolean =
False);

// Localize
procedure fnGetLocalInfo;

// ---------- encryption ----------
function fnBlowFishEncrypt(Src: string): string;
function fnBlowFishDecrypt(Src: string): string;
function fnRC2Encrypt(Str: string): string;
function fnRC2Decrypt(Str: string): string;

// DATABASE FUNCTION
function fnGenerateId: string;
function fnFindValue(TableName, FieldValue, FieldCondition, Condition: string): string;
procedure prExplodeStr(SourceStr: string; Delimiter: char; var List: TStringList);
function fnCheckTableStructure(TableList: TStringList; ArrayTable: array of string;
  TableName: string): boolean;
function fnCheckIndexStructure(IndexList: array of string; TableName: string;
  TableList: TStringList; IndexName: string): boolean;
procedure fnSQLAdd(Query: TADOQuery; SQL: string; ClearPrior: boolean = False);
overload;
procedure fnSQLAdd(Query: TADOCommand; SQL: string); overload;
procedure fnSQLAdd(Query: TADODataSet; SQL: string); overload;
procedure fnSQLOpen(Query: TADOQuery); overload;
procedure fnSQLOpen(Query: TADODataSet); overload;
procedure fnExecSQL(Query: TADOQuery); overload;
procedure fnExecSQL(Query: TADOCommand); overload;
procedure fnSQLParamByName(Query: TADOQuery; ParamStr: string; Value: Variant);
overload;
procedure fnSQLParamByName(Query: TADOCommand; ParamStr: string; Value:
Variant); overload;
procedure fnSQLParamByName(Query: TADODataSet; ParamStr: string; Value:
Variant); overload;
procedure fnCleanGrid(Stg: TAdvStringGrid; InitializeRow: integer = 2);
procedure fnCleanLog;
procedure fnStartTransaction;
procedure fnCommit;
procedure fnRollBack;
function fnInTransaction: boolean;
procedure fnRefreshDB(Qry: TADOQuery);
function fnGetServerDate: TDateTime;
function fnInputDateDB(Value: TDateTime; FromServer: boolean = True): string;
function fnTimeToStr(const S: TDateTime): string;
function fnStrToTime(const S: string): TDateTime;
```

```
// Error/Crash handle
procedure fnCheckNetwork(E: Exception; FormSender: TForm = nil);
procedure fnWriteCrashLog(ErrorType: string);

function fnDiv(a, b: currency): Double;
function fnTranslateFormula(StrFormula: string): string;
function fnGetTransNumber(TransType: string; var IsManual: boolean;
  TableName: string = 'transaction_history'; FieldType: string = 'th_type';
  FieldNumber: string = 'th_number'; FieldDate: string = 'th_date';
  FieldStatus: String = 'th_status'; Order: string = 'th_id DESC';
  IsDateManual: boolean = False; DateManual: TDateTime = 0): string;
function IncString(Source: string; LenStr: integer): string;
function fnGetAccNumber(TransType: string; var IsManual: boolean; Own: boolean =
False;
  ADate: TDateTime = 0): string;

// untuk keperluan input transaksi akuntansi
function fnAccTrans(TransType, TransDate, TransNumber, TransStatus: string;
  TransTotal: Double; TransNote, TransId: string; AGlId: string = ''; FromTable: string =
''): string;
procedure fnAccTransItem(GlId: string; ItemType: string; AccountId: string; ItemNote:
string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = ''); overload;
procedure fnAccTransItem(GlId: string; ItemType: TAccPostItem; AccountId: string;
ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = ''); overload;
procedure fnAccTransItemName(GlId: string; ItemType: string; AccountName: string;
ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = ''); overload;
procedure fnAccTransItemName(GlId: string; ItemType: TAccPostItem; AccountName:
string; ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = ''); overload;

// date time
function fnFormatDate(DateIn: TDate; FullMode: Boolean = false;
  TimeMode: Boolean = false): string;
function fnFormatDateTimeDB(const Value: TDateTime;
  const DisplayTime: boolean = True;
  const DisplayMilliSecond: boolean = True): string; overload;
function fnStartDay(const AValue: TDateTime): TDateTime;
function fnEndDay(const AValue: TDateTime): TDateTime;
function fnEndOfTheMonth(const AValue: TDateTime): TDateTime;
function fnMonthsBetween(const ANow, AThen: TDate): Integer;

function fnFormatDateIndo(const Value: TDateTime; ShowTime: boolean = True;
  ShowMs: boolean = False): string;

// registry
function fnWriteReg(RegName: string; RegType: string; Value: Variant): boolean;
//String,DateTime
function fnReadReg(RegName: string; RegType: string): variant; //String,DateTime

function fnReadIniFiles(Section, Identity, Default: string): string;
procedure fnWriteIniFiles(Section, Identity, Default: string);
procedure fnLoadWallpaper;
procedure fnChangeSkin;
procedure fnLoadSkin;
procedure fnLoadLocalSetting;
procedure fnLoadPrintSetting;
procedure fnLoadRegionalIni(ID: LCID);
procedure fnLoadRegionalSetting;

procedure fnLogoutOperator;

//untuk comm, Cust Display & Cash Drower
function fnCommOpen(VaComm: TVaComm; Port: byte; BaudRate: string;
  DataBits: string; Parity: string): boolean;
function fnCommClose(VaComm: TVaComm): boolean;
procedure fnCDispClean(objComm: TVaComm);
procedure fnCDispWrite(objComm: TVaComm; CDispType: string; xPos: integer;
  yPos: integer; str: string);
procedure fnCDrawOpen(VaComm: TVaComm; CDrawType: string);
procedure fnCDispRunningText(objComm: TVaComm; TextLine1: string; TextLine2:
string; textWidth: integer);

procedure fnLoadPrint(RptName: string; DvName: string = 'DvStg';
  Stg: TAdvStringGrid = nil; StartRow: integer = 0; EndRow: Integer = 0); overload;
procedure fnLoadPrint(RptName: string; DbGrid: TDBGrdEh); overload;
function fnGetPrinterProperties(propType: string): string;

function SQLConfigDataSource(HwndParent: HWND; FRequest: WORD; Driver: PChar;
  Attributes: Pchar): boolean; Stdcall;
function SQLGetInstalledDrivers(Size: string; Buff: WORD; BuffOut: WORD): boolean;
Stdcall;

function SaveToExcel(DBGrid: TDBGrid): boolean, overload;
function SaveToExcel(StGrid: TAdvStringGrid): boolean; overload;
function SaveToExcel(DBGrid: TDBGridEh): boolean; overload;

function FMod(x, y: int64): int64;
function fnTerbilang(x: int64): string; overload;
function fnTerbilang(x: currency): string; overload;

procedure fnDataSetFormat(DataSet: TDataset);

// fungsi untuk System
function FindWindowX(ACaption, AClass: string): THandle;
function IsCompressed: boolean;

function fnCheckTransManual(TransNum, TransType: string): boolean;
```

```
const
  mWARN = 'WARN';
  mINFO = 'INFO';
  mERROR = 'ERROR';
  mCONFIRM = 'CONFIRM';

  NoDataTo = 'Tidak ada data untuk ';
  NoDataToEdit = NoDataTo + 'diedit !';
  NoDataToPrint = NoDataTo + 'dicetak !';
  NoDataToDelete = NoDataTo + 'dihapus !';

  DataAlreadyUsed = 'Kode / Nama sudah dipakai.';
  DataSaved = 'Data sudah disimpan.';
  DataDeleted = 'Data sudah dihapus.';
  DataCannotDelete = 'Data tidak dapat dihapus karena sudah digunakan dalam
transaksi !';
  DataEdited = 'Data sudah diubah.';
  DataNotComplete = 'Data belum diisi lengkap !';
  DataCannotEdit = 'Data tidak dapat diubah karena sudah digunakan dalam transaksi !';

  DataPrinted = 'Data telah dicetak.';
  ErrorPrinting = 'Pencetakan gagal dilakukan.';

  SearchNotFound = 'Data tidak ditemukan.';
  CrashLog = 'DATE: %s | FORM: %s | ERROR: %s';

  // ODBC
  ODBC_ADD_DSN = 1;
  ODBC_CONFIG_DSN = 2;
  ODBC_REMOVE_DSN = 3;
  ODBC_ADD_SYS_DSN = 4;
  ODBC_CONFIG_SYS_DSN = 5;
  ODBC_REMOVE_SYS_DSN = 6;
  ODBC_REMOVE_DEFAULT_DSN = 7;

var
  ConStr: string;
  FirstEnter: boolean;
  DBName: string;
  vrDBName: string;
  vrDecimal: integer = 2;
  vrKeyLocation: string;
  vrAppShowBugs: boolean = True;
  vrThemeFile: string;
  vrRounded: boolean = True;
  vrOpenPrice: boolean;
  IsTerminated: boolean;
  CasName: string;
  vrShowMothName: boolean = True;
  vrCanChangePriceBuy: boolean = False;

  // ODBC Name
  vrDSNNameFB: string;
  vrDriverNameFB: string;
  vrDSNNameMy: string;
  vrDriverNameMy: string;

  OpId: string;
  OpName: string;
  OpLevel: string;
  IsMaster: boolean;
  OpFirstCash: currency; // set on operator first cash = set this if level is Cashier
  OpFirstDate: TDateTime;
  OpCashId: string;
  OpDateIn: TDateTime;
  OpUserName: string;

  OpAllowDiscTotal: boolean,
  OpAllowDiscItem: boolean;
  OpAllowTax: boolean;
  OpAllowPPN: Boolean = False;

  vrLogoutPrint: boolean;
  vrSaveColumnSize: boolean = False;
  IsPaste: boolean;
  IsCopy: boolean;

  vrSkinName: string = 'MacOS';
  vrSkinBuiltIn: boolean = True;

  vrPrintMarginTop: currency,
  vrPrintMarginLeft: currency,
  vrPrintDefaultDest: string,
  vrPrinterName: string;
  vrPrinterPaper: string;
  vrPrintPaperSize: currency = 0;
  vrPrintType: string,
  fnFooter: string;

  //Company
  CompType: string;
  CompName: string,
  CompAddr: string,
  CompTelp: string;
  CompNPWP: string,
  CompFax: string,
  CompCont: string;
```

```
  ConnectionType: TSQLConnection = scSQLServer; // default connection to SQL
Server!
  vrRgnFS: TFormatSettings;

  //CDisp
  vrCDispActive: boolean = false;
  vrCDispType: string;
  vrCDispPort: byte;
  vrCDispBaudRate: string;
  vrCDispDataBits: string;
  vrCDispParity: string;
  vrCDispTextLength: byte;
  vrCDispTimer: integer;
  vrCDispPos: integer;
  vrCDispDelay: integer = 20;
  vrCDispText1: string;
  vrCDispText2: string;

  //CashDraw
  vrCDrawActive: boolean = false;
  vrCDrawType: string;
  vrCDrawPort: byte;
  vrCDrawBaudRate: string;
  vrCDrawDataBits: string;
  vrCDrawParity: string;

  //Scaner
  vrBScanSource: string;
  vrBScanType: string;
  vrBScanPort: byte;
  vrBScanBaudRate: string;
  vrBScanDataBits: string;
  vrBScanParity: string;

implementation

uses UMain, UDM, DCPmd5, DCPrc2, DCPBlowFish, DCPsha512, USaveExcel,
  UReconnectDB, AdvPicture;

function SQLConfigDataSource; external 'odbccp32.dll' name 'SQLConfigDataSource';
function SQLGetInstalledDrivers; external 'odbccp32.dll' name 'SQLGetInstalledDrivers';

// Koneksi Database

function fnCheckDB(ConStr: string; DBName: string): TConnectionResult;
var ADOTest: TADOConnection;
  ADOCmd: TADOCommand;
begin
  Result := crSuccess;
  ADOTest := TADOConnection.Create(nil);
  ADOCmd := TADOCommand.Create(nil);
  try
    ADOTest.ConnectionTimeout := 10;
    ADOTest.LoginPrompt := False;
    ADOTest.ConnectionString := ConStr;
    ADOTest.Open;

    if (ConnectionType = scSQLServer) or (ConnectionType = scMySQL) then
    begin
      try
        if DBName <> '' then
        begin
          ADOCmd.Connection := ADOTest;
          ADOCmd.CommandType := cmdText;
          ADOCmd.CommandText := 'USE ' + DBName;
          ADOCmd.Execute
        end;
      except
        Result := crDBNotFound
      end
    end;
  except
    Result := crServerNotFound;
  end;
  FreeAndNil(ADOTest);
  FreeAndNil(ADOCmd)
end;

function fnConStr: string;
var
  HostName: string;
  UserName: string;
  Password: string;
  ConnType: string;
begin
  HostName := fnRC2Decrypt(fnReadReg('HostName', 'string'));
  UserName := fnRC2Decrypt(fnReadReg('UserName', 'string'));
  Password := fnRC2Decrypt(fnReadReg('Password', 'string'));
  DBName := fnRC2Decrypt(fnReadReg('DBName', 'string'));
  ConnType := fnRC2Decrypt(fnReadReg('Database', 'string'));

  if ConnType = '' then
    ConnType := 'Microsoft SQL Server';

  if ConnType = 'Microsoft SQL Server' then
    ConnectionType := scSQLServer
  else if ConnType = 'Firebird' then
  begin
    ConnectionType := scFirebird;

//    UserName := fnRC2Decrypt('6MAkAstu'); // SYSDBA
//    Password := fnRC2Decrypt('ybjfulhztg=='); // repentz
  end
  else if ConnType = 'MySQL' then
  begin
    ConnectionType := scMySQL;
  end;

  if (HostName = '') and (UserName = '') then
  begin
    case ConnectionType of
      scSQLServer:
        begin
          HostName := fnRC2Decrypt('k6F3wmw5AQ=='); // (local)
          UserName := fnRC2Decrypt('yNQ='); // sa
          Password := '';
          DBName := vrDBName;
        end;
      scFirebird:
        begin
          HostName := fnRC2Decrypt('19cZA89ieXTG'); // localhost
          UserName := fnRC2Decrypt('6MAkAstu'); // SYSDBA
          Password := fnRC2Decrypt('1pvCNdIC3dXk'); // masterkey
          DBName := fnRC2Decrypt('6aqFvF/1lR3Cuw=='); // RETAIL2_DB
        end;
      scMySQL:
        begin
          HostName := fnRC2Decrypt('19cZA89ieXTG'); // localhost
          UserName := fnRC2Decrypt('ybL3ZQ=='); // root
          Password := fnRC2Decrypt('ybjfulhztg=='); // repentz
          DBName := fnRC2Decrypt('6aqFvF/1lR3Cuw=='); // RETAIL2_DB
        end;
    end
  end;

  Result := fnGetConnectionString(HostName, DBName, UserName, Password);
end;

function fnGetConnectionString(HostName, Catalog, UserName, Password: string):
string;
var s: string;
begin
  case ConnectionType of
    scSQLServer:
      begin
        //Provider=SQLOLEDB.1;Persist Security Info=True;User ID=%s;
        //Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;
        //Use Encryption for Data=False;Tag with column collation when possible=False;
        //Password=%s;Initial Catalog=%s;Data Source=%s
        s := fnRC2Decrypt(
          '614SfNi08ZvnQgjqWax2O8wrrUsdPQ7fhG/q6FxjjeQ2NzaeYBp+kuMgh4mg' +
          'ZvE0Tojx8Joy+co3IigC8canLscDWqyicHHW563kL71yTc3JwAwUhsJPxZ0s' +
          'J0nBx4zffdc/VgbgNdAJzZNGk77HixvKp9PvdzHDJ+7jHyuFVc2AlhEsGz3N' +
          '00Am1JGnK4nIDTQ4DZh1Hf5yM1ovaqiEklt7fYzZTqx3b+ngonwB2foul4u7' +
          'UpE/dpRL2uH5DCzaJ6UZQPN1HUIF6IC5ilcxHnB3yvzE7blZgbw81RokweC3T'
+
          'SM4HHHAvhuDSSsUsv480+ONS5Q==');

        Result := Format(s,
          [UserName, Password, Catalog, HostName]);
      end;
    scFirebird:
      begin
        //Provider=MSDASQL.1;Password=%s;Persist Security Info=True; '
        //'User ID=%s;Extended
Properties="DSN=MBIntermediate;Driver={Firebird/InterBase(r) driver}; ' +
        //'Dbname=%s;CHARSET=NONE;FWD=%s;UID=%s;"';
        s := fnRC2Decrypt(
          '614SfNi08ZvnXBjgfeXBxvUqUxm1Jzz8MQA3szxq/tsBJ4egomBgHyih0fawr' +
          'Y2MuRZjB/2sKOx3zaU3TedkjLkjSgoVYlnc9hu/KCkmbX/oyT/wv/uQeM48m0'
+
          'bbNzP6ZQceuJt7kQZVYs1ZkJ19UEVh1PCn/hrxOwLGkZoqihjdSer8/BHnmx' +

'JFYG8fziWGL7kg/ZOAJ0NQjhKLfbpgPwoS2FCTciARNWJdAzFk9LqdPUJRjwBr5o
=');

        Result := Format(s, [Password, UserName, Catalog, Password, UserName]);
      end;
    scMySQL:
      begin
        //Provider=MSDASQL.1;Password=%s;Persist Security Info=True;User ID=%s;
        //Extended Properties="DESCRIPTION=Intermediate MySQL Connection;

//DSN=MBIntermediateM;OPTION=0;PWD=%s;PORT=3306;SERVER=%s;UID=%s"
        s := fnRC2Decrypt(
          '614SfNi08ZvnXBjgfeXBxvUqUxm1Jzz8MQA3szxq/tsBJ4egomBgHyih0fawr' +
          'Y2MuRZjB/2sKOx3zaU3TedkjLkjSgoVYlnc9hu/KCkmbX/oyT/wv/uQeM48m1'
+

'CDyHFRm9M1RG4H2Lpy4tDmqEelHMc/jTuK1nzwRYvH5XWybRy8pbpFuGdPr3N'
+

'KztJ7MKQxiTmwdulikI35d7Xm0ha81w8ml+Gb+PVAv8XmF2nh410rKPN1s3W' +
          'UM0KT8TAG9eEXAIK');

        Result := Format(s, [Password, UserName, Password, HostName, UserName]);
      end
    end;
  end;
```

```
// form control

function fnMessage(Text: string; MsgType: string): integer;
begin
  Result := 0;
  if MsgType = mINFO then
  begin
    Result := Application.MessageBox(PAnsiChar(Text), 'Informasi', MB_OK +
MB_ICONINFORMATION);
  end
  else if MsgType = mCONFIRM then
  begin
    Result := Application.MessageBox(PAnsiChar(Text), 'Konfirmasi', MB_YESNO +
MB_ICONQUESTION);
  end
  else if UpperCase(MsgType) = 'CONFIRM2' then
  begin
    Result := Application.MessageBox(PAnsiChar(Text), 'Konfirmasi', MB_OKCANCEL
+ MB_ICONQUESTION);
  end
  else if MsgType = mWARN then
  begin
    Result := Application.MessageBox(PAnsiChar(Text), 'Perhatian', MB_OK +
MB_ICONWARNING);
  end
  else if MsgType = mERROR then
  begin
    Result := Application.MessageBox(PAnsiChar(Text), 'Error', MB_OK +
MB_ICONERROR);
  end;
end;

procedure prCreateForm(InstanceClass: TComponentClass; var Reference; ShowMode:
string = 'SHOW');
begin
  if TComponentClass(Reference) = nil then
  begin
    if Assigned(FMain) and (ShowMode = 'SHOW') then
      LockWindowUpdate(FMain.Handle);
    Application.CreateForm(InstanceClass, Reference);

    if Assigned(FMain) then
    begin
      if FMain.suiSkinEngine1.Active then
      begin
        FMain.suiSkinEngine1.AddForm(TSkinnedForm(Reference));
        TSkinnedForm(Reference).OnDestroy := TSkinnedForm(Reference).RemoveSkin;
      end
    end;
    if (TSkinnedForm(Reference).FormStyle = fsMDIChild)
      and (UpperCase(TSkinnedForm(Reference).Name) <> 'FLOGIN') then
    begin
      TSkinnedForm(Reference).WindowState := wsMaximized;
    end;
    if Trim(UpperCase(ShowMode)) = 'SHOW' then
    begin
      TSkinnedForm(Reference).Show;
    end
    else if Trim(UpperCase(ShowMode)) = 'SHOWMODAL' then
    begin
      TSkinnedForm(Reference).ShowModal;
    end;
    if Assigned(FMain) and (ShowMode = 'SHOW') then
      LockWindowUpdate(0);
  end;
end;

procedure fnResetMenu(Permission: string; Menu: TMainMenu);
  function SetMenu(Menu: TMenuItem; Access: array of boolean; Count: integer): integer;
  var
    i: integer;
  begin
    for i := 0 to Menu.Count - 1 do
    begin
      if menu[i].Caption <> '-' then
      begin
        if High(Access) >= Count then
        begin
          Menu[i].Enabled := Access[Count];
          Menu[i].Visible := Access[Count];
        end
        else
        begin
          Menu[i].Enabled := False;
          Menu[i].Visible := False;
        end;
        Inc(Count);
        if Menu[i].Count > 0 then
        begin
          Count := SetMenu(Menu[i], Access, Count);
        end;
      end;
    end;
    SetMenu := Count;
  end;

var i: integer;
  MenuAccess: array of boolean;
begin
  SetLength(MenuAccess, Length(Permission));
  for i := 1 to Length(Permission) do
  begin
    if (Copy(Permission, i, 1) = '1') then
      MenuAccess[i - 1] := False
    else
      MenuAccess[i - 1] := True;
  end;
  SetMenu(Menu.Items, MenuAccess, 0);
  SetLength(MenuAccess, 0);
  if Assigned(FMain) then
    FMain.suiSkinEngine1.UpdateTopMenu;
end;

procedure prExplodeStr(SourceStr: string; Delimiter: char; var List: TStringList);
var
  i: integer;
begin
  List.Clear;
  while Length(SourceStr) > 0 do
  begin
    i := Pos(Delimiter, SourceStr);
    if (i > 0) then
    begin
      List.Add(Copy(SourceStr, 1, i - 1));
      SourceStr := Copy(SourceStr, i + 1, Length(SourceStr) - i);
    end // if (i > 0) then
    else if Length(SourceStr) > 0 then
    begin
      List.Add(SourceStr);
      SourceStr := '';
    end // if Length(SourceStr) > 0 then
  end; //while Length(SourceStr) > 0 do
end;

function fnCheckTableStructure(TableList: TStringList; ArrayTable: array of string;
  TableName: string): boolean;
  function fnConvertSQLServerToFirebird(S: string): string;
  begin
    if Pos('MONEY', UpperCase(S)) > 0 then
      Result := StringReplace(S, 'MONEY', 'DECIMAL(18,4)', [rfReplaceAll])
    else if Pos('DATETIME', UpperCase(S)) > 0 then
      Result := StringReplace(S, 'DATETIME', 'TIMESTAMP', [rfReplaceAll])
    else if Pos('INT', UpperCase(S)) > 0 then
      Result := StringReplace(S, 'INT', 'INTEGER', [rfReplaceAll])
    else if Pos('DEC (38,18)', UpperCase(S)) > 0 then
      Result := StringReplace(S, 'DEC (38,18)', 'DECIMAL', [rfReplaceAll])
    else if Pos('IMAGE', UpperCase(S)) > 0 then
      Result := StringReplace(S, 'IMAGE', 'BLOB', [rfReplaceAll])
    else
      Result := S
  end;

var
  QryTemp: TADOQuery;
  QryTemp1: TADOQuery;

  i: integer;
  FieldName: string;
  FieldSize: integer;
  FieldType: string;

  isPrimaryKey: boolean;
  TempPrimary: string;
  isTableExist: boolean;
  IsImage: boolean;

  FieldList: TStringList;
  isFieldExist: boolean;
  isEmptyTable: boolean;
  TempList: TStringList;

  ChangeSize: boolean;
  SpecialFieldSize: string;
begin
  FieldList := TStringList.Create;
  TempList := TStringList.Create;
  IsPrimaryKey := False;
  ChangeSize := False;

  Result := True;

  QryTemp := TADOQuery.Create(nil);
  QryTemp.Connection := DM.DB;
  QryTemp1 := TADOQuery.Create(nil);
  QryTemp1.Connection := DM.DB;

  try
    //------------------------------------------------------------------------
    // periksa apakah ada table
    isTableExist := False;
    if TableList.IndexOf(TableName) >= 0 then
      IsTableExist := True;

    if not IsTableExist then
    begin
      fnSQLAdd(QryTemp, 'CREATE TABLE ' + TableName + '(', True);

      for i := 0 to High(ArrayTable) do
        if ConnectionType = scSQLServer then
```

```
      fnSQLAdd(QryTemp, ArrayTable[i])
    else
      fnSQLAdd(QryTemp, fnConvertSQLServerToFirebird(ArrayTable[i]));

  fnSQLAdd(QryTemp, ')');

  fnExecSQL(QryTemp)
 end
 // ------------------------------------------------------------
 else // if table has already exist in database
 begin
  dm.DB.GetFieldNames(TableName, FieldList); // get list field in table

  // ------------------------------------------------------------
  for i := 0 to High(ArrayTable) do // last line not proceed because it is primary key
declaration
   begin
    fnSQLAdd(QryTemp1, Format('SELECT TOP 1 * FROM %s', [TableName]), True);
    fnSQLOpen(QryTemp1);

    IsEmptyTable := True;
    if not QryTemp1.IsEmpty then
     IsEmptyTable := False;

    if ConnectionType = scFirebird then
     ArrayTable[i] := fnConvertSQLServerToFirebird(ArrayTable[i]);

    if i = High(ArrayTable) then
    begin
     // check wether this line is primary key's line
     if Pos('PRIMARY KEY', ArrayTable[i]) <> 0 then
     begin
      IsPrimaryKey := True;
      TempPrimary := ArrayTable[i];
     end
     else
     begin
      IsPrimaryKey := False;
      TempPrimary := '';
     end
    end; // if i = High(ArrayTable) then

    if not IsPrimaryKey then
    begin
     // ------------------------------------------------------------
     // split string to its type
     TempList.Clear;
     prExplodeStr(ArrayTable[i], ' ', TempList);

     FieldName := TempList[0];
     FieldType := TempList[1];
     FieldSize := 0; // just try to convert it to numeric from TempList
     SpecialFieldSize := ''; // save the size of special field type, e.g. DECIMAL and
NUMERIC data type

     // check if field already exist
     IsFieldExist := False;
     if FieldList.IndexOf(FieldName) >= 0 then
      IsFieldExist := True;

     if Pos('IMAGE', UpperCase(ArrayTable[i])) > 0 then
      IsImage := True
     else
      IsImage := False;

     // remove branket to specify fieldsize
     TempList[2] := StringReplace(TempList[2], '(', '', [rfReplaceAll, rfIgnoreCase]);
     TempList[2] := StringReplace(TempList[2], ')', '', [rfReplaceAll, rfIgnoreCase]);
     TryStrToInt(TempList[2], FieldSize); // if convert succeed means here is field size
                                          // otherwise none of field size

     // cari jika ada tipe data tertentu yang berbeda / spesial (DECIMAL / NUMERIC)
     if (Pos(',', TempList[2]) > 0) and
      ((UpperCase(FieldType) = 'DEC') or (UpperCase(FieldType) = 'DECIMAL')
      or (UpperCase(FieldType) = 'NUMERIC')) then
      SpecialFieldSize := TempList[2];

     fnSQLAdd(QryTemp, 'ALTER TABLE ' + TableName, True);

     if isFieldExist then
     begin
      if ConnectionType = scMySQL then
       fnSQLAdd(QryTemp, 'CHANGE ' + FieldName + ' ' + FieldName + ' ' +
FieldType)
      else
       fnSQLAdd(QryTemp, 'ALTER COLUMN ' + FieldName + ' ' + FieldType);

      ChangeSize := False;
      if FieldSize > 0 then
      begin
       fnSQLAdd(QryTemp1, 'SELECT TOP 1 ' + FieldName + ' FROM ' +
TableName, True);
       fnSQLOpen(QryTemp1);

       if SpecialFieldSize <> '' then
        fnSQLAdd(QryTemp, Format('(%s)', [SpecialFieldSize]))
       else
       begin
        if QryTemp1.FieldByName(FieldName).Size <> FieldSize then ChangeSize :=
True;

        if QryTemp1.FieldByName(FieldName).Size > FieldSize then
        begin
         FieldSize := QryTemp1.FieldByName(FieldName).Size;
        end;

        fnSQLAdd(QryTemp, Format('(%d)', [FieldSize]));
       end;
      end;

      if IsEmptyTable then
       if Pos('NOT NULL', ArrayTable[i]) > 0 then
        fnSQLAdd(QryTemp, ' NOT NULL')
       else
        fnSQLAdd(QryTemp, ' NULL')
      else
       fnSQLAdd(QryTemp, ' NULL');
     end
     else // if isFieldExist then
     begin
      fnSQLAdd(QryTemp, Format('ADD %s %s ', [FieldName, FieldType]));

      if FieldSize <> 0 then
      begin
       if SpecialFieldSize <> '' then
        fnSQLAdd(QryTemp, Format('(%s)', [SpecialFieldSize]))
       else
        fnSQLAdd(QryTemp, Format('(%d)', [FieldSize]));
      end;

      fnSQLAdd(QryTemp1, Format('SELECT TOP 1 * FROM %s', [TableName]),
True);
      fnSQLOpen(QryTemp1);

      // New field must have null value except there is still empty table
      if QryTemp1.IsEmpty then
       if Pos('NOT NULL', ArrayTable[i]) > 0 then
        if Pos('DEFAULT', ArrayTable[i]) > 0 then // pasti isi yang paling belakang
adalah koma dan spasi
         fnSQLAdd(QryTemp, Copy(ArrayTable[i], Pos('DEFAULT', ArrayTable[i]),
Length(ArrayTable[i]) - Pos('DEFAULT', ArrayTable[i])))
        else
         fnSQLAdd(QryTemp, ' NOT NULL')
       else
        fnSQLAdd(QryTemp, ' NULL')
      else
       fnSQLAdd(QryTemp, ' NULL');

      fnSQLAdd(QryTemp, TempPrimary)
     end; // else if isFieldExist then

     if ((not isFieldExist) or (ChangeSize))
      or ((not isFieldExist) and (isImage)) then
      fnExecSQL(QryTemp);
    end; // if not IsPrimaryKey then

   end; // for i := 0 to High(ArrayTable) do
   // ------------------------------------------------------------
  end; // else if not isTableExist
 except
  Result := False;
 end;

 FreeAndNil(QryTemp);
 FreeAndNil(QryTemp1);
 FreeAndNil(FieldList);
 FreeAndNil(TempList);
end;

function fnCheckIndexStructure(IndexList: array of string; TableName: string;
 TableList: TStringList; IndexName: string): boolean;
var IndexExists: boolean;
 TableExists: boolean;
 i: integer;

 Qry1: TADOQuery;
begin
 Result := True;

 Qry1 := TADOQuery.Create(nil);
 Qry1.Connection := DM.DB;
 try
  TableExists := False;
  if TableList.IndexOf(TableName) >= 0 then
   TableExists := True;

  if TableExists then
  begin
   IndexExists := False;

   case ConnectionType of
    scSQLServer:
     begin
      fnSQLAdd(Qry1, 'SELECT name FROM sysindexes WHERE name = ' +
QuotedStr(IndexName), True);
      fnSQLOpen(Qry1);
     end;
    scFirebird:
     begin
```

```
          fnSQLAdd(Qry1, 'SELECT * FROM RDB$INDEX_SEGMENTS WHERE
RDB$INDEX_NAME = ' + QuotedStr(IndexName), True);
          fnSQLOpen(Qry1);
        end;
      scMySQL:
        begin
          fnSQLAdd(Qry1, 'SELECT * FROM information_schema.statistics ' +
            'WHERE INDEX_NAME = ' + QuotedStr(IndexName) +
            'AND INDEX_SCHEMA = ' + QuotedStr(DBName) +
            'AND TABLE_NAME = ' + QuotedStr(TableName), True);
          fnSQLOpen(Qry1);
        end;
    end;

    if not Qry1.IsEmpty then
      IndexExists := True;

    if IndexExists then
    begin
      case ConnectionType of
        scSQLServer:
          begin
            fnSQLAdd(Qry1, Format('DROP INDEX %s.%s', [TableName, IndexName]),
True);
            fnExecSQL(Qry1);
          end;
        scFirebird:
          begin
            fnSQLAdd(Qry1, Format('DROP INDEX %s', [IndexName]), True);
            fnExecSQL(Qry1);
          end;
        scMySQL:
          begin
            fnSQLAdd(Qry1, Format('DROP INDEX %s ON %s', [IndexName,
TableName]), True);
            fnExecSQL(Qry1);
          end;
      end;

      IndexExists := False;
    end;

    if not IndexExists then
    begin
      fnSQLAdd(Qry1, Format('CREATE INDEX %s ON %s (', [IndexName,
TableName]), True);

      for i := 0 to High(IndexList) do
      begin
        fnSQLAdd(Qry1, IndexList[i]);
        end;

        fnSQLAdd(Qry1, ')');
        fnExecSQL(Qry1);
      end
    end
    else
    begin
      Result := False
    end
  except
    Result := False;
  end;
  FreeAndNil(Qry1);
end;

function fnGetServerDate: TDateTime;
var
  Qry: TADOQuery;
  s: string;
begin
  Qry := TADOQuery.Create(nil);
  Qry.Connection := DM.DB;
  result := 0;
  try
    case ConnectionType of
      scSQLServer: s := 'SELECT GETDATE() AS DateNow';
      scFirebird: s := 'SELECT current_timestamp(3) AS DateNow FROM
RDB$DATABASE';
      scMySQL: s := 'SELECT Now() AS DateNow';
    end;

    Qry.SQL.Clear;
    Qry.SQL.Add(s);
    Qry.Open;

    if ConnectionType = scMySQL then
    begin
      // MySQL doesn't support milliseconds for Now() function
      // so, create milliseconds from local computer
      result := IncMilliSecond(Qry.FieldByName('DateNow').AsDateTime,
MilliSecondOf(Now));
    end
    else
      result := Qry.FieldByName('DateNow').AsDateTime;
  except
    on E: Exception do fnCheckNetwork(E);
  end;
  FreeAndNil(Qry);
end;
```

```
function fnGenerateId: string;
begin
  Sleep(55);
  result := FormatDateTime('YYYYMMDDHHNNSSZZZ', fnGetServerDate);
end;

function fnFindValue(TableName, FieldValue, FieldCondition, Condition: string): string;
var
  qry: TADOQuery;
begin
  qry := TADOQuery.Create(nil);
  qry.Connection := DM.DB;

  qry.SQL.Clear;
  fnSQLAdd(qry,
    'SELECT ' + FieldValue + ' FROM ' + TableName + ' ' +
    'WHERE ' + FieldCondition + ' = :paramCondition ', True),

  fnSQLParamByName(qry, 'paramCondition', Condition),
  fnSQLOpen(qry);

  Result := qry.FieldByName(FieldValue).AsString;
  FreeAndNil(qry);
end;

function fnStrToCurr(Value: string): Currency;
var TempCurr: Currency;
begin
  tempCurr := 0;
  Value := StringReplace(Value, thousandSeparator, '', [rfReplaceAll, rfIgnoreCase]);
  Value := StringReplace(Value, '%', '', [rfReplaceAll, rfIgnoreCase]);

  TryStrToCurr(Value, TempCurr);

  Result := TempCurr;
end;

function fnRound(total: Double; Default: Integer = 100): double;
var
  rnd: integer;
  SisaBlt: Integer;
  SisaKma: Double;
begin
  rnd := Default; //round-nya
  SisaBlt := trunc(total) mod rnd;
  SisaKma := total - trunc(total);
  if SisaBlt + SisaKma > 0 then
    fnRound := Default - (SisaBlt + SisaKma)
  else
    fnRound := 0;
end;

function fnStrToBool(S: string; DefaultTrue: boolean = False): boolean;
begin
  if (S = '') then
    if (DefaultTrue) then
      S := 'True'
    else
      S := 'False';

  Result := StrToBool(S)
end;

function fnFormatCurr(const Value: Currency; const isInput: boolean = True): string;
var s: string;
begin
  if isInput then
  begin
//    Result := CurrToStrF(Value, ffNumber, 2)
    s := '###,###,###,###,##0.##';
    Result := FormatCurr(s, Value);
  end
  else
  begin
    Result := CurrToStrF(Value, ffCurrency, 2);
{   case CurrencyFormat of
    0: s := CurrencyString + '###,###,###,###,##0.00';
    1: s := '###,###,###,###,##0.00' + CurrencyString;
    2: s := CurrencyString + ' ###,###,###,###,##0.00';
    3: s := '###,###,###,###,##0.00 ' + CurrencyString;
    end;}
  end;
end;

function fnFormatCurr(const Value: string; const isInput: boolean = True): string;
var s: string;
  val: currency;
  Cvt: string;
begin
  Cvt := StringReplace(Value, ThousandSeparator, '', [rfReplaceAll, rfIgnoreCase]);
  if not TryStrToCurr(Cvt, val) then
  begin
//    MessageDlg(EConvertString, mtError, [mbOK], 0);
    result := Value;
  end else
  begin
    if isInput then
      s := '###,###,###,###,##0.##'
    else
```

```
    s := '###,###,###,###,##0.00';
   result := FormatCurr(s, Val);
  end;
end;

function fnReplaceStr(strSource, strReplaceFrom, strReplaceWith: string; goTrim:
boolean = true): string;
begin
 if goTrim then strSource := Trim(strSource);
 Result := StringReplace(strSource, StrReplaceFrom, StrReplaceWith, [rfReplaceAll,
rfIgnoreCase])
end;

function fnInputDisc(Value: string): string;
begin
 if Pos('%', Value) > 0 then
  Result := '%'
 else
  Result := 'A';
end;

function fnUpperFirstLetter(Value: string): string;
var s: string;
begin
 if Value <> '' then
 begin
  s := UpperCase(Copy(Value, 1, 1));
  if Length(Value) > 1 then
   Result := s + Copy(Value, 2, Length(Value))
  else
   Result := s
 end
end;

function fnInput(Key: char; Text: string; MaxChar: integer; InputType: TInputType;
 DecimalAllowed: boolean = False; MinusAllowed: boolean = False): string;
var
 CanAdd: boolean;
begin
 CanAdd := False;

 if (InputType = itNumeric) or (InputType = itDiscount) then
 begin
  if (MinusAllowed) and (Key = '-') and (FirstEnter) then
  begin
   CanAdd := True;
  end;

  Text := StringReplace(Text, ThousandSeparator, '', [rfReplaceAll]);

  if Key = '.' then Key := DecimalSeparator;
  if not (DecimalAllowed) and (Key = DecimalSeparator) then Key := #0;

  if InputType = itDiscount then
  begin
   if (Length(Text) > 0) and (not FirstEnter) then
   begin
    if Key = '%' then
    begin
     if Pos('%', Text) > 0 then
      CanAdd := False
     else
      CanAdd := True
    end
   end
  end; // if InputType = 'DISCOUNT' then

  if Key in ['0'..'9', DecimalSeparator] then
  begin
   if Pos('%', Text) > 0 then
    if FirstEnter then
     CanAdd := True
    else
     CanAdd := False
   else CanAdd := True;

   if (Key = DecimalSeparator) and (Pos(',', Text) > 0) then
   begin
    CanAdd := False;
   end;

   if Pos(',', Text) > 0 then
   begin
    if (Length(Text) - Pos(',', Text) > 1) then
     CanAdd := False;
   end;

   if Length(Text + Key) > MaxChar then
    CanAdd := False;
  end
 end // if (InputType = 'NUMERIC') or (InputType = 'DISCOUNT') then
 else
 begin
  CanAdd := True;

  if Length(Text + Key) > MaxChar then
   CanAdd := False;
 end;

 case Ord(Key) of

  VK_RETURN:
   begin
    FirstEnter := True;
    CanAdd := False;
   end;
  VK_BACK:
   begin
    Text := Copy(Text, 1, Length(Text) - 1);
    if (InputType = itNumeric) or (InputType = itDiscount) then
     if Text = '' then
     begin
      Text := '0';
      FirstEnter := True
     end;
    CanAdd := False;
   end
 end;

 if CanAdd then
 begin
  if FirstEnter then
  begin
   if Key = DecimalSeparator then
    Result := Text + Key
   else
    Result := Key;

   FirstEnter := False;
  end
  else
  begin
   if InputType in [itDiscount, itNumeric] then
   begin
    if (Key <> DecimalSeparator) and (Text = '0') then
    begin
     Result := Key;
    end
    else
     if (InputType = itDiscount) and (Key = '%') then
     begin
      if fnStrToCurr(Text) > 100 then Text := '100'
     end;

    if RightStr(Text, 1) = ',' then
    begin
     Text := fnFormatCurr(Text + Key);
     if Key = '0' then
      Text := Text + ',' + Key
    end
    else
     Text := fnFormatCurr(Text + Key);

    if Key = DecimalSeparator then
     Result := Text + Key
    else
     Result := Text;
   end
   else  // inputType = itString
   begin
    Result := Text + Key
   end
  end
 end
 else
 begin
  if (InputType = itNumeric) or (InputType = itDiscount) then
   Result := fnFormatCurr(Text)
  else
   Result := Text
 end
end;

procedure fnInputBox(Sender: TObject; var Key: Char; MaxChar: Integer;
 InputType: TInputType; DecimalAllowed: Boolean = False; MinusAllowed: boolean =
False);
begin
 with Sender as TEdit do
 begin
  if SelLength = Length(Text) then FirstEnter := True;
  Text := fnInput(Key, Text, MaxChar, InputType, DecimalAllowed, MinusAllowed);
  Key := #0;
  SelStart := Length(Text);
 end;
end;

procedure fnGetLocalInfo;
begin
 ShortDateFormat := 'dd/mm/yyyy';
 LongDateFormat := 'dddd/mmmm/yyyy';
 ThousandSeparator := '.';
 DecimalSeparator := ',';
 CurrencyString := 'Rp'; // localize to indonesian
 DateSeparator := '/';
 LongTimeFormat := 'hh:nn:ss.zzz';
 ShortTimeFormat := 'hh:nn:ss.zzz';
end;

// ---------------------------------------------------
// ---------------- Encryption -----------------------
// ---------------------------------------------------
```

```
function fnBlowFishEncrypt(Src: string): string;
var
  Cipher: TDCP_blowfish;
  KeyStr: string;
begin
  KeyStr := 'glPopMatrix';
  Cipher := TDCP_blowfish.Create(nil);
  Cipher.InitStr(KeyStr, TDCP_md5);
  Result := Cipher.EncryptString(Src);
  Cipher.Burn;
  Cipher.Free;
end;

function fnBlowFishDecrypt(Src: string): string;
var
  Cipher: TDCP_blowfish;
  KeyStr: string;
begin
  KeyStr := 'glPopMatrix';
  Cipher := TDCP_blowfish.Create(nil);
  Cipher.InitStr(KeyStr, TDCP_md5);
  Result := Cipher.DecryptString(Src);
  Cipher.Burn;
  Cipher.Free;
end;

function fnRC2Encrypt(Str: string): string;
var Cipher: TDCP_rc2;
  KeyStr: string;
begin
  KeyStr := 'glPushMatrix';
  Cipher := TDCP_rc2.Create(nil);
  Cipher.InitStr(KeyStr, TDCP_sha512); // initialize the cipher with a hash of the
passphrase
  Result := Cipher.EncryptString(Str);
  Cipher.Burn;
  Cipher.Free;
end;

function fnRC2Decrypt(Str: string): string;
var Cipher: TDCP_rc2;
  KeyStr: string;
begin
  KeyStr := 'glPushMatrix';
  Cipher := TDCP_rc2.Create(nil);
  Cipher.InitStr(KeyStr, TDCP_sha512); // initialize the cipher with a hash of the
passphrase
  Result := Cipher.DecryptString(Str);
  Cipher.Burn;
  Cipher.Free;
end;
// ---------- END OF ENCRYPTION ----------

// ---------- DATABASE ----------

procedure fnSQLAdd(Query: TADOQuery; SQL: string; ClearPrior: boolean = False);
var s: string;
begin
  if ClearPrior then
    Query.SQL.Clear;

  case ConnectionType of
    scFirebird:
      begin
        s := StringReplace(SQL, 'TOP', 'FIRST', [rfReplaceAll]);
      end;
    scMySQL:
      begin
        s := fnReplaceStr(SQL, 'GETDATE', 'CURRENT_DATE');
        s := fnReplaceStr(s, 'ISNULL', 'IFNULL');
      end;
    scSQLServer: s := SQL;
  end;

  Query.SQL.Add(S);
end;

procedure fnSQLAdd(Query: TADOCommand; SQL: string);
var s: string;
begin
  Query.CommandType := cmdText;

  case ConnectionType of
    scFirebird:
      begin
        s := StringReplace(SQL, 'TOP', 'FIRST', [rfReplaceAll]);
      end;
    scMySQL:
      begin
        s := fnReplaceStr(SQL, 'GETDATE', 'CURRENT_DATE');
        s := fnReplaceStr(s, 'ISNULL', 'IFNULL');
      end;
    scSQLServer: s := SQL;
  end;

  Query.CommandText := s;
end;

procedure fnSQLAdd(Query: TADODataSet; SQL: string);
var s: string;
begin
  Query.CommandType := cmdText;

  case ConnectionType of
    scFirebird:
      begin
        s := StringReplace(SQL, 'TOP', 'FIRST', [rfReplaceAll]);
      end;
    scMySQL:
      begin
        s := fnReplaceStr(SQL, 'GETDATE', 'CURRENT_DATE');
        s := fnReplaceStr(s, 'ISNULL', 'IFNULL');
      end;
    scSQLServer: s := SQL;
  end;

  Query.Close;  // tutup dahulu DataSet yang lama
  Query.CommandText := s;
end;

procedure fnSQLOpen(Query: TADOQuery);
var L: TStringList;
  s: string;
  s1: string;
  TempS: string;
  x1: integer;
  x2: integer;
begin
  if ConnectionType = scMySQL then
  begin
    L := TStringList.Create;

    s := Query.SQL.Text;
    x1 := Pos('SELECT TOP', UpperCase(s));
    if x1 > 0 then
    begin
      if s[x1 + 1] = '(' then // ==> berarti ada Sub Query di dalam Query, perlu diparse lagi
      begin
        x2 := Pos(')', s); // ambil akhir dari sub query
        s1 := UpperCase(Copy(s, x1, x2 - x1));
        prExplodeStr(s1, ' ', L);

        TempS := L[1] + ' ' + L[2];

        Insert(' LIMIT ' + L[2], s, x2);
        s := fnReplaceStr(s, TempS, '');
        Query.SQL.Text := s;
      end
      else
      begin
        // ambil jumlahnya
        prExplodeStr(UpperCase(s), ' ', L);

        TempS := L[1] + ' ' + L[2];

        s := fnReplaceStr(s, TempS, '');
        s := s + ' LIMIT ' + L[2];
        Query.SQL.Text := s;
      end
    end;

    FreeAndNil(L);
  end;

  Query.Prepared;
  Query.Open;
end;

procedure fnSQLOpen(Query: TADODataSet);
var L: TStringList;
  s: string;
  s1: string;
  TempS: string;
  x1: integer;
  x2: integer;
begin
  if ConnectionType = scMySQL then
  begin
    L := TStringList.Create;

    s := Query.CommandText;
    x1 := Pos('SELECT TOP', UpperCase(s));
    if x1 > 0 then
    begin
      if s[x1 + 1] = '(' then // ==> berarti ada Sub Query di dalam Query, perlu diparse lagi
      begin
        x2 := Pos(')', s); // ambil akhir dari sub query
        s1 := UpperCase(Copy(s, x1, x2 - x1));
        prExplodeStr(s1, ' ', L);

        TempS := L[1] + ' ' + L[2];

        Insert(' LIMIT ' + L[2], s, x2);
        s := fnReplaceStr(s, TempS, '');
        Query.CommandText := s;
      end
      else
      begin
        // ambil jumlahnya
```

```
      prExplodeStr(UpperCase(s), ' ', L);

      TempS := L[1] + ' ' + L[2];

      s := fnReplaceStr(s, TempS, '');
      s := s + ' LIMIT ' + L[2];
      Query.CommandText := s;
    end
  end;

  FreeAndNil(L);
 end;

 Query.Prepared;
 Query.Open;
end;

procedure fnExecSQL(Query: TADOQuery);
begin
 Query.Prepared;
 Query.ExecSQL;
end;

procedure fnExecSQL(Query: TADOCommand);
begin
 Query.Prepared;
 Query.Execute;
end;

procedure fnSQLParamByName(Query: TADOQuery; ParamStr: string; Value: Variant);
begin
 Query.Parameters.ParamByName(ParamStr).Value := Value
end;

procedure fnSQLParamByName(Query: TADOCommand; ParamStr: string; Value:
Variant);
begin
 Query.Parameters.ParamByName(ParamStr).Value := Value
end;

procedure fnSQLParamByName(Query: TADODataSet; ParamStr: string; Value:
Variant);
begin
 Query.Parameters.ParamByName(ParamStr).Value := Value
end;

procedure fnStartTransaction;
begin
 dm.DB.BeginTrans;
end;
```

```
procedure fnCommit;
begin
 dm.DB.CommitTrans;
end;

procedure fnRollBack;
begin
 dm.DB.RollbackTrans
end;

function fnInTransaction: boolean;
begin
 Result := dm.DB.InTransaction
end;

procedure fnRefreshDB(Qry: TADOQuery);
begin
 if Qry.Active then Qry.Close;
 Qry.Open;
end;

function fnInputDateDB(Value: TDateTime; FromServer: boolean = True): string;
begin
 if FromServer then
   Result := QuotedStr(fnFormatDateTimeDB(fnGetServerDate))
 else
   Result := QuotedStr(fnFormatDateTimeDB(Value))
end;

function fnTimeToStr(const S: TDateTime): string;
var St: string;
 Local: TFormatSettings;
begin
 GetLocaleFormatSettings(1033, Local);  // pake US English

 St := FormatDateTime('hh:nn:ss', S, Local);
 Result := St;
end;

function fnStrToTime(const S: string): TDateTime;
var St: string;
begin
 St := fnReplaceStr(S, ':', TimeSeparator);
 Result := StrToTime(St);
end;


procedure fnClearGrid(Stg: TAdvStringGrid; InitializeRow: integer = 2);
var i, j: integer;
```

```
begin
 for i := 1 to Stg.RowCount do
   for j := 0 to Stg.ColCount - 1 do
   begin
     Stg.Cells[j, i] := '';
     Stg.RemoveCheckBox(j, i);
   end;

 Stg.RowCount := InitializeRow;
end;

procedure fnCleanLog;
var Qry: TADOQuery;
begin
 Qry := TADOQuery.Create(nil);
 Qry.Connection := dm.DB;
 Qry.CommandTimeOut := 0;
 //---------- khusus untuk mssql server ----------
 //---------- untuk membersihkan transaction log ----------
 try
   if dm.DB.Connected and (ConnectionType = scSQLServer) then
   begin
     Qry.SQL.Clear;
     Qry.SQL.Add('BACKUP LOG ' + DBName + ' WITH TRUNCATE_ONLY');
     Qry.ExecSQL;

     Qry.SQL.Clear;
     Qry.SQL.Add('DBCC SHRINKDATABASE (' + DBName + ', 0)');
     Qry.ExecSQL;
   end;
 except
 end;
 FreeAndNil(Qry)
end;
// ---------- DATABASE ----------

procedure fnCheckNetwork(E: Exception; FormSender: TForm = nil);
var FormName: string;
 ErrorType: integer;
 Qry: TADOQuery;
 PrgBar: TProgressBar;
begin
 Qry := TADOQuery.Create(nil);
 Qry.Connection := dm.DB;
 PrgBar := TProgressBar.Create(nil);

 FormName := '';
 if FormSender <> nil then
   FormName := FormSender.Name
 else
```

```
   FormName := '-';

 // periksa apakah yang error adalah jaringan atau database
 try
   fnSQLAdd(Qry, 'SELECT * FROM operator', True);
   fnSQLOpen(Qry);

   ErrorType := 1; // menandakan kesalahan bukan di jaringan tetapi di program
 except
   ErrorType := 2; // kesalahan ada di jaringan
 end;

 if ErrorType = 1 then // kesalahan di program / struktur database
 begin
   try
     if dm.DB.InTransaction then dm.DB.RollbackTrans;

     dm.fnCheckDBStructure(PrgBar); ;
   except
   end;

   fnWriteCrashLog(Format('CrashLog, [fnFormatDateTimeDB(Now), FormName,
E.Message]));
   if vrAppShowBugs then
     fnMessage('Bug: ' + E.Message, mERROR);
 end
 else // kesalahan di jaringan
 begin
   // belum ada pengendalian error untuk jaringan
   // buat form reconnect !!
   prCreateForm(TFReconnectDB, FReconnectDB, 'ShowModal');
   fnWriteCrashLog('Network Error');
   fnMessage('Koneksi jaringan terputus.', mERROR);
 end;

 FreeAndNil(Qry);
 FreeAndNil(PrgBar);
end;


procedure fnWriteCrashLog(ErrorType: string);
var FileName: string;
 CrashFile: TextFile;
begin
 FileName := ExtractFilePath(Application.ExeName) + 'Crash.Log';

 try
   AssignFile(CrashFile, FileName);
   FileMode := fmOpenReadWrite;
```

```
    if FileExists(FileName) then
      Append(CrashFile)
    else
      Rewrite(CrashFile);

    WriteLn(CrashFile, ErrorType);
  finally
    CloseFile(CrashFile);
  end;
end;

function fnDiv(a, b: currency): Double; // Double data type to save decimal result
begin                                   // Currency Result only have 4 digit decimal
  if b <> 0 then
  begin
    Result := a / b;
  end
  else
  begin
    Result := 0;
  end;
end;

function fnGetFormulaDigit(StrFormula: string): integer;
begin
  try
    if Copy(StrFormula, Pos(',', StrFormula) - 1, Length(StrFormula)) <> '' then // kalo ada
definisi
      Result := StrToInt(Copy(StrFormula, Pos(',', StrFormula) + 1, Length(StrFormula)))
    else // kalo tidak ada definisinya, buat default 6 digit
      Result := 6;
  except
    Result := 0
  end
end;

function fnGetFormulaSeparator(StrFormula: string): char;
var x: integer;
begin
  x := Pos(',', StrFormula);
  if x <> 0 then
    Result := Copy(StrFormula, x + 1, 1)[1]
  else
    Result := Copy(StrFormula, Length(StrFormula), 1)[1];
end;

function fnTranslateFormula(StrFormula: string): string;
var TransFormula: string;
  ServerDate: TDateTime;
begin
  try
    ServerDate  = fnGetServerDate;
    TransFormula := StrFormula;
    TransFormula := StringReplace(TransFormula, '[NO]', CasName, [rfReplaceAll]);

    { ---------------------------------------------
    // for future only
    TransFormula := StringReplace(TransFormula, '[THN]', FormatDateTime('YYYY',
ServerDate), [rfReplaceAll]);
    TransFormula := StringReplace(TransFormula, '[BLN]', FormatDateTime('MM',
ServerDate), [rfReplaceAll]);
    TransFormula := StringReplace(TransFormula, '[TGL]', FormatDateTime('DD',
ServerDate), [rfReplaceAll]);
    --------------------------------------------- }

    TransFormula := fnReplaceStr(TransFormula, '[THN][BLN][TGL]',
FormatDateTime('YYYYMMDD', ServerDate));
    if Pos(',', TransFormula) > 0 then
      TransFormula := Copy(TransFormula, 1, Pos(',', TransFormula) - 1);

    Result := TransFormula;
  except
    Result := 'ERROR';
  end;
end;

function fnGetTransNumber(TransType: string; var IsManual: boolean;
  TableName: string = 'transaction_history'; FieldType: string = 'th_type';
  FieldNumber: string = 'th_number'; FieldDate: string = 'th_date';
  FieldStatus: String = 'th_status'; Order: string = 'th_id DESC';
  IsDateManual: boolean = False; DateManual: TDateTime = 0): string;
var
  TransNo: LongInt;
  NumOk: boolean;
  TempTransNum: string,
  TransDate: TDateTime;

  Qry: TADOQuery;
  Qry2: TADOQuery;

  TransFormula: string;
  Digit: integer;
begin
  qry := TADOQuery.Create(nil);
  qry.Connection := DM.DB;
  qry2 := TADOQuery.Create(nil);
  qry2.Connection := DM.DB;

  if IsDateManual then
    TransDate := DateManual
  else
    TransDate := fnGetServerDate;

  fnSQLAdd(Qry, 'SELECT * FROM my_index WHERE mi_name = ''Manual:' +
TransType + ''' ', True);
  fnSQLOpen(Qry);

  if Qry.FieldByName('mi_value').AsString <> 'Y' then
  begin
    IsManual := False;

    // ambil rumus nomor transaksi
    fnSQLAdd(Qry, 'SELECT * FROM my_index WHERE mi_name = ''TransNumber:' +
TransType  + ''' ', True);
    fnSQLOpen(Qry);

    TransFormula := fnTranslateFormula(Qry.FieldByName('mi_value').AsString);

    Digit := fnGetFormulaDigit(Qry.FieldByName('mi_value').AsString);

    fnSQLAdd(Qry, Format('SELECT TOP 100 %s FROM %s ' +
      'WHERE %s = ''%s'' ' +
      'AND (%s >= ' + fnInputDateDB(fnStartDay(TransDate), False) +
      ' AND %s <= ' + fnInputDateDB(fnEndDay(TransDate), False) +
      ') AND %s LIKE ''%%/%s/%%'' AND %s IN (''OK'', ''FINISH'', ''PENDING'', ''1'')
ORDER BY %s',
      [FieldNumber, TableName, FieldType, TransType, FieldDate, FieldDate,
FieldNumber,
      CasName, FieldStatus, Order]), True);
    fnSQLOpen(Qry);

    NumOK := False;
    TransNo := 0;
    while (not Qry.Eof) and (not NumOK) do
    begin
      TempTransNum := fnReplaceStr(Qry.FieldByName(FieldNumber).AsString,
Copy(Qry.FieldByName(FieldNumber).AsString, 1,
        Pos(fnGetFormulaSeparator(TransFormula),
Qry.FieldByName(FieldNumber).AsString)), '');

      if TryStrToInt(TempTransNum, TransNo) then
      begin
        while not NumOK do
        begin
          TransNo := TransNo + 1;
          TempTransNum := TransFormula + Format('%.' + IntToStr(Digit) + 'd', [TransNo]);

          fnSQLAdd(Qry2, Format('SELECT TOP 1 * FROM %s ' +
            'WHERE %s = ''%s'' AND %s = ''%s'' AND %s IN (''OK'', ''FINISH'',
''PENDING'', ''1'') ',
            [TableName, FieldNumber, TempTransNum, FieldType, TransType, FieldStatus]),
True);
          fnSQLOpen(Qry2);

          if Qry2.IsEmpty then
            NumOk := True;
        end;
      end;
      Qry.Next
    end; // while not Qry.Eof do

    if TransNo = 0 then TransNo := 1;
    TempTransNum := TransFormula + Format('%.' + IntToStr(Digit) + 'd', [TransNo]);

    Result := TempTransNum;
  end
  else
  begin
    IsManual := True;
    Result := '';
  end;

  FreeAndNil(qry);
  FreeAndNil(qry2);
end;

function fnFormatDateIndo(const Value: TDateTime; ShowTime: boolean = True;
  ShowMs: boolean = False): string;
var s: string;
//  m: string;
//  d: word;
//  MM: word;
  L: TStringList;
  i: integer;
//  Len: integer;
begin
//  d := DayOf(Value);
//  MM := MonthOf(Value);

  L := TStringList.Create;
  prExplodeStr(ShortDateFormat, DateSeparator, L);

  for i := 0 to L.Count - 1 do
  begin
    if Copy(UpperCase(L[i]), 1, 1) = 'M' then
    begin
      s := fnReplaceStr(ShortDateFormat, L[i], 'MMM');
      Break;
    end
  end;
```

```
 if ShowTime then
   s := s + ' ' + LongTimeFormat;

 s := FormatDateTime(s, Value);
{  if vrShowMothName then
  begin
    case MM of
      1: m := 'Jan';
      2: m := 'Feb';
      3: m := 'Mar';
      4: m := 'Apr';
      5: m := 'Mei';
      6: m := 'Jun';
      7: m := 'Jul';
      8: m := 'Ags';
      9: m := 'Sep';
      10: m := 'Okt';
      11: m := 'Nov';
      12: m := 'Des';
    end;
  end
  else
  begin
    m := IntToStr(MM)
  end;

  if ShowTime then
  begin
    if not ShowMs then
      s := Format('%.2d-%s-%d %s', [d, m, YearOf(Value), FormatDateTime('hh:nn:ss',
TimeOf(Value))])
    else
      s := Format('%.2d-%s-%d %s', [d, m, YearOf(Value), TimeToStr(TimeOf(Value))])
  end
  else
    s := Format('%.2d-%s-%d', [d, m, YearOf(Value)]);
}
  Result := s;
end;

function fnFormatDate(DateIn: TDate; FullMode: Boolean = false; TimeMode: Boolean
= false): string;
var
  Year: string;
  Month: string;
  Day: string;
  Hour: string;
  Minute: string;
  Second: string;
begin
  Year := IntToStr(YearOf(DateIn));
  if FullMode = false then
  begin
    case MonthOf(DateIn) of
      1: Month := 'Jan';
      2: Month := 'Feb';
      3: Month := 'Mar';
      4: Month := 'Apr';
      5: Month := 'Mei';
      6: Month := 'Jun';
      7: Month := 'Jul';
      8: Month := 'Agt';
      9: Month := 'Sep';
      10: Month := 'Okt';
      11: Month := 'Nop';
      12: Month := 'Des';
    end;
  end
  else
  begin
    case MonthOf(DateIn) of
      1: Month := 'Januari';
      2: Month := 'Februari';
      3: Month := 'Maret';
      4: Month := 'April';
      5: Month := 'Mei';
      6: Month := 'Juni';
      7: Month := 'Juli';
      8: Month := 'Agustus';
      9: Month := 'September';
      10: Month := 'Oktober';
      11: Month := 'Nopember';
      12: Month := 'Desember';
    end;
  end;
  Day := IntToStr(DayOf(DateIn));
  if TimeMode = true then
  begin
    Hour := IntToStr(HourOf(DateIn));
    if length(Hour) < 2 then Hour := '0' + Hour;
    Minute := IntToStr(MinuteOf(DateIn));
    if length(Minute) < 2 then Minute := '0' + Minute;
    Second := IntToStr(SecondOf(DateIn));
    if length(Second) < 2 then Second := '0' + Second;
    Year := Year + ' ' + Hour + ':' + Minute + ':' + Second;
  end;
  if Length(Day) < 2 then Day := '0' + Day;
  fnFormatDate := Day + '-' + Month + '-' + Year;
end;

function fnFormatDateTimeDB(const Value: TDateTime;
  const DisplayTime: boolean = True;
  const DisplayMilliSecond: boolean = True): string;
var s: string;
  Local: TFormatSettings;
begin
  GetLocaleFormatSettings(1033, Local); // pake US English

  case ConnectionType of
    scFirebird, scSQLServer:
      begin
        if DisplayTime then
        begin
          s := 'mm/dd/yyyy hh:nn:ss';
          if DisplayMilliSecond then
            s := s + '.zzz';
        end else
          s := 'mm/dd/yyyy';
      end;
    scMySQL:
      begin
        // MySQL doesn't support millisecond
        // so, don't display millisecond
        if DisplayTime then
        begin
          s := 'yyyy/mm/dd hh:nn:ss';
        end else
          s := 'yyyy/mm/dd';
      end;
  end;
  result := FormatDateTime(s, Value, Local)
end;

function fnStartDay(const AValue: TDateTime): TDateTime;
begin
  Result := Trunc(AValue);
end;

function fnEndDay(const AValue: TDateTime): TDateTime;
begin
  Result := RecodeTime(AValue, 23, 59, 59, 997); // untuk SQL Server 999 dibulatkan
jadi 0 menit berikutnya
end; // sehingga diganti 997 saja

function fnEndOfTheMonth(const AValue: TDateTime): TDateTime;
begin
  Result := fnEndDay(EndOfTheMonth(AValue));
end;

{ ================================================================
================
  Fungsi MontshBetween dari DateUtils menggunakan rata-rata perbulan sebanyak 30.4
hari
  sehingga untuk perhitungan sebenarnya jika mengambil banyaknya bulan antara
  1 Januari dan 1 Maret hanya menghasilkan 1, seharusnya 2, dikarenakan Februari
  mempunyai jumlah hari 28 (29 pada Tahun Kabisat)
  Sehingga di sini menggunakan perhitungan sebenarnya untuk per bulan

================================================================
================ }
function fnMonthsBetween(const ANow, AThen: TDate): Integer;
var Y1, M1, D1: Word;
  Y2, M2, D2: Word;
  Value: Integer;
begin
  DecodeDate(ANow, Y1, M1, D1);
  DecodeDate(AThen, Y2, M2, D2);

  if (Y2 - Y1) > 1 then
    Value := 12 + M2 + (12 - M1)
  else if (Y2 - Y1) = 1 then
    Value := M2 + (12 - M1)
  else if (Y2 - Y1) = 0 then
    Value := M2 - M1
  else
    Value := 0;

  Result := Value;
end;

function IncString(Source: string; LenStr: integer): string;
var TempVal: integer;
  s: string;
begin
  try
    TempVal := 0;
    TryStrToInt(Source, TempVal);
    Inc(TempVal);
    s := Format('%%.%d', [LenStr]);
    Result := Format(s + 'd', [TempVal]);
  except
  end
end;

function fnGetAccNumber(TransType: string; var IsManual: boolean; Own: boolean =
False;
  ADate: TDateTime = 0): string;
var
  TransNo: LongInt;
```

```
  NumOk: boolean;
  TempTransNum: string;
  TransDate: TDateTime;

  Qry: TADOQuery;
  Qry2: TADOQuery;

  TransFormula: string;
  Digit: integer;
begin
  qry := TADOQuery.Create(nil);
  qry.Connection := DM.DB;
  qry2 := TADOQuery.Create(nil);
  qry2.Connection := DM.DB;

  if Own then
    TransDate := ADate
  else
    TransDate := fnGetServerDate;

  fnSQLAdd(Qry, 'SELECT * FROM my_index WHERE mi_name = "Manual:' +
TransType + '" ', True);
  fnSQLOpen(Qry);

  if Qry.FieldByName('mi_value').AsString <> 'Y' then
  begin
    IsManual := False;

    // ambil rumus nomor transaksi
    fnSQLAdd(Qry, 'SELECT * FROM my_index WHERE mi_name = "TransNumber:' +
TransType + '" ', True);
    fnSQLOpen(Qry);

    TransFormula := fnTranslateFormula(Qry.FieldByName('mi_value').AsString);

    Digit := fnGetFormulaDigit(Qry.FieldByName('mi_value').AsString);

    fnSQLAdd(Qry, Format('SELECT TOP 100 gl_number FROM general_ledger ' +
      'WHERE gl_type = "%s" ' +
      'AND (gl_date >= ' + fnInputDateDB(fnStartDay(TransDate), False) +
      ' AND gl_date <= ' + fnInputDateDB(fnEndDay(TransDate), False) +
      ') AND gl_number LIKE "%%/%s.%%" ORDER BY gl_id DESC',
      [TransType, CasName]), True);
    fnSQLOpen(Qry);

    NumOK := False;
    TransNo := 0;
    while (not Qry.Eof) and (not NumOK) do
    begin
      TempTransNum := fnReplaceStr(Qry.FieldByName('gl_number').AsString,
Copy(Qry.FieldByName('gl_number').AsString, 1,
        Pos(fnGetFormulaSeparator(TransFormula),
Qry.FieldByName('gl_number').AsString)), '');

      if TryStrToInt(TempTransNum, TransNo) then
      begin
        while not NumOK do
        begin
          TransNo := TransNo + 1;
          TempTransNum := TransFormula + Format('%.' + IntToStr(Digit) + 'd', [TransNo]);

          fnSQLAdd(Qry2, Format('SELECT TOP 1 * FROM general_ledger ' +
            'WHERE gl_number = "%s" ' +
            'AND gl_type = "%s" ',
            [TempTransNum, TransType]), True);
          fnSQLOpen(Qry2);

          if Qry2.IsEmpty then
            NumOk := True;
        end;
      end;
      Qry.Next
    end; // while not Qry.Eof do

    if TransNo = 0 then TransNo := 1;
    TempTransNum := TransFormula + Format('%.' + IntToStr(Digit) + 'd', [TransNo]);

    Result := TempTransNum;
  end
  else
  begin
    IsManual := True;
    Result := '';
  end;

  FreeAndNil(qry);
  FreeAndNil(qry2);
end;

function fnAccTrans(TransType, TransDate, TransNumber, TransStatus: string;
  TransTotal: Double; TransNote, TransId: string; AGlId: string = ''; FromTable: string =
''): string;
var GlId: string;
  QryTemp1: TADOQuery;
begin
  QryTemp1 := TADOQuery.Create(nil);
  QryTemp1.Connection := dm.DB;

  if AGlId = '' then
  begin
    GlId := fnGenerateId
  else
    GlId := AGlId;

  fnSQLAdd(QryTemp1, 'INSERT INTO general_ledger (gl_id, gl_type, gl_number, ' +
    'gl_date, gl_status, gl_total, gl_note, operator_id, th_id, from_table) VALUES ' +
    '(:paramId, :paramType, :paramNumber, ' + TransDate +
    ', :paramStatus, :paramTotal, :paramNote, :paramOpId, :paramThId, :paramFT)', True);
  fnSQLParamByName(QryTemp1, 'paramId', GlId);
  fnSQLParamByName(QryTemp1, 'paramType', TransType);
  fnSQLParamByName(QryTemp1, 'paramNumber', TransNumber);
  fnSQLParamByName(QryTemp1, 'paramStatus', TransStatus);
  fnSQLParamByName(QryTemp1, 'paramTotal', TransTotal);
  fnSQLParamByName(QryTemp1, 'paramNote', TransNote);
  fnSQLParamByName(QryTemp1, 'paramOpId', OpId);
  fnSQLParamByName(QryTemp1, 'paramThId', TransId);
  fnSQLParamByName(QryTemp1, 'paramFT', FromTable);
  fnExecSQL(QryTemp1);

  Result := GlId;
  FreeAndNil(QryTemp1);
end;

procedure fnAccTransItem(GlId, ItemType, AccountId, ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = '');
var
  QryTemp1: TADOQuery;
begin
  QryTemp1 := TADOQuery.Create(nil);
  QryTemp1.Connection := dm.DB;

  fnSQLAdd(QryTemp1, 'INSERT INTO general_ledger_item (gli_id, gli_type, ' +
    'gl_id, acc_id, gli_number, gli_debet, gli_credit, gli_note, gli_ref) ' +
    'VALUES (:paramId, :paramType, :paramGlId, :paramAccId, :paramNumber, ' +
    ':paramDebet, :paramCredit, :paramNote, :paramRef)', True);
  fnSQLParamByName(QryTemp1, 'paramType', ItemType);
  fnSQLParamByName(QryTemp1, 'paramGlId', GlId);
  fnSQLParamByName(QryTemp1, 'paramId', fnGenerateId);
  fnSQLParamByName(QryTemp1, 'paramAccId', AccountId);
  fnSQLParamByName(QryTemp1, 'paramNumber', ItemNumber);

  if ItemType = 'Debet' then
  begin
    fnSQLParamByName(QryTemp1, 'paramDebet', ItemBalance);
    fnSQLParamByName(QryTemp1, 'paramCredit', 0);
  end
  else
  begin
    fnSQLParamByName(QryTemp1, 'paramDebet', 0);
    fnSQLParamByName(QryTemp1, 'paramCredit', ItemBalance);
  end;

  fnSQLParamByName(QryTemp1, 'paramNote', ItemNote);
  fnSQLParamByName(QryTemp1, 'paramRef', Ref);
  fnExecSQL(QryTemp1);

  FreeAndNil(QryTemp1);
end;

procedure fnAccTransItem(GlId: string; ItemType: TAccPostItem; AccountId: string;
ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = '');
begin
  case ItemType of
    apDebet: fnAccTransItem(GlId, 'Debet', AccountId, ItemNote, ItemNumber,
      ItemBalance, Ref);
    apCredit: fnAccTransItem(GlId, 'Credit', AccountId, ItemNote, ItemNumber,
      ItemBalance, Ref);
  end
end;

procedure fnAccTransItemName(GlId, ItemType, AccountName, ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = '');
var AccId: string;
  QryTemp1: TADOQuery;
begin
  QryTemp1 := TADOQuery.Create(nil);
  QryTemp1.Connection := dm.DB;

  fnSQLAdd(QryTemp1, 'SELECT * FROM account WHERE acc_name = ' +
QuotedStr(AccountName) +
    'AND acc_level = 4 AND acc_active = "Y" ', True);
  fnSQLOpen(QryTemp1);

  AccId := QryTemp1.FieldByName('acc_id').AsString;

  fnSQLAdd(QryTemp1, 'INSERT INTO general_ledger_item (gli_id, gli_type, ' +
    'gl_id, acc_id, gli_number, gli_debet, gli_credit, gli_note, gli_ref) ' +
    'VALUES (:paramId, :paramType, :paramGlId, :paramAccId, :paramNumber, ' +
    ':paramDebet, :paramCredit, :paramNote, :paramRef)', True);
  fnSQLParamByName(QryTemp1, 'paramType', ItemType);
  fnSQLParamByName(QryTemp1, 'paramGlId', GlId);
  fnSQLParamByName(QryTemp1, 'paramId', fnGenerateId);
  fnSQLParamByName(QryTemp1, 'paramAccId', AccId);
  fnSQLParamByName(QryTemp1, 'paramNumber', ItemNumber);

  if ItemType = 'Debet' then
  begin
    fnSQLParamByName(QryTemp1, 'paramDebet', ItemBalance);
    fnSQLParamByName(QryTemp1, 'paramCredit', 0);
```

```
  end
  else
  begin
    fnSQLParamByName(QryTemp1, 'paramDebet', 0);
    fnSQLParamByName(QryTemp1, 'paramCredit', ItemBalance);
  end;

  fnSQLParamByName(QryTemp1, 'paramNote', ItemNote);
  fnSQLParamByName(QryTemp1, 'paramRef', Ref);
  fnExecSQL(QryTemp1);

  FreeAndNil(QryTemp1);
end;

procedure fnAccTransItemName(GlId: string; ItemType: TAccPostItem; AccountName,
ItemNote: string;
  ItemNumber: integer; ItemBalance: Double; Ref: string = '');
begin
  case ItemType of
    apDebet: fnAccTransItemName(GlId, 'Debet', AccountName, ItemNote,
      ItemNumber, ItemBalance, Ref);
    apCredit: fnAccTransItemName(GlId, 'Credit', AccountName, ItemNote,
      ItemNumber, ItemBalance, Ref);
  end
end;

function fnWriteReg(RegName: string; RegType: string; Value: Variant): boolean;
var
  reg: TRegistry;
begin
  reg := TRegistry.Create(KEY_ALL_ACCESS);
  result := true;
  try
    reg.RootKey := HKEY_LOCAL_MACHINE;
    if reg.OpenKey('\Software\' + vrKeyLocation, true) then
    begin
      if Trim(LowerCase(RegType)) = 'string' then
      begin
        reg.WriteString(RegName, value);
      end
      else if Trim(LowerCase(RegType)) = 'datetime' then
      begin
        reg.WriteDateTime(RegName, value);
      end;
    end
    else
    begin
      result  := false;
    end;
  except
    result := false;
  end;
  FreeAndNil(reg);
end;

function fnReadReg(RegName: string; RegType: string): variant;
var
  reg: TRegistry;
begin
  reg := TRegistry.Create(KEY_ALL_ACCESS);
  try
    reg.RootKey := HKEY_LOCAL_MACHINE;
    if reg.OpenKey('\Software\' + vrKeyLocation, true) then
    begin
      if Trim(LowerCase(RegType)) = 'string' then
      begin
        result := reg.ReadString(RegName);
      end
      else if Trim(LowerCase(RegType)) = 'datetime' then
      begin
        result := reg.ReadDateTime(RegName);
      end;
    end;
  except
  end;
  FreeAndNil(reg);
end;

function fnReadIniFiles(Section, Identity, Default: string): string;
var Ini: TIniFile;
begin
  Result := Default;
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName) + 'config.ini');
  try
    Result := Ini.ReadString(Section, Identity, Default);
  finally
    Ini.Free
  end
end;

procedure fnWriteIniFiles(Section, Identity, Default: string);
var Ini: TIniFile;
begin
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName) + 'config.ini');
  try
    Ini.WriteString(Section, Identity, Default);
  finally
    Ini.Free
  end
end;
```

```
procedure fnLoadWallPaper;
var FN, Position: string;
begin
  FN := fnReadIniFiles('WALLPAPER', 'Path', '');
  Position := fnReadIniFiles('WALLPAPER', 'Posisi', 'Stretched');

  if (FN <> '') and (FileExists(FN)) then
  begin
    if Position = 'Kanan Atas' then
      Position := 'bpTopRight'
    else if Position = 'Kanan Bawah' then
      Position := 'bpBottomRight'
    else if Position = 'Kiri Atas' then
      Position := 'bpTopLeft'
    else if Position = 'Kiri Bawah' then
      Position := 'bpBottomLeft'
    else if Position = 'Stretched' then
      Position := 'bpStretched'
    else if Position = 'Stretched dengan Aspect' then
      Position := 'bpStretchedWithAspect'
    else if Position = 'Tengah' then
      Position := 'bpCenter'
    else if Position = 'Tile' then
      Position := 'bpTiled';

    FMain.Image1.PicturePosition :=
TPicturePosition(GetEnumValue(TypeInfo(TPicturePosition), Position));

    try
      FMain.Image1.Picture := nil;
      FMain.Image1.Picture.LoadFromFile(FN);
    except
    end;
  end;
end;

procedure fnChangeSkin;
var SkinNum: TBuiltInSkin,
  Num: integer;
begin
  try
    Num := 0;
    if vrSkinName <> 'Do Disabled Skin' then
    begin
      if Assigned(FMain) then
      begin
        if not FMain.suiSkinEngine1.Active then
          FMain.suiSkinEngine1.Active := True;
      end;
      if vrSkinBuiltIn then
      begin
        if vrSkinName = 'MacOS' then
          FMain.suiSkinEngine1.ApplyMainBuiltInSkin
        else
        begin
          SkinNum := TBuiltInSkin(GetEnumValue(TypeInfo(TBuiltInSkin), vrSkinName));

          case SkinNum of
            WinXPGreen: Num := 0;
            Vista1: Num := 1;
            Vista2: Num := 2;
            RealOne: Num := 3;
            MediaPlayer10: Num := 4;
            Office2007: Num := 5;
            DiamondBlue: Num := 6;
            Carlmness: Num := 7;
          end;

          if Assigned(FMain) then
            FMain.suiSkinEngine1.ApplyAdditionalBuiltInSkins(Num);
        end
      end
      else
      begin
        if Assigned(FMain) then
          FMain.suiSkinEngine1.SkinFile := vrSkinName;
      end;

//      if Assigned(FMain) then
//        FMain.suiSkinEngine1.DoSkinAllForms;
    end
    else
    begin
      if Assigned(FMain) then
        FMain.suiSkinEngine1.Active := False;
    end;
  finally
    if Assigned(FMain) then
      FMain.suiSkinEngine1.DoSkinAllForms;
  end;
  fnWriteReg('SkinName', 'string', vrSkinName);
  fnWriteReg('SkinBuiltIn', 'string', vrSkinBuiltIn);
end;

procedure fnLoadSkin;
var
  TempName: string;
  TempBuiltIn: boolean;
```

```
  s: string;
begin
  try
    TempName := fnReadReg('SkinName', 'string');
    if TempName <> '' then
      vrSkinName := TempName;

    try
      s := fnReadReg('SkinBuiltIn', 'string');

      if s = '' then s := 'True';
      TempBuiltIn := StrToBool(s)
    except
      TempBuiltIn := True;
    end;

    vrSkinBuiltIn := TempBuiltIn;

    fnChangeSkin;
  except
  end
end;

procedure fnLogoutOperator;
var Qry: TADOQuery;
begin
  Qry := TADOQuery.Create(nil);
  Qry.Connection := DM.DB;
  fnSQLAdd(dm.QryTemp2, 'UPDATE operator SET ' +
  ' operator_logout = "Y" WHERE operator_id = ' + QuotedStr(OpId), True);
  fnExecSQL(dm.QryTemp2);
  try
  except
    on E: exception do fnCheckNetwork(E);
  end;
  FreeAndNil(Qry)
end;

function fnCommOpen(VaComm: TVaComm; Port: byte; BaudRate: string;
  DataBits: string; Parity: string): boolean;
  function StrToBaudRate(BaudRate: string): TVaBaudrate;
  begin
    result := br9600;
    if BaudRate = 'br110' then result := br110
    else if BaudRate = 'br300' then result := br300
    else if BaudRate = 'br600' then result := br600
    else if BaudRate = 'br1200' then result := br1200
    else if BaudRate = 'br2400' then result := br2400
    else if BaudRate = 'br4800' then result := br4800
    else if BaudRate = 'br9600' then result := br9600
    else if BaudRate = 'br14400' then result := br14400
    else if BaudRate = 'br19200' then result := br19200
    else if BaudRate = 'br57600' then result := br57600
    else if BaudRate = 'br38400' then result := br38400
    else if BaudRate = 'br56000' then result := br56000
    else if BaudRate = 'br115200' then result := br115200
    else if BaudRate = 'br128000' then result := br128000
    else if BaudRate = 'br256000' then result := br256000;
  end;

  function StrToDataBits(DataBits: string): TVaDataBits;
  begin
    result := db8;
    if DataBits = 'db4' then result := db4
    else if DataBits = 'db5' then result := db5
    else if DataBits = 'db6' then result := db6
    else if DataBits = 'db7' then result := db7
    else if DataBits = 'db8' then result := db8;
  end;

  function StrToParity(Parity: string): TVaparity;
  begin
    result := paNone;
    if Parity = 'paEven' then result := paEven
    else if Parity = 'paMark' then result := paMark
    else if Parity = 'paNone' then result := paNone
    else if Parity = 'paOdd' then result := paOdd
    else if Parity = 'paSpace' then result := paSpace;
  end;
begin
  try
    VaComm.Close;
    VaComm.PortNum := Port;
    VaComm.Baudrate := StrToBaudRate(Baudrate);
    VaComm.Databits := StrToDataBits(DataBits);
    VaComm.Parity := StrToParity(Parity);
    Sleep(100);
    VaComm.Open;
    result := true;
  except
    result := false;
  end;
end;

function fnCommClose(VaComm: TVaComm): boolean;
begin
  try
    Sleep(100);
    VaComm.Close;
    result := true;
  except
    result := false;
  end
end;

procedure fnCDispClean(objComm: TVaComm);
begin
  try
    if vrCDispType = 'Wincor Nixdorf' then objComm.WriteText(#27 + '[2J')
    else if vrCDispType = 'EPSON' then objComm.WriteText(#$0C);
  except
  end;
end;

procedure fnCDispWrite(objComm: TVaComm; CDispType: string; xPos: integer;
  yPos: integer; str: string);
begin
  try
    if CDispType = 'Wincor Nixdorf' then
      objComm.WriteText(#27 + '[' + IntToStr(yPos) + ';' + IntToStr(xPos) + 'H' +
        copy(str, 1, vrCDispTextLength))
    else if CDispType = 'EPSON' then
      objComm.WriteText(#$1F + #$24 + chr(xPos) + chr(ypos) +
        copy(str, 1, vrCDispTextLength));
  except
  end;
end;

procedure fnCDrawOpen(VaComm: TVaComm; CDrawType: string);
begin
  try
    if CDrawType = 'Wincor Nixdorf' then
      VaComm.WriteText('A')
    else if CDrawType = 'Integral' then
      VaComm.WriteText('A')
    else
      VaComm.WriteText('A');
  except
  end;
end;

procedure fnCDispRunningText(objComm: TVaComm; TextLine1: string; TextLine2:
string; textWidth: integer);
var //i: integer;
  TextLength: integer;
  TempTimer: integer;
begin
  try
    TextLength := textWidth;
    TextLine1 := DupeString(' ', TextLength) + TextLine1;
    TextLine2 := DupeString(' ', TextLength) + TextLine2;

    if (Length(TextLine1) > Length(TextLine2)) then
      TextLength := Length(TextLine1)
    else
      TextLength := Length(TextLine2);

{   for i := 1 to TextLength do
    begin
      TextLine2 := TextLine2 + ' ';
    end;
}
    if Copy(TextLine2, Length(TextLine2), 1) <> ' ' then
      TextLine2 := TextLine2 + ' ';

    if vrCDispPos = 0 then fnCDispClean(objComm);

    vrCDispPos := vrCDispPos + 1;
    if vrCDispPos > TextLength then vrCDispPos := 0;

    tempTimer := vrCDispTimer;
    fnCDispWrite(objComm, vrCDispType, 1, 1, copy(TextLine1, TextLength -
vrCDispPos + 1, textWidth));
    fnCDispWrite(objComm, vrCDispType, 1, 2, copy(TextLine2, 1 + vrCDispPos,
textWidth));
    vrCDispTimer := tempTimer;
  except
  end;
end;

procedure fnLoadPrint(RptName: string; DvName: string = 'DvStg';
  Stg: TAdvStringGrid = nil; StartRow: integer = 0; EndRow: integer = 0);
var
  myPage: TRavePage;
  myRegion: TRaveRegion;
  i: integer;
  Size: string;
  Qry: TADOQuery;
  TempList: TStringList;
begin
  DM.RvReport.Close;
  DM.RvReport.Open;

  fnLoadPrintSetting;

  Qry := TADOQuery.Create(nil);
  Qry.Connection := dm.DB;

  // cari di global untuk setting Big, Medium, Small di lokal dahulu, jika tidak
```

```
// ditemukan cari di global
// pencarian ini khusus untuk printing yang ada di pengaturan printing
fnSQLAdd(Qry, 'SELECT REPLACE(mi_name, "PRINTTYPE:", "") AS MiName, ' +
  'mi_name, mi_value FROM my_index WHERE mi_name LIKE "PRINTTYPE:%" ' +
  'AND mi_value LIKE :paramValue', True);
fnSQLParamByName(Qry, 'paramValue', RptName + ';%');
fnSQLOpen(Qry);

if not Qry.IsEmpty then
begin
  Size := fnReadIniFiles('PRINTTYPE', Qry.FieldByName('MiName').AsString, '');

  TempList := TStringList.Create;
  if Size = '' then
    prExplodeStr(Qry.FieldByName('mi_value').AsString, ';', TempList)
  else
    prExplodeStr(Size, ';', TempList);

  // tambahkan karakter B, M, atau S
  RptName := RptName + TempList[1];

  FreeAndNil(TempList);
end;

DM.RvReport.SelectReport(RptName, False);

myPage  = DM.RvReport.ProjMan.FindRaveComponent(dm.RvReport.ReportName +
'.Page1', nil) as TRavePage;
if myPage <> nil then
begin
  //---------- cari region, posisikan sesuai margintop & marginleft ---------
  for i := 0 to myPage.ChildCount - 1 do
  begin
    if myPage.Child[i].ClassType = TRaveRegion then
    begin
      myRegion := myPage.Child[i] as TRaveRegion;
      myRegion.Left := vrPrintMarginLeft;
      myRegion.Top := vrPrintMarginTop;
    end;
  end;
end;

DM.RvReport.SetParam('parCompType', CompType);
DM.RvReport.SetParam('parCompName', CompName);
DM.RvReport.SetParam('parCompAddr', CompAddr);
DM.RvReport.SetParam('parCompCont', CompCont);
DM.RvReport.SetParam('parCompTelp', CompTelp);
DM.RvReport.SetParam('parCompFax', CompFax);
DM.RvReport.SetParam('parCompNPWP', CompNPWP);
DM.RvReport.SetParam('parOperator', OpName);
DM.RvReport.SetParam('parFooter', fnFooter);
DM.RvReport.SetParam('parDate', fnFormatDateTimeDB(fnGetServerDate));


//Stg
if Stg <> nil then
begin
  DM.RvStg.FAdvStg := stg;
  DM.RvStg.RowBegin := StartRow;
  DM.RvStg.RowEnd := EndRow
end;
//---------

//refresh
CreateFields(DM.RvReport.ProjMan.FindRaveComponent(DvName, nil) as
TRaveBaseDataView, nil, nil, true);
//---------
DM.RvReport.Execute;
DM.RvReport.Close;
end;

procedure fnLoadPrint(RptName: string; DbGrid: TDBGridEh);
begin
  dm.RvClient.DataSet := DBGrid.DataSource.DataSet;
  fnLoadPrint(RptName, 'DvClient')
end;

procedure fnLoadLocalSetting;
var
  temp: string;
begin
  //Kasir
  CasName := fnReadReg('Cashier', 'String');

  //Open Price
  if fnReadReg('OpenPrice', 'String') = 'T' then
    vrOpenPrice := true
  else
    vrOpenPrice := false;

  //Customer Display
  if fnReadReg('CDispActive', 'String') = 'T' then
    vrCDispActive := true
  else
    vrCDispActive := false;
  temp := fnReadReg('CDispType', 'String');
  if temp = '' then
    vrCDispType := 'Custom'
  else
    vrCDispType := temp;
  temp := fnReadReg('CDispPort', 'String');
  if temp = '' then
    vrCDispPort := 1
  else
    vrCDispPort := StrToInt(temp);
  temp := fnReadReg('CDispBaudRate', 'String');
  if temp = '' then
    vrCDispBaudRate := 'br9600'
  else
    vrCDispBaudRate := temp;
  temp := fnReadReg('CDispDataBits', 'String');
  if temp = '' then
    vrCDispDataBits := 'db8'
  else
    vrCDispDataBits := temp;
  temp := fnReadReg('CDispParity', 'String');
  if temp = '' then
    vrCDispParity := 'paNone'
  else
    vrCDispParity := temp;
  temp := fnReadReg('CDispTextLength', 'String');
  if temp = '' then
    vrCDispTextLength := 20
  else
    vrCDispTextLength := StrToInt(temp);
  temp := fnReadReg('CDispDelay', 'String');
  if temp = '' then
    vrCDispDelay := 20
  else
    vrCDispDelay := StrToInt(temp);
  temp := fnReadReg('CDispText1', 'String');
  if temp = '' then
    vrCDispText1 := 'Terima Kasih'
  else
    vrCDispText1 := temp;
  temp := fnReadReg('CDispText2', 'String');
  if temp = '' then
    vrCDispText2 := 'Terima Kasih'
  else
    vrCDispText2 := temp;

  //Cash Drawer
  if fnReadReg('CDrawActive', 'String') = 'T' then
    vrCDrawActive := true
  else
    vrCDrawActive := false;
  temp := fnReadReg('CDrawType', 'String');
  if temp = '' then
    vrCDrawType := 'Custom'
  else
    vrCDrawType := temp;
  temp := fnReadReg('CDrawPort', 'String');
  if temp = '' then
    vrCDrawPort := 1
  else
    vrCDrawPort := StrToInt(temp);
  temp := fnReadReg('CDrawBaudRate', 'String');
  if temp = '' then
    vrCDrawBaudRate := 'br9600'
  else
    vrCDrawBaudRate := temp;
  temp := fnReadReg('CDrawDataBits', 'String');
  if temp = '' then
    vrCDrawDataBits := 'db8'
  else
    vrCDrawDataBits := temp;
  temp := fnReadReg('CDrawParity', 'String');
  if temp = '' then
    vrCDrawParity := 'paNone'
  else
    vrCDrawParity := temp;

  //Barcode Scanner
  temp := fnReadReg('BScanSource', 'String');
  if temp = '' then
    vrBScanSource := 'Keyboard Port'
  else
    vrBScanSource := temp;
  temp := fnReadReg('BScanType', 'String');
  if temp = '' then
    vrBScanType := 'Custom'
  else
    vrBScanType := temp;
  temp := fnReadReg('BScanPort', 'String');
  if temp = '' then
    vrBScanPort := 1
  else
    vrBScanPort := StrToInt(temp);
  temp := fnReadReg('BScanBaudRate', 'String');
  if temp = '' then
    vrBScanBaudRate := 'br9600'
  else
    vrBScanBaudRate := temp;
  temp := fnReadReg('BScanDataBits', 'String');
  if temp = '' then
    vrBScanDataBits := 'db8'
  else
    vrBScanDataBits := temp;
  temp := fnReadReg('BScanParity', 'String');
  if temp = '' then
```

```
    vrBScanParity := 'paNone'
  else
    vrBScanParity := temp;

  //printing
  try
    fnSQLAdd(DM.qryTemp1, 'SELECT * FROM my_index WHERE mi_name =
"ROUNDING" ', True);
    fnSQLOpen(DM.qryTemp1);

    if DM.qryTemp1.FieldByName('mi_value').AsString <> '' then
      vrRounded := StrToBool(DM.qryTemp1.FieldByName('mi_value').AsString)
    else
      vrRounded := False;
  except
  end
end;

procedure fnLoadPrintSetting;
var hPrinter: THandle;
begin
  try
    //---------------load print output ---------------
    vrPrintDefaultDest := fnReadIniFiles('PRINTER', 'OUTPUT', '');
    if trim(vrPrintDefaultDest) = '' then vrPrintDefaultDest := 'PREVIEW';

    if vrPrintDefaultDest = 'PRINTER' then dm.RvSystem.DefaultDest := rdPrinter;
    if vrPrintDefaultDest = 'PREVIEW' then dm.RvSystem.DefaultDest := rdPreview;

    //---------------printer name ---------------
    vrPrinterName := fnReadIniFiles('PRINTER', 'NAME', '');

    //---------------check apakah printer valid ---------------
    try
      if not OpenPrinter(pChar(vrPrinterName), hPrinter, nil) then vrPrinterName := '';
      ClosePrinter(hPrinter);
    except
      ClosePrinter(hPrinter);
      vrPrinterName := '';
    end;
    //---------------------------------------------------

    if Trim(vrPrinterName) = '' then vrPrinterName := fnGetPrinterProperties('Name');

    //---------------printer paper ---------------
    vrPrinterPaper := fnReadIniFiles('PRINTER', 'PAPER', '');
    if trim(vrPrinterPaper) = '' then vrPrinterPaper := fnGetPrinterProperties('Paper');

    //---------------paper size ---------------
    vrPrintPaperSize := 0;
    tryStrToCurr(fnReadIniFiles('PRINTER', 'PAPERSIZE', ''), vrPrintPaperSize);

    //---------------paper margin top ---------------
    vrPrintMarginTop := 0.3;
    tryStrToCurr(fnReadIniFiles('PRINTER', 'MARGINTOP', ''), vrPrintMarginTop);

    //---------------paper margin left ---------------
    vrPrintMarginLeft := 0.3;
    tryStrToCurr(fnReadIniFiles('PRINTER', 'MARGINLEFT', ''), vrPrintMarginLeft);

    //---------------output type ---------------
    vrPrintType := fnReadIniFiles('PRINTER', 'PRINT TYPE', '');
    if trim(vrPrintType) = '' then vrPrintType := 'GRAPHIC';

    fnFooter := fnReadIniFiles('FOOTER', 'FOOTER', '');

    RpDev.SelectPrinter(vrPrinterName, true);
    RpDev.SelectPaper(vrPrinterPaper, true);
  except
  end;
end;

procedure fnLoadRegionalId(ID: LCID);
var i: integer;
begin
  GetLocaleFormatSettings(ID, vrRgnFS);

  CurrencyFormat := vrRgnFS.CurrencyFormat;
  NegCurrFormat  := vrRgnFS.NegCurrFormat;
  ThousandSeparator := vrRgnFS.ThousandSeparator;
  DecimalSeparator := vrRgnFS.DecimalSeparator;
  CurrencyDecimals := vrRgnFS.CurrencyDecimals;
  DateSeparator := vrRgnFS.DateSeparator;
  TimeSeparator := vrRgnFS.TimeSeparator;
  ListSeparator := vrRgnFS.ListSeparator;

  // khusus untuk Kode mata uang dihapus saja
  // CurrencyString := vrRgnFS.CurrencyString;
  CurrencyString := '';
  ShortDateFormat := vrRgnFS.ShortDateFormat;
  LongDateFormat := vrRgnFS.LongDateFormat;
  TimeAMString := vrRgnFS.TimeAMString;
  TimePMString := vrRgnFS.TimePMString;
  ShortTimeFormat := vrRgnFS.ShortTimeFormat;
  LongTimeFormat := vrRgnFS.LongTimeFormat;

  for i := 0 to 12 do
    ShortMonthNames[i] := vrRgnFS.ShortMonthNames[i];

  for i := 0 to 12 do
    LongMonthNames[i] := vrRgnFS.LongMonthNames[i];

  for i := 0 to 7 do
    ShortDayNames[i] := vrRgnFS.ShortDayNames[i];

  for i := 0 to 7 do
    LongDayNames[i] := vrRgnFS.LongDayNames[i];

  TwoDigitYearCenturyWindow := vrRgnFS.TwoDigitYearCenturyWindow;
end;

procedure fnLoadRegionalSetting;
var Rgn: string;
  LID: integer;
begin
  Rgn := Trim(fnReadReg('Regional', 'String'));
  if Rgn = '' then
  begin
    Rgn := 'Indonesian';
    LID := 1057;
  end
  else
    LID := fnReadReg('RegionalID', 'String');

  fnLoadRegionalId(LID);
end;

function fnGetPrinterProperties(propType: string): string;
var
  pDMode: PDevMode;
  tempResult: string;
begin
  tempResult := '';

  pDMode := RPDev.DevMode;

  if pDMode <> nil then
  begin
    if propType = 'Name' then tempResult := pDMode^.dmDeviceName;
    if propType = 'Paper' then tempResult := pDMode^.dmFormName;
  end;

  Result := tempResult;
end;

procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
var
  L: Word;
const
{$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
{$J-}
begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;

function SaveToExcel(DBGrid: TDBGrid): boolean;
const
{$J+}CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
var
  FStream: TFileStream;
  I, J: Integer;
  AFileName: string;
begin
  prCreateForm('TFSaveExcel, FSaveExcel, 'ShowModal');
  AFileName := Trim(FSaveExcel.EFileName.Text);

  Result := False;
  if AFileName <> '' then
  begin
    FStream := nil;

    try
      FStream := TFileStream.Create(PChar(AFileName), fmCreate or fmOpenWrite);

      CXlsBof[4] := 0;
      FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));

      //---------- tulis judul ----------
      for i := 0 to dbGrid.Columns.Count - 1 do
      begin
        XlsWriteCellLabel(FStream, i, 0, dbGrid.Columns[i].Title.Caption);
      end;

      i := 1;

      dbGrid.DataSource.DataSet.First;

      while not dbGrid.DataSource.DataSet.Eof do
      begin
        //---------- tulis isi ----------
        for j := 0 to dbGrid.Columns.Count - 1 do
```

```
      begin
        if dbGrid.Columns[j].Field <> nil then
          if (dbGrid.Columns[j].Field is TDateTimeField) then
            XlsWriteCellLabel(FStream, j, i, FormatDateTime('dd/mm/yyyy hh:nn:ss',
dbGrid.Columns[j].Field.AsDateTime))
          else
            XlsWriteCellLabel(FStream, j, i, dbGrid.Columns[j].Field.AsString);
      end;

      i := i + 1;
      dbGrid.DataSource.DataSet.Next;
    end;

    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;

  if Result then
  begin
    fnMessage(DataSaved, mINFO);
  end
  else
  begin
    fnMessage('Data gagal disimpan.', mERROR);
  end;
 end;
 FreeAndNil(FSaveExcel)
end;

function SaveToExcel(StGrid: TAdvStringGrid): boolean; overload;
var
 AFileName: string;
 GrdExcel: TAdvGridExcelIO;
begin
 GrdExcel := TAdvGridExcelIO.Create(nil);
 GrdExcel.AdvStringGrid := StGrid;

 prCreateForm('TFSaveExcel, FSaveExcel, 'ShowModal');
 AFileName := Trim(FSaveExcel.EFileName.Text);

 Result := False;
 if AFileName <> '' then
 begin
   try
     Screen.Cursor := crHourGlass;
     //StGrid.SaveToXLS(AFileName); // diganti metode yang lebih bagus
     GrdExcel.XLSExport(AFileName); // menjadi export karena hasilnya sesuai
     screen.Cursor := crDefault;
     Result := True;
   except
     Result := False
   end;
 end;

 if Result then
 begin
   fnMessage(DataSaved, mINFO);
 end
 else
 begin
   if AFileName <> '' then
     fnMessage('Data gagal disimpan.', mERROR);
 end;
 FreeAndNil(FSaveExcel);
 FreeAndNil(GrdExcel);
end;

function SaveToExcel(DBGrid: TDBGridEh): boolean;
var
 AFileName: string;
begin
 prCreateForm('TFSaveExcel, FSaveExcel, 'ShowModal');
 AFileName := Trim(FSaveExcel.EFileName.Text);

 Result := False;
 if (not FSaveExcel.Canceled) then
 begin
   if AFileName <> '' then
   begin
     try
       Screen.Cursor := crHourGlass;
       SaveDBGridEhToExportFile(TDBGridEhExportAsXLS, DBGrid, AFileName, true);
       screen.Cursor := crDefault;
       Result := True;
     except
       Result := False
     end;
   end;

   if Result then
   begin
     fnMessage(DataSaved, mINFO);
   end
   else
   begin
     if AFileName <> '' then
       fnMessage('Data gagal disimpan.', mERROR);
   end;
```

```
 end;
 FreeAndNil(FSaveExcel);
end;

function FMod(x, y: int64): int64;
begin
 Result := (x - (Trunc(x / y) * y))
end;

function fnTerbilang(x: int64): string;
const
 abil: array[0..11] of string[10] = ('', 'Satu', 'Dua', 'Tiga',
   'Empat', 'Lima', 'Enam', 'Tujuh', 'Delapan', 'Sembilan',
   'Sepuluh', 'Sebelas');

begin
 if (x < 12) then
   Result := ' ' + abil[x]
 else if (x < 20) then
   Result := fnTerbilang(x - 10) + ' Belas'
 else if (x < 100) then
   Result := fnTerbilang(x div 10) + ' Puluh' + fnTerbilang(x mod 10)
 else if (x < 200) then
   Result := ' Seratus' + fnTerbilang(x - 100)
 else if (x < 1000) then
   Result := fnTerbilang(x div 100) + ' Ratus' + fnTerbilang(x mod 100)
 else if (x < 2000) then
   Result := ' Seribu' + fnTerbilang(x - 1000)
 else if (x < 1000000) then
   Result := fnTerbilang(x div 1000) + ' Ribu' + fnTerbilang(x mod 1000)
 else if (x < 1000000000) then // milyar
   Result := fnTerbilang(Trunc(x / 1000000)) + ' Juta' + fnTerbilang(FMod(x, 1000000))
 else if (x < 1000000000000) then // trilyun
   Result := fnTerbilang(Trunc(x / 1000000000)) + ' Milyar' + fnTerbilang(FMod(x,
1000000000))
 else if (x < 1000000000000000) then // bilyun
   Result := fnTerbilang(Trunc(x / 1000000000000)) + ' Trilyun' + fnTerbilang(FMod(x,
1000000000000))
 else if (x < 1000000000000000000) then // ??
   Result := fnTerbilang(Trunc(x / 1000000000000000)) + ' Bilyun' +
fnTerbilang(FMod(x, 1000000000000000));
end;

function fnTerbilang(x: currency): string;
var Value1: int64;
  Value2: int64;
begin
 Value1 := Trunc(x);
 Value2 := Round((x - Value1) * 100);
 if Value2 = 0 then
   Result := fnReplaceStr(Trim(fnTerbilang(Value1)), '  ', ' ')
 else if Value2 < 10 then
   Result := fnReplaceStr(Trim(fnTerbilang(Value1) + ' Koma Nol' +
fnTerbilang(Value2)), '  ', ' ')
 else
   Result := fnReplaceStr(Trim(fnTerbilang(Value1) + ' Koma' + fnTerbilang(Value2)), '
', ' ');
end;

procedure fnDataSetFormat(DataSet: TDataset);
var
 c: Integer;
 sFormat: string;
begin
 try
   for c := 0 to DataSet.FieldCount - 1 do
   begin
     //---------- khusus untuk TFloatField ----------
     if (DataSet.Fields[c] is TFloatField) or (DataSet.Fields[c] is TBCDField) then
     begin
       sFormat := '#,#0 ' + DupeString('0', vrDecimal);
       if DataSet.Fields[c] is TFloatField then
         (DataSet.Fields[c] as TFloatField).DisplayFormat := sFormat
       else
         (DataSet.Fields[c] as TBCDField).DisplayFormat := sFormat
     end

     //---------- khusus untuk TIntegerField ----------
     else if DataSet.Fields[c] is TIntegerField then
     begin
       sFormat := '#,#0';
       (DataSet.Fields[c] as TIntegerField).DisplayFormat := sFormat;
     end;
   end;
 except
 end;
end;

function FindWindowX(ACaption, AClass: string): THandle;
var t: THandle;
  Title: array[0..255] of char;
  TheClass: array[0..64] of char;
//  i: integer;
//  j: integer;
begin
 t := GetWindow(Application.Handle, GW_HWNDFIRST);
 while t > 0 do
 begin
   {i := }GetWindowText(t, Title, 255);
```

```
    {j := }GetClassName(t, TheClass, 64);

   if (Title = ACaption) and (TheClass = AClass) then
     Break
   else
     t := GetWindow(t, GW_HWNDNEXT);
  end;

  Result := t;
end;

function IsCompressed: boolean;
var
  iFileHandle: Integer;
  iFileLength: Integer;
  iBytesRead: Integer;
  Buffer: PChar;
  i: Integer;
  Str: string;
  FileName: string;
begin
  Str := '';
  try
    FileName := Application.ExeName;
    if FileExists(ChangeFileExt(FileName, '.tmp')) then
    begin
      SysUtils.DeleteFile(ChangeFileExt(FileName, '.tmp'));
    end;

    if CopyFile(PChar(FileName), PChar(ChangeFileExt(FileName, '.tmp')), False) then
    begin
      iFileHandle := FileOpen(ChangeFileExt(FileName, '.exe'), fmOpenRead);
      iFileLength := FileSeek(iFileHandle, 0, 2);
      FileSeek(iFileHandle, 0, 0);
      Buffer := PChar(AllocMem(iFileLength + 1));
      iBytesRead := FileRead(iFileHandle, Buffer^, iFileLength);

      FileClose(iFileHandle);

      if iFileLength > 0 then
      begin
        for i := 0 to 1000 do // find data
        begin
          case Integer(Buffer[i]) of
            0..31: Str := Str + '';
          else
            Str := Str + Chr(Integer(Buffer[i]));
          end;
        end;
      end;
      if (PosEx(fnRC2Decrypt('IbAItPJZ6A=='), Str) > 0) or
  (PosEx(fnRC2Decrypt('laUSOrhD8w=='), Str) > 0) then
        Result := True
      else
        Result := False;
      end
      else
        Result := False;

      if FileExists(ChangeFileExt(FileName, '.tmp')) then
        SysUtils.DeleteFile(ChangeFileExt(FileName, '.tmp'))
    finally
      FreeMem(Buffer);
    end;
  end;

  { TSkinnedForm }

  procedure TSkinnedForm.RemoveSkin(Sender: TObject);
  begin
    if Assigned(FMain) then
      FMain.suiSkinEngine1.RemoveForm(Self);
    inherited;
  end;

  function fnCheckTransManual(TransNum, TransType: string): boolean;
  var Qry: TADOQuery;
  begin
    Qry := TADOQuery.Create(nil);
    Qry.Connection := dm.DB;

    fnSQLAdd(Qry, 'SELECT * FROM transaction_history WHERE th_number
  = :paramNumber ' +
      'AND th_type = :paramType ' +
      //AND th_status <> "Cancel"
      ' ', True);
    fnSQLParamByName(Qry, 'paramNumber', TransNum);
    fnSQLParamByName(Qry, 'paramType', TransType);
    fnSQLOpen(Qry);

    if not Qry.IsEmpty then
    begin
      if DM.DB.InTransaction then DM.DB.RollbackTrans;
      Result := False;
    end
    else
      Result := True;
  end;
```

```
end.
```

Umain.pas

```
unit UMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ComCtrls, mxProtector, ExtCtrls, ExeInfo, AdvPicture,
  SUISkinEngine, SUISkin_Rave, XPMan;

type
  TFMain = class(TForm)
    Menu1: TMenuItem;
    AksesKeluar1: TMenuItem;
    N1: TMenuItem;
    Keluar1: TMenuItem;
    HakAkses1: TMenuItem;
    Kewenangan1: TMenuItem;
    Operator1: TMenuItem;
    ransaksi1: TMenuItem;
    Laporan1: TMenuItem;
    Arsip1: TMenuItem;
    AlatBantu1: TMenuItem;
    Database1: TMenuItem;
    N6: TMenuItem;
    Calculator1: TMenuItem;
    Paiman: TmxProtector;
    DatabaseServer1: TMenuItem;
    N7: TMenuItem;
    StrukturDatabase1: TMenuItem;
    MasterData1: TMenuItem;
    ProfilPerusahaan1: TMenuItem;
    ExeInfo1: TExeInfo;
    Timer1: TTimer;
    Image1: TAdvPicture;
    MainMenu1: TMainMenu;
    suiStatusBar1: TStatusBar;
    suiSkinEngine1: TsuiSkinEngine;
    N9: TMenuItem;
    XPManifest1: TXPManifest;
    procedure PaimanDayTrial(Sender: TObject; DaysRemained: Integer);
    procedure PaimanExpiration(Sender: TObject);
    procedure PaimanGetSerialNumber(Sender: TObject; var UserName,
      SerialNumber: string);
    procedure PaimanInvalidSerialNumber(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure AksesKeluar1Click(Sender: TObject);
    procedure Keluar1Click(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure Kewenangan1Click(Sender: TObject);
    procedure DatabaseServer1Click(Sender: TObject);
    procedure StrukturDatabase1Click(Sender: TObject);
    procedure Operator1Click(Sender: TObject);
    procedure Registrasi1Click(Sender: TObject);
    procedure NomorKomputer1Click(Sender: TObject);
    procedure ProfilPerusahaan1Click(Sender: TObject);
    procedure Calculator1Click(Sender: TObject);
    procedure Printer1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure GeneralSetting1Click(Sender: TObject);
    procedure General1Click(Sender: TObject);
    procedure Profil1Click(Sender: TObject);
  private
    Trial: Integer;
    procedure ShowLogin;
    procedure CloseForms;
    { Private declarations }
  public
    Logined: Boolean;
    DaysRemaining: Integer;
    { Public declarations }
  end;

var
  FMain: TFMain;

implementation

uses UFunc, URegister, ULogin, UPrivillege, UDBServer, UCheckDB, UOperator,
  UCashierSetting, UCompany, UPrinterSetting, UOSettingGeneral, UAbout,
  UOSettingGlobal;

{$R *.dfm}
{$R LOGO.RES}

procedure TFMain.CloseForms;
var
  i: Integer;
begin
  for i := 0 to Screen.FormCount - 1 do
  begin
    if (Screen.Forms[i].Name <> 'FMain') and
      (Screen.Forms[i].Name <> 'FLogin') then
    begin
      Screen.Forms[i].Close;
    end;
  end;
end;
```

```
procedure TFMain.PaimanDayTrial(Sender: TObject;
  DaysRemained: Integer);
begin
  DaysRemaining := DaysRemained;
end;

procedure TFMain.PaimanExpiration(Sender: TObject);
begin
  if not PaiMan.IsRegistered then
    prCreateForm(TFRegister, FRegister, 'ShowModal');
  if not PaiMan.IsRegistered then
  begin
    IsTerminated := True;
    Close;
  end
end;

procedure TFMain.PaimanGetSerialNumber(Sender: TObject; var UserName,
  SerialNumber: string);
begin
  UserName := FRegister.EUserName.Text;
  SerialNumber := FRegister.ESerial.Text;
end;

procedure TFMain.PaimanInvalidSerialNumber(Sender: TObject);
begin
  if Trial < 3 then
  begin
    fnMessage('Serial yang anda masukkan salah.', mERROR);
    Inc(Trial)
  end
  else
  begin
    fnMessage('Anda sudah memasukkan nomor serial yang salah sebanyak 3 kali.',
mERROR);
    Close
  end
end;

procedure TFMain.FormCreate(Sender: TObject);
begin
  Trial := 0;
  fnLoadSkin;
  fnLoadWallpaper;
end;

procedure TFMain.ShowLogin;
begin
  prCreateForm(TFLogin, FLogin, 'ShowModal');
  if not FLogin.LoginOK then
    Application.Terminate
  else
  begin
    Logined := True;
  end;

  FLogin.Cleanup
end;

procedure TFMain.AksesKeluar1Click(Sender: TObject);
begin
  if fnMessage('Anda akan akses keluar program ?', mCONFIRM) = IDYES then
  begin
    Caption := Application.Title;
    Logined := False;

    fnResetMenu('', MainMenu1);

    fnLogoutOperator;
    CloseForms;
    ShowLogin;

    if not Logined then
      Close;
  end
end;

procedure TFMain.Keluar1Click(Sender: TObject);
begin
  Close
end;

procedure TFMain.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if Logined and (Trial < 3) then
  begin
    if fnMessage('Anda belum akses keluar.'#13'Apakah akan keluar program ?',
mCONFIRM) = IDYES then
      begin
        Logined := False;
      end
      else
        CanClose := False
    end
      {  else
        begin
          if mfMsgConfirm('Anda akan keluar program ?') = mrOK then
          begin

        Logined := False;
      end
      else
        CanClose := False
    end}
end;

procedure TFMain.Kewenangan1Click(Sender: TObject);
begin
  CloseForms;
  prCreateForm(TFPrivillege, FPrivillege);
end;

procedure TFMain.DatabaseServer1Click(Sender: TObject);
begin
  CloseForms;
  prCreateForm(TFDBServer, FDBServer, 'ShowModal');
end;

procedure TFMain.StrukturDatabase1Click(Sender: TObject);
begin
  CloseForms;
  prCreateForm(TFCheckDB, FCheckDB, 'ShowModal');
end;

procedure TFMain.Operator1Click(Sender: TObject);
begin
  CloseForms;
  prCreateForm(TFOperator, FOperator);
end;

procedure TFMain.Registrasi1Click(Sender: TObject);
begin
  prCreateForm(TFRegister, FRegister, 'ShowModal');
end;

procedure TFMain.NomorKomputer1Click(Sender: TObject);
begin
  CloseForms;
  prCreateForm(TFCashierSetting, FCashierSetting, 'ShowModal');
end;

procedure TFMain.ProfilPerusahaan1Click(Sender: TObject);
begin
  prCreateForm(TFCompany, FCompany, 'ShowModal');
end;

procedure TFMain.Calculator1Click(Sender: TObject);
begin
  if WinExec('CALC.EXE', SW_SHOWNORMAL + WS_EX_DLGMODALFRAME) <
32 then
    fnMessage('Aplikasi kalkulator tidak dapat dijalankan.', mERROR);
end;

procedure TFMain.Printer1Click(Sender: TObject);
begin
  prCreateForm(TFPrinterSetting, FPrinterSetting, 'ShowModal');
end;

procedure TFMain.Timer1Timer(Sender: TObject);
begin
  suiStatusBar1.Panels[0].Text := OpName;
  suiStatusBar1.Panels[1].Text := FormatDateTime('dd-mmmm-yyyy hh:nn:ss', Now);
  suiStatusBar1.Panels[3].Text := 'Compiled: ' +
fnFormatDateIndo(FileDateToDateTime(FileAge(Application.ExeName)));
end;

procedure TFMain.GeneralSetting1Click(Sender: TObject);
begin
  prCreateForm(TFOSettingGeneral, FOSettingGeneral, 'ShowModal');
end;

procedure TFMain.General1Click(Sender: TObject);
begin
  prCreateForm(TFOSettingGlobal, FOSettingGlobal, 'ShowModal');
end;

procedure TFMain.Profil1Click(Sender: TObject);
begin
  prCreateForm(TFAbout, FAbout, 'ShowModal');
end;


end.
```

UDM.pas

```
unit UDM;

interface

uses
  SysUtils, Classes, DB, ADODB, RpBase, RpSystem, RpRenderText, StrUtils,
  RpRenderPDF, RpRenderHTML, RpRender, RpRenderRTF, RpDefine, RpRave, Forms,
  RpCon, UMBStgCon, StdCtrls, ComCtrls, RpConDS;

type
  TResultDB = array of string;

  Tdm = class(TDataModule)
```

```
    DB: TADOConnection;
    QryTemp1: TADOQuery;
    QryTemp2: TADOQuery;
    QryTemp3: TADOQuery;
    QryTemp4: TADOQuery;
    RvReport: TRvProject;
    RvRenderRTF1: TRvRenderRTF;
    RvRenderHTML1: TRvRenderHTML;
    RvRenderPDF1: TRvRenderPDF;
    RvRenderText1: TRvRenderText;
    RvSystem: TRvSystem;
    Cmd1: TADOCommand;
    QryTempList: TADOQuery;
    DsTempList: TDataSource;
    RvStg: TMBRvStgConnection;
    RvClient: TRvDataSetConnection;
    QryStock: TADOQuery;
    QryTemp5: TADOQuery;
    ADOTable1: TADOTable;
    ADODataSet1: TADODataSet;
    ADODataSet2: TADODataSet;
    ADOTable2: TADOTable;

    procedure DataSetFormat(DataSet: TDataSet);
    function fnCheckDefaultData: Boolean;

    procedure DataModuleDestroy(Sender: TObject);
    procedure DataModuleCreate(Sender: TObject);
  private
    function fnSetIndex(TableName: string): TResultDB;
    function fnSetTable(TableName: string): TResultDB;
    { Private declarations }
  public
    function fnCheckDBStructure(ProgressBar: TProgressBar = nil; ACaption: TLabel =
nil): Boolean;
    { Public declarations }
  end;

function fnCheckTransManual(TransNum, TransType: string; TableName: string = 'th'):
Boolean;

var
  dm: Tdm;

implementation

uses UFunc;

{$R *.dfm}
```

```
function TDM fnCheckDBStructure(ProgressBar: TProgressBar = nil; ACaption: TLabel
= nil): Boolean;

  procedure IncProgress(StrCaption: string = '');
  begin
    if Assigned(ProgressBar) then
      ProgressBar.Position := ProgressBar.Position + 1;

    if Assigned(ACaption) then
      ACaption.Caption := StrCaption;

    Application.ProcessMessages
  end;

var
  TempResult: Boolean;
  TableList: TStringList;
  TableName: string;
begin
  if Assigned(ProgressBar) then
    ProgressBar.Max := 80; // ---> ingat ini harus selalu diganti jika menambah atau
mengurangi table

  if (ConnectionType = scSQLServer) or (ConnectionType = scMySQL) then
  begin
    //Check and create database
    try
      fnSQLAdd(Cmd1, 'USE ' + DBName);
      fnExecSQL(Cmd1);
      IncProgress;
    except
      fnSQLAdd(Cmd1, 'CREATE DATABASE ' + DBName);
      fnExecSQL(Cmd1);

      fnSQLAdd(Cmd1, 'USE ' + DBName);
      fnExecSQL(Cmd1);
    end;
  end;

  //Check tabel
  TableList := TStringList.Create;
  TempResult := True;
  DB.GetTableNames(TableList);
  TableName := 'COMPANY';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'GENERAL_LEDGER';
  if TempResult then
```

```
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'GENERAL_LEDGER_ITEM';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'MY_INDEX';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'OPERATOR';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'OP_PERMISSION';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'PRIVILEGE';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'PR';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'PR_HISTORY';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'TRANSACTION_HISTORY';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);
  TableName := 'TRANSACTION_ITEM';
  if TempResult then
    TempResult := fnCheckTableStructure(TableList, fnSetTable(TableName),
TableName);
  IncProgress(TableName);

  // INDEX TABLE
```

```
  DB.GetTableNames(TableList);
  TableName := 'COMPANY';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'GENERAL_LEDGER';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'GENERAL_LEDGER_ITEM';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'MY_INDEX';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'OPERATOR';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'OP_PERMISSION';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'PRIVILEGE';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'PR';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'TRANSACTION_HISTORY';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
  IncProgress('IDX_' + TableName);
  TableName := 'TRANSACTION_ITEM';
  if TempResult then
    TempResult := fnCheckIndexStructure(fnSetIndex(TableName), TableName,
TableList, 'IDX_' + TableName);
```

```
  IncProgress('IDX_' + TableName);

  // stored procedure / user defined function
  Result := TempResult;
  FreeAndNil(TableList);
end;

procedure Tdm.DataModuleDestroy(Sender: TObject);
begin
  RvReport.Close;
  fnCleanLog
end;

function Tdm.fnCheckDefaultData: Boolean;

  procedure InsertMenuTransaction(TransType, TransName, TransNumFormula: string);
  begin
    fnSQLAdd(QryTemp1,
      'SELECT * FROM my_index WHERE mi_name = ' + QuotedStr(TransName), True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      fnSQLAdd(Cmd1,
        'INSERT INTO my_index (mi_id, mi_name, mi_value) ' +
        'VALUES (:paramId, :paramName, :paramValue)');
      fnSQLParamByName(Cmd1, 'paramId', fnGenerateId);
      fnSQLParamByName(Cmd1, 'paramName', TransName);
      fnSQLParamByName(Cmd1, 'paramValue', TransType);
      fnExecSQL(Cmd1);
    end;

    fnSQLAdd(QryTemp1,
      'SELECT * FROM my_index WHERE mi_name = "TransType" AND mi_value = ' +
QuotedStr(TransName), True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      fnSQLAdd(Cmd1,
        'INSERT INTO my_index (mi_id, mi_name, mi_value) ' +
        'VALUES (:paramId, :paramName, :paramValue)');
      fnSQLParamByName(Cmd1, 'paramId', fnGenerateId);
      fnSQLParamByName(Cmd1, 'paramName', 'TransType');
      fnSQLParamByName(Cmd1, 'paramValue', TransName);
      fnExecSQL(Cmd1);
    end;

    fnSQLAdd(QryTemp1,
      'SELECT * FROM my_index WHERE mi_name = "TransNumber:' + TransType + '" ',
True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      fnSQLAdd(Cmd1,
        'INSERT INTO my_index (mi_id, mi_name, mi_value) ' +
        'VALUES (:paramId, :paramName, :paramValue)');
      fnSQLParamByName(Cmd1, 'paramId', fnGenerateId);
      fnSQLParamByName(Cmd1, 'paramName', 'TransNumber:' + TransType);
      fnSQLParamByName(Cmd1, 'paramValue', TransNumFormula);
      fnExecSQL(Cmd1);
    end;
  end;

  procedure InsertPrintOption(TransType, PrintSize: string);
  begin
    fnSQLAdd(QryTemp1,
      'SELECT * FROM my_index WHERE mi_name = "PRINTTYPE.' + TransType + '" ',
True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      fnSQLAdd(Cmd1,
        'INSERT INTO my_index (mi_id, mi_name, mi_value) ' +
        'VALUES (:paramId, :paramName, :paramValue)');
      fnSQLParamByName(Cmd1, 'paramId', fnGenerateId);
      fnSQLParamByName(Cmd1, 'paramName', 'PRINTTYPE.' + TransType);
      fnSQLParamByName(Cmd1, 'paramValue', PrintSize);
      fnExecSQL(Cmd1);
    end;
  end;

  procedure InsertAcc1(Id, Code1, Name: string);
  begin
    fnSQLAdd(Cmd1, 'INSERT INTO account(acc_id, acc_code1, acc_level. acc_name, '
+
      'acc_active, acc_system) VALUES
(:paramId, :paramCode1, :paramLevel, :paramName, ' +
      ':paramActive, "Y")');
    fnSQLParamByName(Cmd1, 'paramId', Id);
    fnSQLParamByName(Cmd1, 'paramCode1', Code1);
    fnSQLParamByName(Cmd1, 'paramLevel', '1');
    fnSQLParamByName(Cmd1, 'paramName', Name);
    fnSQLParamByName(Cmd1, 'paramActive', 'Y');
    fnExecSQL(Cmd1);
  end;

  procedure InsertAcc2(Id, ParentId, Code1, Code2, Name: string);
  begin
    fnSQLAdd(Cmd1, 'INSERT INTO account(acc_id, acc_code1, acc_code2, acc_level,
acc_name, ' +
      'acc_active, acc_parent_id, acc_system) VALUES (:paramId, :paramCode1, ' +
      ':paramCode2, :paramLevel, :paramName, :paramActive, :paramParentId, "Y")');
    fnSQLParamByName(Cmd1, 'paramId', Id);
    fnSQLParamByName(Cmd1, 'paramCode1', Code1);
    fnSQLParamByName(Cmd1, 'paramCode2', Code2);
    fnSQLParamByName(Cmd1, 'paramLevel', '2');
    fnSQLParamByName(Cmd1, 'paramName', Name);
    fnSQLParamByName(Cmd1, 'paramActive', 'Y');
    fnSQLParamByName(Cmd1, 'paramParentId', ParentId);
    fnExecSQL(Cmd1);
  end;

  procedure InsertAcc3(Id, ParentId, Code1, Code2, Code3, Name: string);
  begin
    fnSQLAdd(Cmd1, 'INSERT INTO account(acc_id, acc_code1, acc_code2, acc_code3,
' +
      'acc_level, acc_name, acc_active, acc_parent_id, acc_system) ' +
      'VALUES (:paramId, :paramCode1, :paramCode2, ' +
      ':paramCode3, :paramLevel, :paramName, ' +
      ':paramActive, :paramParentId, "Y")');
    fnSQLParamByName(Cmd1, 'paramId', Id);
    fnSQLParamByName(Cmd1, 'paramCode1', Code1);
    fnSQLParamByName(Cmd1, 'paramCode2', Code2);
    fnSQLParamByName(Cmd1, 'paramCode3', Code3);
    fnSQLParamByName(Cmd1, 'paramLevel', '3');
    fnSQLParamByName(Cmd1, 'paramName', Name);
    fnSQLParamByName(Cmd1, 'paramActive', 'Y');
    fnSQLParamByName(Cmd1, 'paramParentId', ParentId);
    fnExecSQL(Cmd1);
  end;

  procedure InsertAcc4(Id, ParentId, Code1, Code2, Code3, Code4, Name, NS: string);
  begin
    fnSQLAdd(Cmd1, 'INSERT INTO account(acc_id, acc_code1, acc_code2, acc_code3,
' +
      'acc_code4, acc_level, acc_name, acc_active, acc_parent_id, acc_normal_saldo, ' +
      'acc_system) VALUES (:paramId, :paramCode1, ' +
      ':paramCode2, :paramCode3, :paramCode4, :paramLevel, :paramName, :paramActive,
' +
      ':paramParentId, :paramNS, "Y")');
    fnSQLParamByName(Cmd1, 'paramId', Id);
    fnSQLParamByName(Cmd1, 'paramCode1', Code1);
    fnSQLParamByName(Cmd1, 'paramCode2', Code2);
    fnSQLParamByName(Cmd1, 'paramCode3', Code3);
    fnSQLParamByName(Cmd1, 'paramCode4', Code4);
    fnSQLParamByName(Cmd1, 'paramLevel', '4');
    fnSQLParamByName(Cmd1, 'paramName', Name);
    fnSQLParamByName(Cmd1, 'paramActive', 'Y');
    fnSQLParamByName(Cmd1, 'paramParentId', ParentId);
    fnSQLParamByName(Cmd1, 'paramNS', NS);
    fnExecSQL(Cmd1);
  end;

var
  OpLevelId: string;
  TempId1: string;
  TempId2: string;
  TempId3: string;
  TempId4: string;
begin
  Result := True;
  try
    //data Golongan harga jual
    fnSQLAdd(QryTemp1, 'SELECT * FROM privilege', True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      OpLevelId := 'MASTER';

      fnSQLAdd(Cmd1,
        'INSERT INTO privilege (privilege_id, privilege_name, ' +
        'privilege_note, privilege_permission) ' +
        'VALUES (:paramId, :paramName, :paramDesc, :paramPermission)');
      fnSQLParamByName(Cmd1, 'paramId', OpLevelId);
      fnSQLParamByName(Cmd1, 'paramName', 'Master');
      fnSQLParamByName(Cmd1, 'paramDesc', 'Level Dasar Master');
      fnSQLParamByName(Cmd1, 'paramPermission',
'icld5pnPOJzfOMp2fW6Ph4j3RVsQUpiVxYcwHQjmE9fzcioxCi4='); // not yet
implemented
      fnExecSQL(Cmd1);
    end
    else
    begin
      OpLevelId := QryTemp1.FieldByName('privilege_id').AsString;
    end;

    //data user_level
    fnSQLAdd(QryTemp1, 'SELECT * FROM operator', True);
    fnSQLOpen(QryTemp1);

    if QryTemp1.IsEmpty then
    begin
      OpId := 'MASTER';

      fnSQLAdd(Cmd1,
```

```
      'INSERT INTO operator (operator_id, privilege_id, ' +
      'operator_name, operator_user, operator_pass, operator_logout) ' +
      'VALUES ( :paramId, :paramPrivId, :paramName, :paramUserName, ' +
      ':paramPassword, :paramLogout)');
    fnSQLParamByName(Cmd1, 'paramId', OpId);
    fnSQLParamByName(Cmd1, 'paramPrivId', OpLevelId);
    fnSQLParamByName(Cmd1, 'paramName', 'Master');
    fnSQLParamByName(Cmd1, 'paramUserName', 'master'); // not yet implemented
    fnSQLParamByName(Cmd1, 'paramPassword', '1pvCNdlC');
    fnSQLParamByName(Cmd1, 'paramLogout', 'Y');
    fnExecSQL(Cmd1);
  end;



//    InsertMenuTransaction('PURCHASE ORDER', 'Pesanan Pembelian',
'PBL/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('BONUS', 'Transaksi Bonus', 'BN/[NO].[THN][BLN][TGL]-
;6');
//    InsertMenuTransaction('PURCHASE', 'Pembelian', 'BL/[NO].[THN][BLN][TGL]-
;6');
//    InsertMenuTransaction('OTHER', 'Transaksi Penerimaan Barang Lain-lain',
'OTHER/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('OUT-OTHER', 'Transaksi Pengeluaran Barang Lain-lain',
'OUT-OTHER/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('CASH IN', 'Cash In', 'CI/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('CASH OUT', 'Cash Out', 'CO/[NO].[THN][BLN][TGL]-;6');
//
//    // untuk akuntansi
//    InsertMenuTransaction('ACC GENERAL', 'Jurnal Umum',
'JU/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('ACC ADJUSTMENT', 'Jurnal Penyesuaian',
'JS/[NO].[THN][BLN][TGL]-;6');
//    InsertMenuTransaction('ACC CLOSING', 'Jurnal Penutup',
'JP/[NO].[THN][BLN][TGL]-;6');

    // printing
//    InsertPrintOption('Data Akun (COA)', 'DAkun;B');
//    InsertPrintOption('Trans Pesanan Pembelian', 'TPesanBeli;M');
//    InsertPrintOption('Trans Pembelian', 'TBeli;M');
//    InsertPrintOption('Trans Penerimaan Produk Indent', 'TTerimaProdukIndent;M');
//    InsertPrintOption('Trans Bonus', 'TBonus;M');
//    InsertPrintOption('Trans Pemasukan Lain Lain', 'TPemasukanLainLain;M');
//    InsertPrintOption('Trans Peminjaman Barang', 'TPeminjamanBarang;M');
//    InsertPrintOption('Trans Pengembalian Barang', 'TPengembalianBarang;M');
//    InsertPrintOption('Trans Ambang', 'TAmbang;M');
//    InsertPrintOption('Trans Booking', 'TBooking;S');
//    InsertPrintOption('Trans Registrasi', 'TRegistrasi;M');
//    InsertPrintOption('Trans Pesanan Penjualan Langsung', 'TPesanJual;M');
//    InsertPrintOption('Trans Penjualan Langsung', 'TJualLangsung;M');
//    InsertPrintOption('Trans Penjualan Melalui Nota Ambang', 'TJualNotaAmbang;M');
//    InsertPrintOption('Trans Penjualan Dari Nota Pinjam', 'TJualNotaPinjam;M');
//    InsertPrintOption('Trans Penyerahan Indent', 'TPenyerahanIndent;M');
//    InsertPrintOption('Trans Penerimaan', 'TPenerimaanKas;S');
//    InsertPrintOption('Trans Pengeluaran', 'TPengeluaranKas;S');
//    InsertPrintOption('Trans PembayaranSewa', 'TPembayaranSewa;S');
//    InsertPrintOption('Arsip Pesanan Pembelian', 'APesanBeli;M');
//    InsertPrintOption('Arsip Pesanan Pembelian Detail', 'APesanBeliDetail;M');
//    InsertPrintOption('Arsip Pesanan Pembelian Koreksi', 'APesanBeliKoreksi;M');
//    InsertPrintOption('Arsip Pembelian', 'ABeli,M');
//    InsertPrintOption('Arsip Pembelian Detail', 'ABeliDetail;M');
//    InsertPrintOption('Arsip Pembelian Koreksi', 'ABeliKoreksi;M');
//    InsertPrintOption('Arsip Penerimaan Indent', 'ATerimaIndent;M');
//    InsertPrintOption('Arsip Penerimaan Indent Detail', 'ATerimaIndentDetail;M');
//    InsertPrintOption('Arsip Penerimaan Indent Koreksi','ATerimaIndentKoreksi;M');
//    InsertPrintOption('Arsip Bonus', 'ABonus;M');
//    InsertPrintOption('Arsip Bonus Detail', 'ABonusDetail;M');
//    InsertPrintOption('Arsip Bonus Koreksi', 'ABonusKoreksi;M');
//    InsertPrintOption('Arsip Pemasukan Lain Lain', 'APemasukanLainLain;M');
//    InsertPrintOption('Arsip Pemasukan Lain Lain Detail',
'APemasukanLainLainDetail;M');
//    InsertPrintOption('Arsip Pemasukan Lain Lain Koreksi',
'APemasukanLainLainKoreksi;M');
//    InsertPrintOption('Arsip Peminjaman Barang', 'APeminjamanBarang;M');
//    InsertPrintOption('Arsip Peminjaman Barang Detail', 'APeminjamanBarangDetail;M');
//    InsertPrintOption('Arsip Peminjaman Barang Koreksi',
'APeminjamanBarangKoreksi;M');
//    InsertPrintOption('Arsip Pengembalian Barang', 'APengembalianBarang;M');
//    InsertPrintOption('Arsip Pengembalian Barang Detail',
'APengembalianBarangDetail;M');
//    InsertPrintOption('Arsip Pengembalian Barang Koreksi',
'APengembalianBarangKoreksi;M');
//    InsertPrintOption('Arsip Ambang', 'AAmbang;M');
//    InsertPrintOption('Arsip Ambang Detail', 'AAmbangDetail;M');
//    InsertPrintOption('Arsip Ambang Koreksi', 'AAmbangKoreksi;M');
//    InsertPrintOption('Arsip Pesanan Penjualan Langsung', 'APesanJual;M');
//    InsertPrintOption('Arsip Pesanan Penjualan Langsung Detail', 'APesanJualDetail;M');
//    InsertPrintOption('Arsip Penjualan Langsung', 'AJualLangsung;M');
//    InsertPrintOption('Arsip Penjualan Langsung Detail', 'AJualLangsungDetail;M');
//    InsertPrintOption('Arsip Penjualan Melalui Nota Ambang', 'AJualNotaAmbang;M');
//    InsertPrintOption('Arsip Penjualan Melalui Nota Ambang Detail',
'AJualNotaAmbangDetail;M');
//    InsertPrintOption('Arsip Penjualan Dari Nota Pinjam', 'AJualNotaPinjam;M');
//    InsertPrintOption('Arsip Penjualan Dari Nota Pinjam Detail',
'AJualNotaPinjamDetail;M');
//    InsertPrintOption('Arsip Penyerahan Indent', 'APenyerahanIndent;M');
//    InsertPrintOption('Arsip Penyerahan Indent Detail', 'APenyerahanIndentDetail;M');
//    InsertPrintOption('Arsip Penerimaan', 'APenerimaan;M');
//    InsertPrintOption('Arsip Pengeluaran', 'Apengeluaran;M');
  except
```

```
    Result := False;
  end;
end;

function Tdm.fnSetIndex(TableName: string): TResultDB;
var
  Table: TResultDB;
begin
  if UpperCase(TableName) = 'COMPANY' then
  begin
    SetLength(Table, 1);
    Table[0] := 'company_name';
  end
  else if UpperCase(TableName) = 'GENERAL_LEDGER' then
  begin
    SetLength(Table, 5);
    Table[0] := 'gl_number ,';
    Table[1] := 'gl_date ,';
    Table[2] := 'gl_type ,';
    Table[3] := 'gl_status ,';
    Table[4] := 'th_id ';
  end
  else if UpperCase(TableName) = 'GENERAL_LEDGER_ITEM' then
  begin
    SetLength(Table, 3);
    Table[0] := 'gl_id ,';
    Table[1] := 'gli_type ,';
    Table[2] := 'acc_id ';
  end
  else if UpperCase(TableName) = 'MY_INDEX' then
  begin
    SetLength(Table, 2);
    Table[0] := 'mi_name ,';
    Table[1] := 'mi_value';
  end
  else if UpperCase(TableName) = 'OPERATOR' then
  begin
    SetLength(Table, 2);
    Table[0] := 'privilege_id ,';
    Table[1] := 'operator_name';
  end
  else if UpperCase(TableName) = 'OP_PERMISSION' then
  begin
    SetLength(Table, 3);
    Table[0] := 'privilege_id ,';
    Table[1] := 'permission_type ,';
    Table[2] := 'permission_value';
  end
  else if UpperCase(TableName) = 'PRIVILEGE' then
  begin
    SetLength(Table, 1);
    Table[0] := 'privilege_name';
  end
  else if UpperCase(TableName) = 'PR' then
  begin
    SetLength(Table, 3);
    Table[0] := 'th_id ,';
    Table[1] := 'pr_type ,';
    Table[2] := 'pr_due_date';
  end
  else if UpperCase(TableName) = 'PR_HISTORY' then
  begin
    SetLength(Table, 4);
    Table[0] := 'pr_id ,';
    Table[1] := 'operator_id ,';
    Table[2] := 'pr_history_type ,';
    Table[3] := 'pr_history_date';
  end
  else if UpperCase(TableName) = 'TRANSACTION_HISTORY' then
  begin
    SetLength(Table, 8);
    Table[0] := 'supplier_id ,';
    Table[1] := 'member_id ,';
    Table[2] := 'stokist_id ,';
    Table[3] := 'operator_id ,';
    Table[4] := 'th_number ,';
    Table[5] := 'th_type ,';
    Table[6] := 'th_date ,';
    Table[7] := 'th_status';
  end
  else if UpperCase(TableName) = 'TRANSACTION_ITEM' then
  begin
    SetLength(Table, 5);
    Table[0] := 'ti_type ,';
    Table[1] := 'th_id ,';
    Table[2] := 'stock_id ,';
    Table[3] := 'saldo_quantity ,';
    Table[4] := 'saldo_total ';
  end
    ;

  Result := Table;
end;

function Tdm.fnSetTable(TableName: string): TResultDB;
var
  Table: TResultDB;
begin
  if UpperCase(TableName) = 'COMPANY' then
  begin
```

```
  SetLength(Table, 6);
  Table[0] := 'company_name VARCHAR (100) NOT NULL ,';
  Table[1] := 'company_addr VARCHAR (100) ,';
  Table[2] := 'company_telp VARCHAR (100) ,';
  Table[3] := 'company_npwp VARCHAR (100) ,';
  Table[4] := 'company_contact VARCHAR (100) ,';
  Table[5] := 'PRIMARY KEY (company_name)';
end
else if TableName = 'GENERAL_LEDGER' then
begin
  SetLength(Table, 11);
  Table[0] := 'gl_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'gl_type VARCHAR (30),';
  Table[2] := 'gl_number VARCHAR (50) ,';
  Table[3] := 'gl_date DATETIME ,';
  Table[4] := 'gl_status VARCHAR (10) NULL ,';
  Table[5] := 'gl_total DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[6] := 'gl_note VARCHAR (500) ,';
  Table[7] := 'operator_id VARCHAR (17) ,';
  Table[8] := 'th_id VARCHAR (17) ,'; // penghubung ke transaction_history
  Table[9] := 'from_table VARCHAR (50) ,';
  Table[10] := 'PRIMARY KEY (gl_id)';
end
else if TableName = 'GENERAL_LEDGER_ITEM' then
begin
  SetLength(Table, 10);
  Table[0] := 'gli_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'gli_type VARCHAR (30) NOT NULL ,';
  Table[2] := 'gl_id VARCHAR (17) ,';
  Table[3] := 'acc_id VARCHAR (17) ,';
  Table[4] := 'gli_number INT ,';
  Table[5] := 'gli_debet DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[6] := 'gli_credit DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[7] := 'gli_note VARCHAR (500) ,';
  Table[8] := 'gli_ref VARCHAR (100) ,';
  Table[9] := 'PRIMARY KEY (gli_id)';
end
else if UpperCase(TableName) = 'MY_INDEX' then
begin
  SetLength(Table, 4);
  Table[0] := 'mi_id CHAR (17) NOT NULL ,';
  Table[1] := 'mi_name VARCHAR (100) NOT NULL ,';
  Table[2] := 'mi_value VARCHAR (200) NOT NULL ,';
  Table[3] := 'PRIMARY KEY (mi_id)';
end
else if UpperCase(TableName) = 'OPERATOR' then
begin
  SetLength(Table, 9);
  Table[0] := 'operator_id VARCHAR(17) NOT NULL ,';
  Table[1] := 'privilege_id VARCHAR (17) ,';
  Table[2] := 'operator_status VARCHAR (1) ,';
  Table[3] := 'operator_user VARCHAR (50) ,';
  Table[4] := 'operator_pass VARCHAR (50) ,';
  Table[5] := 'operator_name VARCHAR (50) ,';
  Table[6] := 'operator_logout VARCHAR (50) ,';
  Table[7] := 'operator_last_login DATETIME ,';
  Table[8] := 'PRIMARY KEY (operator_id) '
end
else if UpperCase(TableName) = 'OP_PERMISSION' then
begin
  SetLength(Table, 3);
  Table[0] := 'privilege_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'permission_type VARCHAR (20) ,';
  Table[2] := 'permission_value CHAR (1) ';
  //    Table[3] := 'PRIMARY KEY (privilege_id) '
end
else if UpperCase(TableName) = 'PRIVILEGE' then
begin
  SetLength(Table, 5);
  Table[0] := 'privilege_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'privilege_name VARCHAR (50) ,';
  Table[2] := 'privilege_note VARCHAR (50) ,';
  Table[3] := 'privilege_permission VARCHAR (1000) ,';
  Table[4] := 'PRIMARY KEY (privilege_id) '
end
else if UpperCase(TableName) = 'PR' then
begin
  SetLength(Table, 7);
  Table[0] := 'pr_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'th_id VARCHAR (17) ,';
  Table[2] := 'pr_type VARCHAR (50) ,';
  Table[3] := 'pr_saldo DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[4] := 'pr_due_date DATETIME ,';
  Table[5] := 'pr_status VARCHAR (50) ,';
  Table[6] := 'PRIMARY KEY (pr_id)';
end
else if UpperCase(TableName) = 'PR_HISTORY' then
begin
  SetLength(Table, 10);
  Table[0] := 'pr_history_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'pr_id VARCHAR (17) ,';
  Table[2] := 'operator_id VARCHAR (17) ,';
  Table[3] := 'pr_history_type VARCHAR (50) ,';
  Table[4] := 'pr_history_amount DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[5] := 'pr_history_date DATETIME ,';
  Table[6] := 'pr_history_payment VARCHAR (20) ,';
  Table[7] := 'bank_id VARCHAR (17) ,';
  Table[8] := 'pr_history_note VARCHAR (255) ,';
  Table[9] := 'PRIMARY KEY (pr_history_id)'
end
else if UpperCase(TableName) = 'TRANSACTION_HISTORY' then
begin
  SetLength(Table, 35);
  Table[0] := 'th_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'supplier_id VARCHAR (17) ,';
  Table[2] := 'member_id VARCHAR (17) ,';
  Table[3] := 'stokist_id VARCHAR (17) ,';
  Table[4] := 'operator_id VARCHAR (17) ,';
  Table[5] := 'th_number VARCHAR (50) ,';
  Table[6] := 'th_type VARCHAR (50) NOT NULL ,';
  Table[7] := 'th_payment_type VARCHAR (50) ,';
  Table[8] := 'th_disc DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[9] := 'th_disc_type VARCHAR (1) DEFAULT "A" ,'; // tipe " atau '%'
  Table[10] := 'th_tax DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[11] := 'th_tax_type VARCHAR (1) DEFAULT "A" ,'; // tipe " atau '%'
  Table[12] := 'th_round DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[13] := 'th_dp DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[14] := 'th_total DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[15] := 'th_date DATETIME NOT NULL ,';
  Table[16] := 'th_date_return DATETIME ,';
  Table[17] := 'th_note VARCHAR (100) ,';
  Table[18] := 'th_status VARCHAR (50) ,';
  Table[19] := 'bank_id VARCHAR (17) ,'; // untuk transaksi kas bank
  Table[20] := 'ma_id VARCHAR (17) ,'; // untuk transaksi kas
  Table[21] := 'ed_id VARCHAR (17) ,'; // untuk event discount
  Table[22] := 'category_id VARCHAR (17) ,'; // untuk pembelian & order pembelian
  Table[23] := 'th_ori_id VARCHAR (17) ,';
  Table[24] := 'th_from VARCHAR (30) ,'; // Diterima dari (Penerimaan Kas)
  Table[25] := 'th_to VARCHAR (30) ,'; // Untuk
  Table[26] := 'member_code VARCHAR (50) ,'; // kode
  Table[27] := 'member_name VARCHAR (100) ,'; // nama
  Table[28] := 'member_id_ref VARCHAR (50)  ,'; // ref id
  Table[29] := 'member_name_ref VARCHAR (100) ,'; // nama
  Table[30] := 'member_id_refs VARCHAR (50)  ,'; // Referensor id
  Table[31] := 'member_name_refs VARCHAR (100) ,'; // nama
  Table[32] := 'stokist_code VARCHAR (50) ,';
  Table[33] := 'stokist_name VARCHAR (100) ,';
  Table[34] := 'PRIMARY KEY (th_id)'
end
else if UpperCase(TableName) = 'TRANSACTION_ITEM' then
begin
  SetLength(Table, 23);
  Table[0] := 'ti_id VARCHAR (17) NOT NULL ,';
  Table[1] := 'ti_number INT DEFAULT 0 NOT NULL ,';
  Table[2] := 'ti_type VARCHAR (50) ,';
  Table[3] := 'th_id VARCHAR (17) ,';
  Table[4] := 'stock_parent_id VARCHAR (20) ,';
  Table[5] := 'stock_id VARCHAR (17) ,';
  Table[6] := 'stock_type VARCHAR (15) ,';
  Table[6] := 'ti_disc DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[7] := 'ti_disc_type VARCHAR (1) DEFAULT "A" NOT NULL ,'; // tipe " atau '%'
  Table[8] := 'buy_quantity DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[9] := 'buy_price DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[10] := 'sell_quantity DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[11] := 'ti_quantity_detail DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[12] := 'sell_hpp DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[13] := 'sell_price DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[14] := 'other_quantity DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[15] := 'other_hpp DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[16] := 'saldo_quantity DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[17] := 'saldo_total DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[18] := 'ti_assembly_cost DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[19] := 'retur_quantity DEC (38,18) DEFAULT 0 NOT NULL ,';
  Table[20] := 'ed_id VARCHAR (17) ,'; // untuk event discount
  Table[21] := 'stock_type VARCHAR (17) ,'; // untuk event discount
  Table[22] := 'PRIMARY KEY (ti_id)';
end
  ;

  Result := Table;
end;

procedure Tdm.DataSetFormat(DataSet: TDataSet);
var
  c: Integer;
  sFormat: string;
begin
  try
    for c := 0 to DataSet.FieldCount - 1 do
    begin
      //---------------khusus untuk TFloatField ---------------
      if (DataSet.Fields[c] is TFloatField) or (DataSet.Fields[c] is TBCDField) then
      begin
        if vrDecimal <= 0 then
        begin
          sFormat := '#,#0';
        end
        else
        begin
          sFormat := '#,#0.00';
        end;

        if DataSet.Fields[c] is TFloatField then
          (DataSet.Fields[c] as TFloatField).DisplayFormat := sFormat
        else
          (DataSet.Fields[c] as TBCDField).DisplayFormat := sFormat
      end

      //---------------khusus untuk TIntegerField ---------------
```

```
      else if DataSet.Fields[c] is TIntegerField then
      begin
        sFormat := '#,#0';
        (DataSet.Fields[c] as TIntegerField).DisplayFormat := sFormat;
      end;
    end;
  except
  end;
end;

function fnCheckTransManual(TransNum, TransType: string; TableName: string = 'th'):
Boolean;
var
  Qry: TADOQuery;
begin
  Qry := TADOQuery.Create(nil);
  Qry.Connection := dm.DB;

  if TableName = 'th' then
  begin
    fnSQLAdd(Qry, 'SELECT * FROM transaction_history WHERE th_number
= :paramNumber ' +
      'AND th_type = :paramType', True);
    fnSQLParamByName(Qry, 'paramNumber', TransNum);
    fnSQLParamByName(Qry, 'paramType', TransType);
    fnSQLOpen(Qry);
  end
  else if TableName = 'tr' then
  begin
    fnSQLAdd(Qry, 'SELECT * FROM transaction_rent WHERE tr_number
= :paramNumber ' +
      'AND tr_type = :paramType', True);
    fnSQLParamByName(Qry, 'paramNumber', TransNum);
    fnSQLParamByName(Qry, 'paramType', TransType);
    fnSQLOpen(Qry);
  end;

  if not Qry.IsEmpty then
  begin
    if fnIntransaction then
      fnRollBack;
    Result := False;
  end
  else
    Result := True;
end;

procedure Tdm.DataModuleCreate(Sender: TObject);
begin
  try
    RvReport.ProjectFile := ExtractFilePath(Application.ExeName) + 'TIENZ.rav';
    RvReport.Open;
  except
  end
end;

end.
```

Usplash.pas

```
unit USplash;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, jpeg, Gauges;

type
  TFSplash = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Gauge1: TGauge;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FSplash: TFSplash;

implementation

uses UMain;

{$R *.dfm}

procedure TFSplash.FormCreate(Sender: TObject);
begin
  Label1.Caption := 'Build: ' + FMain.ExeInfo1.FileVersion;
  Gauge1.MaxValue := FMain.Paiman.MaxDayNumber;
  Gauge1.Progress := FMain.Paiman.MaxDayNumber - FMain.DaysRemaining;
end;

end.
```

ULogin.pas

```
unit ULogin;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, JPEG, Menus, ADODB;

type
  TFLogin = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    ColorDialog1: TColorDialog;
    PopupMenu1: TPopupMenu;
    GantiWarna1: TMenuItem;
    GroupBox1: TGroupBox;
    btnLogin: TButton;
    btnExit: TButton;
    edtUser: TEdit;
    edtPass: TEdit;
    Image1: TImage;
    procedure suitempbtnExitClick(Sender: TObject);
    procedure suitempedtUserKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure suitempbtnLoginClick(Sender: TObject);
    procedure suitempbtnLoginKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormCreate(Sender: TObject);
    procedure GantiWarna1Click(Sender: TObject);
  private
    Count: integer;
    { Private declarations }
  public
    LoginOK: boolean;
    procedure Cleanup;
    { Public declarations }
  end;

var
  FLogin: TFLogin;

implementation

uses
  UFunc, UDM, UMain, UFirstCash;

{$R *.dfm}

procedure TFLogin.suitempbtnExitClick(Sender: TObject);
begin
  Close;
end;

procedure TFLogin.suitempedtUserKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    VK_RETURN: Perform(WM_NEXTDLGCTL, 0, 0);
    VK_ESCAPE: Close;
  end;
end;

procedure TFLogin.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if OpId = '' then
  begin
    if fnMessage('Apakah anda ingin keluar sekarang ?', mCONFIRM) = IDYES then
    begin
      IsTerminated := true;
      CanClose := True;
      Application.Terminate;
    end
    else
    begin
      CanClose := false;
    end;
  end;
end;

procedure TFLogin.suitempbtnLoginClick(Sender: TObject);
var OpLevelId: string;
  function fnCheckPermission(IdxName: string): boolean;
  var
    Qry: TADOQuery;
  begin
    result := false;
    Qry := TADOQuery.Create(nil);
    Qry.Connection := DM.DB;

    try
      fnSQLAdd(Qry,
        'SELECT * FROM op_permission ' +
        'WHERE privilege_id = ' + QuotedStr(OpLevelId) + ' AND permission_value = ' +
        QuotedStr(IdxName), True);
      fnSQLOpen(Qry);

      if Qry.FieldByName('permission_value').AsString = 'Y' then result := true;
    except
      Result := False
    end;
    FreeAndNil(Qry);
  end;
```

```
var
  User: string;
  Pass: string;
  LoginDate: TDateTime;
begin
  try
    User := Trim(edtUser.Text);
    Pass := Trim(edtPass.Text);

    if User <> '' then
    begin
      fnSQLAdd(DM.QryTemp1,
        'SELECT * FROM operator ' +
        'LEFT JOIN privilege ON privilege.privilege_id = operator.privilege_id ' +
        'WHERE operator_user = '' + User + '' ', True);
      fnSQLOpen(DM.QryTemp1);

      if dm.QryTemp1.RecordCount > 1 then
      begin
        fnSQLAdd(DM.QryTemp1,
          'SELECT * FROM operator ' +
          'LEFT JOIN privilege ON privilege.privilege_id = operator.privilege_id ' +
          'WHERE operator_user = '' + User + '' ', True);
        fnSQLOpen(DM.QryTemp1);
      end;

      if not DM.QryTemp1.IsEmpty then
      begin
        if fnRC2Decrypt(DM.QryTemp1.FieldByName('operator_pass').AsString) = Pass
then
        begin
          OpLevel := DM.QryTemp1.FieldByName('privilege_name').AsString;
          OpLevelId := DM.QryTemp1.FieldByName('privilege_id').AsString;
          OpId := DM.QryTemp1.FieldByName('operator_id').AsString;
          OpName := DM.QryTemp1.FieldByName('operator_name').AsString;
          OpDateIn := fnGetServerDate;
          OpUserName := Trim(DM.qryTemp1.FieldByName('operator_user').AsString);

          LoginOK := True;

          IsMaster := (OpLevel = 'Master');

          FMain.suiStatusBar1.Panels[0].Text := 'Operator : ' + OpName;

          if (dm.QryTemp1.FieldByName('privilege_permission').AsString = '') then
            fnResetMenu(fnRC2Decrypt('1cld'), FMain.MainMenu1)
          else
            fnResetMenu(fnRC2Decrypt(dm.QryTemp1.FieldByName('privilege_permission').AsStri
ng), FMain.MainMenu1);

          fnLoadPrintSetting;

          OpAllowDiscTotal := fnCheckPermission('DISCOUNT TOTAL');
          OpAllowDiscItem := fnCheckPermission('DISCOUNT ITEM');
          OpAllowTax := fnCheckPermission('TAX');

          // update tanggal login operator
          try
            fnStartTransaction;
            if dm.QryTemp1.FieldByName('operator_logout').AsString <> 'Y' then
            begin
              if fnMessage('Anda belum Akses Keluar terakhir kali masuk
program.'#13'Apakah akan melanjutkan ' +
                'sesi sebelumnya ?', mCONFIRM) <> IDYES then
              begin
                OpFirstDate := fnGetServerDate;
                fnSQLAdd(dm.QryTemp2, 'UPDATE operator SET operator_last_login = ' +
fnInputDateDB(OpFirstDate) +
                  ', operator_logout = "N" WHERE operator_id = ' + QuotedStr(OpId), True);
                fnExecSQL(dm.QryTemp2);
              end
            end
            else
            begin
              OpFirstDate := fnGetServerDate;
              fnSQLAdd(dm.QryTemp2, 'UPDATE operator SET operator_last_login = ' +
fnInputDateDB(OpFirstDate) +
                ', operator_logout = "N" WHERE operator_id = ' + QuotedStr(OpId), True);
              fnExecSQL(dm.QryTemp2);
            end;

            // input saldo awal untuk kasir
            if UpperCase(OpLevel) = 'KASIR' then
            begin
              //ambil tanggal login operator yang terbaru
              fnSQLAdd(dm.QryTemp2, 'SELECT operator_last_login FROM operator ' +
                'WHERE operator_id = :paramId ', True);
              fnSQLParamByName(dm.QryTemp2, 'paramId', OpId);
              fnSQLOpen(dm.QryTemp2);

              LoginDate := dm.QryTemp2.FieldByName('operator_last_login').AsDateTime;

              fnSQLAdd(dm.QryTemp2, 'SELECT * FROM transaction_history WHERE
th_type = "First Cash" ' +
                'AND operator_id = :paramOpId AND th_status = "Login" ' +
                'AND th_date >= ' + fnInputDateDB(LoginDate), True);
              fnSQLParamByName(dm.QryTemp2, 'paramOpId', OpId);
              fnSQLOpen(dm.QryTemp2);

              OpFirstCash := 0;
              // kalau belum ada kas awal
              if dm.QryTemp2.Eof then
              begin
//              prCreateForm(TFFirstCash, FFirstCash, 'ShowModal');
//
//              OpFirstDate := fnGetServerDate();
//              OpCashId := fnGenerateId;
//              fnSQLAdd(dm.QryTemp2, 'INSERT INTO transaction_history (th_id, ' +
//                'operator_id, th_type, th_status, th_date, th_total) ' +
//                'VALUES (:paramId, :paramOpId, :paramType, ' +
//                ':paramStatus, ' + fnInputDateDB(OpFirstDate, False) +
//                ', :paramTotal)', True);
//              fnSQLParamByName(dm.QryTemp2, 'paramId', OpCashId);
//              fnSQLParamByName(dm.QryTemp2, 'paramOpId', OpId);
//              fnSQLParamByName(dm.QryTemp2, 'paramType', 'First Cash');
//              fnSQLParamByName(dm.QryTemp2, 'paramStatus', 'Login');
//              fnSQLParamByName(dm.QryTemp2, 'paramTotal', OpFirstCash);
//              fnExecSQL(dm.QryTemp2);
              end
              else
              begin
//              OpCashId := dm.QryTemp2.FieldByName('th_id').AsString;
//              OpFirstCash := dm.QryTemp2.FieldByName('th_total').AsCurrency;
//              OpFirstDate := dm.QryTemp2.FieldByName('th_date').AsDateTime
              end;
            end;

            fnCommit;
          except
            on E: Exception do fnCheckNetwork(E, Self)
          end;

          Close;
        end
        else
        begin
          inc(Count);
          if Count < 3 then
          begin
            if Count > 1 then
              fnMessage('Password masih salah !', mWARN)
            else
              fnMessage('Password salah !', mWARN);

            fnMessage('Password salah !'#13'Anda masih bisa mencoba ' + IntToStr(3 - Count)
+ ' !', mERROR);
          end
          else
          begin
            fnMessage('Password salah !'#13'Anda tidak berhak menggunakan applikasi ini !',
mERROR);
            Application.Terminate;
            Close
          end;
        end;
      end
      else
      begin
        inc(Count);
        if Count < 3 then
          fnMessage('Nama user tidak ditemukan.', mINFO)
        else
        begin
          fnMessage('Nama user tidak ada !'#13'Anda tidak berhak menggunakan applikasi
ini !', mERROR);
          Application.Terminate;
          Close
        end;
      end;
    end
    else
    begin
      fnMessage('Username tidak boleh kosong !', mWARN);
    end;
  except
  end;
end;

procedure TFLogin.suitempbtnLoginKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    VK_ESCAPE: Close;
  end;
end;

procedure TFLogin.FormCreate(Sender: TObject);
begin
  Label1.Font.Color := StringToColor(fnReadIniFiles('LOGIN', 'LABEL COLOR',
'$000000'));
  Label2.Font.Color := Label1.Font.Color;
  LoginOK := False;
//  try
//    Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'Login.JPG');
//  except
```

```
//    Image1.Picture := nil
//  end
end;

procedure TFLogin.Cleanup;
begin
  FreeAndNil(FLogin);
end;

procedure TFLogin.GantiWarna1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
  begin
    Label1.Font.Color := ColorDialog1.Color;
    Label2.Font.Color := ColorDialog1.Color;

    fnWriteIniFiles('LOGIN', 'LABEL COLOR', ColorToString(ColorDialog1.Color));
  end;
end;

end.
```

Uprivillege.pas

```
unit UPrivillege;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, BaseGrid, AdvGrid, DBAdvGrid;

type
  TFPrivillege = class(TForm)
    Label2: TLabel;
    StgMain: TAdvStringGrid;
    GrbMain: TGroupBox;
    GrbNav: TGroupBox;
    BtnClose: TButton;
    Button1: TButton,
    Button2: TButton,
    Button3: TButton,
    Button4: TButton,
    Button5: TButton,
    GrbSearch: TGroupBox;
    btnSearch: TButton;
    ESearch: TEdit;
    procedure Setting;
    procedure ClearGrid;
    procedure SearchData;
    procedure FormCreate(Sender: TObject);
    procedure suitempBtnCloseClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormResize(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure suitempbtnSearchClick(Sender: TObject);
    procedure suitempButton1Click(Sender: TObject);
    procedure suitempButton2Click(Sender: TObject);
    procedure suitempButton3Click(Sender: TObject);
    procedure suitempButton4Click(Sender: TObject);
    procedure suitempButton5Click(Sender: TObject);
    procedure suitempESearchKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure CmbLocChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FPrivillege: TFPrivillege;

implementation

uses UDM, UFunc, UMain, UPrivillegeAdd, UPermission;

{$R *.dfm}

const
  ColName = 1;
  ColNote = 2;
  ColId = 3;

procedure TFPrivillege.Setting;
begin
  try
    if windowState <> wsMaximized then
      GrbSearch.Top := 25
    else
      GrbSearch.Top := 10;

    GrbSearch.Width := ClientWidth - GrbSearch.Left - 10;

    GrbNav.Top := GrbSearch.Top + GrbSearch.Height + 10;
    GrbNav.Left := ClientWidth - GrbNav.Width - 10;
    GrbNav.Height := ClientHeight - GrbNav.Top - 10;

    GrbMain.Top := GrbSearch.Top + GrbSearch.Height + 10;
    GrbMain.Width := GrbNav.Left - GrbMain.Left - 10;

    GrbMain.Height := ClientHeight - GrbMain.Top - 10;

    StgMain.ColWidths[ColName] := Round(30 * StgMain.Width / 100);
    StgMain.ColWidths[ColNote] := Round(30 * StgMain.Width / 100);
    StgMain.ColWidths[ColId] := 0; //50 * (StgMain.Width div 100);
  except
  end
end;

procedure TFPrivillege.ClearGrid;
var i, j: integer;
begin
  for i := 0 to StgMain.ColCount - 1 do
    for j := 1 to StgMain.RowCount - 1 do
      StgMain.Cells[i, j] := '';

  StgMain.RowCount := 2;
end;

procedure TFPrivillege.SearchData;
var i: integer;
  KeyWord: string;
  LastRow: integer;
begin
  try
    LastRow := StgMain.Row;
    ClearGrid;

    KeyWord := Trim(ESearch.Text);
    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM privilege '+
      'WHERE (privilege_name LIKE ''%' + KeyWord + '%'' OR '+
        'privilege_note LIKE ''%' + KeyWord + '%'') ', True);

    fnSQLAdd(dm.QryTemp1, 'ORDER BY privilege_id');
    fnSQLOpen(dm.QryTemp1);

    if dm.QryTemp1.RecordCount > 1 then
      StgMain.RowCount := dm.QryTemp1.RecordCount+1
    else
      StgMain.RowCount := 2;

    i := 0;
    while not dm.QryTemp1.Eof do
    begin
      inc(i);

      StgMain.Cells[0, i] := Format('%d.', [i]);
      StgMain.Cells[ColName, i] :=
dm.QryTemp1.FieldByName('privilege_name').AsString;
      StgMain.Cells[ColNote, i] := dm.QryTemp1.FieldByName('privilege_note').AsString;
      StgMain.Cells[ColId, i] := dm.QryTemp1.FieldByName('privilege_id').AsString;

      if i >= StgMain.RowCount then
        StgMain.RowCount := StgMain.RowCount + 1;

      dm.QryTemp1.Next;
    end;

    if LastRow >= StgMain.RowCount then LastRow := StgMain.RowCount-1;
    StgMain.Row := LastRow;
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

procedure TFPrivillege.FormCreate(Sender: TObject);
begin
  WindowState := wsMaximized;
end;

procedure TFPrivillege.suitempBtnCloseClick(Sender: TObject);
begin
  Close
end;

procedure TFPrivillege.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FPrivillege := nil
end;

procedure TFPrivillege.FormResize(Sender: TObject);
begin
  Setting
end;

procedure TFPrivillege.FormActivate(Sender: TObject);
begin
  SearchData
end;

procedure TFPrivillege.suitempbtnSearchClick(Sender: TObject);
begin
  SearchData;
  if dm.QryTemp1.IsEmpty then
  begin
    fnMessage('Data tidak ditemukan.', mINFO);
  end
end;
```

```
procedure TFPrivillege.suitempButton1Click(Sender: TObject);
begin
  prCreateForm(TFPrivillegeAdd, FPrivillegeAdd, '');

  if Assigned(FPrivillegeAdd) then
  begin
    FPrivillegeAdd.PrivName := '';
    FPrivillegeAdd.FormType := 'Add';
    FPrivillegeAdd.ShowModal;
    SearchData;
  end
end;

procedure TFPrivillege.suitempButton2Click(Sender: TObject);
begin
  if StgMain.Cells[1, 1] = '' then
  begin
    fnMessage(NoDataToEdit, mINFO);
    Exit
  end;

  if Pos('Level Dasar', StgMain.Cells[2, StgMain.Row]) > 0 then
  begin
    fnMessage('Level Dasar tidak dapat diedit.', mWARN);
  end
  else
  begin
    prCreateForm(TFPrivillegeAdd, FPrivillegeAdd, '');

    if Assigned(FPrivillegeAdd) then
    begin
      FPrivillegeAdd.PrivName := StgMain.Cells[1, StgMain.Row];
      FPrivillegeAdd.FormType := 'Edit';
      FPrivillegeAdd.ShowModal;
      SearchData;
    end;
  end
end;

procedure TFPrivillege.suitempButton3Click(Sender: TObject);
begin
  if StgMain.Cells[1, 1] = '' then
  begin
    fnMessage(NoDataToDelete, mINFO);
    Exit;
  end;

  fnSQLADd(dm.QryTemp1, 'SELECT * FROM operator WHERE privilege_id
= :paramPrivId ', True);
  fnSQLParamByName(dm.QryTemp1, 'paramPrivId', StgMain.cells[ColId,
StgMain.Row]);
  fnSQLOpen(dm.QryTemp1);

  if not dm.QryTemp1.IsEmpty then
  begin
    fnMessage(DataCannotDelete, mWARN);
    Exit
  end;

  if Pos('Level Dasar', StgMain.Cells[ColNote, StgMain.Row]) > 0 then
  begin
    fnMessage('Level Dasar tidak dapat dihapus.', mWARN);
  end
  else
  begin
    if fnMessage('Anda akan menghapus Level Akses '''+StgMain.Cells[ColName,
StgMain.Row]+''' ?', mCONFIRM) = IDYES then
    begin
      try
        dm.DB.BeginTrans;

        fnSQLAdd(dm.QryTemp1, 'DELETE FROM privilege WHERE privilege_id
= :paramId', True);
        fnSQLParamByName(dm.QryTemp1, 'paramId', StgMain.Cells[ColId,
StgMain.Row]);
        fnExecSQL(dm.QryTemp1);

        dm.DB.CommitTrans;
      except
        on E: Exception do fnCheckNetwork(E, Self);
      end;

      SearchData;

      fnMessage(DataDeleted, mINFO);
    end
  end
end;

procedure TFPrivillege.suitempButton4Click(Sender: TObject);
begin
  prCreateForm(TFPermission, FPermission, '');

  if Assigned(FPermission) then
  begin
    FPermission.PrivId := StgMain.Cells[colId, StgMain.Row];
    FPermission.ShowModal;
  end
end;
```

```
procedure TFPrivillege.suitempButton5Click(Sender: TObject);
begin
  try
    DM.RvReport.SetParam('parKataKunci', ESearch.Text);
    fnLoadPrint('HKewenanganB', 'DvStg', StgMain, 1, StgMain.RowCount - 1);
    fnMessage(DataPrinted, mINFO);
  except
    on E: Exception do fnMessage(ErrorPrinting+#13'Bug: '+E.Message, mWARN);
  end
end;

procedure TFPrivillege.suitempESearchKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then
    btnSearch.Click;
end;

procedure TFPrivillege.CmbLocChange(Sender: TObject);
begin
  SearchData;
end;

end.
```

UprivillrgeAdd.pas

```
unit UPrivillegeAdd;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TFPrivillegeAdd = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    lblNote1: TLabel;
    lblNote2: TLabel;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    BtnCancel: TButton;
    BtnSave: TButton;
    procedure suitempBtnCancelClick(Sender: TObject);
    procedure suitempBtnSaveClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
    procedure suitempEdit1KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }
  public
    FormType: string;
    PrivName: string;
    { Public declarations }
  end;

var
  FPrivillegeAdd: TFPrivillegeAdd;

implementation

uses UDM, UFunc, UMain, UPrivillege;

{$R *.dfm}

procedure TFPrivillegeAdd.suitempBtnCancelClick(Sender: TObject);
begin
  Close
end;

procedure TFPrivillegeAdd.suitempBtnSaveClick(Sender: TObject);
var id: string;
begin
  try
    if (Edit1.Text = '') or (Edit2.Text = '') then
    begin
      fnMessage(DataNotComplete, mWARN);
      Exit;
    end;

    fnSQLAdd(dm.QryTemp1, 'SELECT privilege_id FROM privilege WHERE
privilege_name = :paramName', True);
    fnSQLParamByName(dm.QryTemp1, 'paramName', PrivName);
    fnSQLOpen(dm.qryTemp1);

    id := dm.QryTemp1.FieldByName('privilege_id').AsString;

    if FormType <> 'Edit' then
    begin
      if UpperCase(Edit1.Text) = 'MASTER' then
      begin
        fnMessage('Level Master tidak dapat ditambahkan karena digunakan untuk Pusat.',
mINFO);
        Exit
      end;
```

```
    fnSQLAdd(dm.QryTemp1, 'SELECT privilege_id FROM privilege WHERE
privilege_name = :paramName '+
      'AND privilege_id <> :paramId', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', id);
    fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit1.Text));
    fnSQLOpen(dm.qryTemp1);

    if not dm.QryTemp1.IsEmpty then
    begin
      fnMessage(DataAlreadyUsed, mWARN);
      Exit
    end;

    dm.DB.BeginTrans;

    fnSQLAdd(dm.QryTemp1, 'INSERT INTO privilege (privilege_id, '+
      'privilege_name, privilege_note) ' +
      'VALUES (:paramId, :paramName, :paramDesc)', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', fnGenerateId);
    fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit1.Text));
    fnSQLParamByName(dm.QryTemp1, 'paramDesc', Trim(Edit2.Text));
    fnExecSQL(dm.qryTemp1);

    dm.DB.CommitTrans;

    fnMessage(DataSaved, mINFO);
  end
  else
  begin
    fnSQLAdd(dm.QryTemp1, 'SELECT privilege_id FROM privilege WHERE
privilege_name = :paramName '+
      'AND privilege_id <> :paramId', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', id);
    fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit1.Text));
    fnSQLOpen(dm.qryTemp1);

    if not dm.QryTemp1.IsEmpty then
    begin
      fnMessage(DataAlreadyUsed, mWARN);
      Exit
    end;

    dm.DB.BeginTrans;

    fnSQLAdd(dm.QryTemp1, 'UPDATE privilege SET privilege_name = :paramName,
'+
      'privilege_note = :paramDesc ' +
      ' WHERE privilege_id = :paramId ', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', id);
    fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit1.Text));
    fnSQLParamByName(dm.QryTemp1, 'paramDesc', Trim(Edit2.Text));
    fnExecSQL(dm.qryTemp1);

    dm.DB.CommitTrans;

    fnMessage(DataEdited, mINFO);
  end;

  Close
 except
  on E: Exception do fnCheckNetwork(E, Self)
 end
end;

procedure TFPrivillegeAdd.FormClose(Sender: TObject;
 var Action: TCloseAction);
begin
 Action := caFree;
 FPrivillegeAdd := nil
end;

procedure TFPrivillegeAdd.FormActivate(Sender: TObject);
begin
 try
  if FormType = 'Edit' then
  begin
    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM privilege '+
      'WHERE privilege_name = :paramName ' +
      'ORDER BY privilege_name', True);
    fnSQLParamByName(dm.QryTemp1, 'paramName', PrivName);
    fnSQLOpen(dm.qryTemp1);

    Caption := 'Ubah Level';
    Edit1.Text := dm.QryTemp1.FieldByName('privilege_name').AsString;
    Edit2.Text := dm.QryTemp1.FieldByName('privilege_note').AsString;
    Edit1.SelectAll;
  end
 except
  on E: Exception do fnCheckNetwork(E, Self)
 end
end;

procedure TFPrivillegeAdd.suitempEdit1KeyDown(Sender: TObject, var Key: Word;
 Shift: TShiftState);
begin
 if Key = VK_RETURN then
  Perform(WM_NEXTDLGCTL, 0, 0);
end;

end
```

Upermission.pas

```
unit UPermission;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, htmltv, Menus, ADODB;

type
  TFPermission = class(TForm)
    HTMLTreeview1: THTMLTreeview;
    GrbMain: TGroupBox;
    GrbNav: TGroupBox;
    BtnClose: TButton;
    Button1: TButton,
    GrbSearch: TGroupBox;
    CheckBox2: TCheckBox;
    CbxDiscTotal: TCheckBox,
    CbxDiscItem: TCheckBox;
    CbxTax: TCheckBox;
    procedure CreateMenu(Item: TMenuItem, Parent: TTreeNode);
    function CheckPermission(Treeview: THTMLTreeview): string;
    procedure CheckMenu(Treeview: THTMLTreeview; StrPerm: string);

    procedure FormCreate(Sender: TObject);
    procedure suitempBtnCloseClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure suitempButton1Click(Sender: TObject);
    procedure HTMLTreeview1CheckBoxClick(Sender: TObject; Node: TTreeNode;
      Check: Boolean);
    procedure FormActivate(Sender: TObject);
  private
    Permission: array of byte;
    { Private declarations }
  public
    PrivId: string;
    LocId: string,
    { Public declarations }
  end;

var
  FPermission: TFPermission;

implementation

uses UMain, UFunc, UDM, UPrivillege;

{$R *.dfm}

procedure TFPermission.FormCreate(Sender: TObject);
var i: integer;
begin
  CreateMenu(FMain.MainMenu1.Items, nil);

  for i := 0 to HTMLTreeview1.Items.Count - 1 do
  begin
    HTMLTreeview1.Items[i].ImageIndex := 1;
  end;
end;

procedure TFPermission.suitempBtnCloseClick(Sender: TObject);
begin
  Close
end;

procedure TFPermission.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FPermission := nil
end;

procedure TFPermission.CreateMenu(Item: TMenuItem; Parent: TTreeNode);
var i: integer;
  TN: TTreeNode;
  S: string;
begin
  for i := 0 to Item.Count - 1 do
  begin
    if Item.Items[i].Caption <> '-' then
    begin
      s := StringReplace(Item.Items[i].Caption, '&', '', [rfReplaceAll, rfIgnoreCase]);

      TN := HTMLTreeview1.Items.AddChild(Parent, Item.Items[i].Caption);
      CreateMenu(Item.Items[i], TN);
    end
  end
end;

procedure TFPermission.suitempButton1Click(Sender: TObject);
  procedure fnSavePermission(IdxName: string; Value: char);
  begin
    fnSQLAdd(dm.Cmd1, 'INSERT INTO op_permission (permission_type, ' +
      'permission_value, privilege_id) VALUES (:paramName, ' +
      ':paramValue, :paramPrivId)');
    fnSQLParamByName(dm.Cmd1, 'paramName', IdxName);
    fnSQLParamByName(dm.Cmd1, 'paramValue', Value);
    fnSQLParamByName(dm.Cmd1, 'paramPrivId', PrivId);
    fnExecSQL(dm.Cmd1);
```

```
  end;

var s: string;
begin
  try
    s := fnRC2Encrypt(CheckPermission(HTMLTreeview1));

    fnStartTransaction;

    fnSQLAdd(dm.Cmd1, 'UPDATE privilege SET privilege_permission = :paramPerm '
+
      ' WHERE privilege_id = :paramPrivId');
    fnSQLParamByName(dm.Cmd1, 'paramPerm', s);
    fnSQLParamByName(dm.Cmd1, 'paramPrivId', PrivId);
    fnExecSQL(dm.Cmd1);

    //
============================================================
==========
    // hapus dulu yang di table op_permission coz langsung insert yang baru
    fnSQLAdd(dm.Cmd1, 'DELETE FROM op_permission WHERE privilege_id
= :paramPrivId ');
    fnSQLParamByName(dm.Cmd1, 'paramPrivId', PrivId);
    fnExecSQL(dm.Cmd1);

    OpAllowDiscTotal := False;
    OpAllowDiscItem := False;
    OpAllowTax := False;

    if CheckBox2.Checked then
    begin
      fnSavePermission('CASHIER REPORT', 'Y');
    end
    else fnSavePermission('CASHIER REPORT', 'N');

    if CbxDiscTotal.Checked then
    begin
      fnSavePermission('DISCOUNT TOTAL', 'Y');
      OpAllowDiscTotal  = True;
    end
    else fnSavePermission('DISCOUNT TOTAL', 'N');

    if CbxDiscTotal.Checked then
    begin
      fnSavePermission('DISCOUNT ITEM', 'Y'),
      OpAllowDiscItem := True
    end
    else fnSavePermission('DISCOUNT ITEM', 'N'),

    if CbxTax.Checked then
      begin
        fnSavePermission('TAX', 'Y');
        OpAllowTax := True
      end
      else fnSavePermission('TAX', 'N');
      //
============================================================
==========

    fnCommit;

    if OpLevel = FPrivillege.StgMain.Cells[1, FPrivillege.StgMain.Row] then
    begin
      fnResetMenu(fnRC2Decrypt(s), FMain.MainMenu1); // langsung diterapkan ke menu
utama
    end;

    fnMessage(DataSaved, mINFO);

    SetLength(Permission, 0); //delete array permission

    Close
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

function TFPermission.CheckPermission(Treeview: THTMLTreeview): string;
var i: integer;
  s: string;
begin
  s := '';
  for i := 0 to Treeview.Items.Count - 1 do
  begin
    s := s + IntToStr(Treeview.Items[i].ImageIndex);
  end;
  Result := s;
end;

procedure TFPermission.HTMLTreeview1CheckBoxClick(Sender: TObject;
  Node: TTreeNode; Check: Boolean);

  procedure CheckAll(Node: TTreeNode; ImgIdx: integer);
  var i: integer;
  begin
    Node.ImageIndex := ImgIdx;

    for i := 0 to Node.Count - 1 do
    begin
      Node[i].ImageIndex := ImgIdx;

      if Node[i].Count > 0 then
        CheckAll(Node[i], ImgIdx);
    end
  end;

//var i: integer;
begin
  if Node.ImageIndex = 1 then
    CheckAll(Node, 1)
  else
    CheckAll(Node, 2)
end;

procedure TFPermission.CheckMenu(Treeview: THTMLTreeview; StrPerm: string);
var i: integer;
begin
  SetLength(Permission, Length(StrPerm));
  for i := 1 to Length(StrPerm) do
  begin
    if (Copy(StrPerm, i, 1) = '2') then
      Permission[i - 1] := 2
    else
      Permission[i - 1] := 1
  end;

  for i := 0 to Treeview.Items.Count - 1 do
  begin
    if i >= Length(Permission) then
      Treeview.Items[i].ImageIndex := 1
    else
      Treeview.Items[i].ImageIndex := Permission[i]
  end
end;

procedure TFPermission.FormActivate(Sender: TObject);
  function fnCheckPermission(IdxName: string): boolean;
  var
    Qry: TADOQuery;
  begin
    result := false;
    Qry := TADOQuery.Create(nil);
    Qry.Connection := DM.DB;

    try
      fnSQLAdd(Qry,
        'SELECT * FROM op_permission ' +
        'WHERE privilege_id = ' + QuotedStr(PrivId) + ' AND permission_type = ' +
        QuotedStr(IdxName) , True);
      fnSQLOpen(Qry);
      if Qry.FieldByName('permission_value').AsString = 'Y' then result := true;
    except
      Result := False
    end;
    FreeAndNil(Qry);
  end;

var TempStr: string;
begin
  try
    CheckBox2.Checked := fnCheckPermission('CASHIER REPORT');
    CbxDiscTotal.Checked := fnCheckPermission('DISCOUNT TOTAL');
    CbxDiscItem.Checked := fnCheckPermission('DISCOUNT ITEM');
    CbxTax.Checked := fnCheckPermission('TAX');

    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM privilege WHERE privilege_id
= :paramId ', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', PrivId);
    fnSQLOpen(dm.QryTemp1);

    TempStr :=
fnRC2Decrypt(dm.QryTemp1.FieldByName('privilege_permission').AsString);

    if TempStr <> '' then
    begin
      CheckMenu(HTMLTreeview1, TempStr);
    end
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

end.
```

UCheckDB.pas

```
unit UCheckDB;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  TFCheckDB = class(TForm)
    Label1: TLabel;
    GroupBox1: TGroupBox;
    ProgressBar1: TProgressBar;
    Button1: TButton;
    Button2: TButton;
```

```
    CheckBox1: TCheckBox;
    procedure suitempButton2Click(Sender: TObject);
    procedure suitempButton1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FCheckDB: TFCheckDB;

implementation

uses UDM, UFunc, UMain;

{$R *.dfm}

procedure TFCheckDB.suitempButton2Click(Sender: TObject);
begin
  Close
end;

procedure TFCheckDB.suitempButton1Click(Sender: TObject);
var
  TableList: TStringList;
  IndexExists: Boolean;
  IndexName: string;
  TableName: string;
  i: Integer;
begin
  ProgressBar1.Position := 0;

  // force dimaksudkan untuk menghapus index terlebih dahulu, karena kadang index
  // membuat ALTER TABLE tidak dapat dilakukan
  if CheckBox1.Checked then
  begin
    if ConnectionType <> scSQLServer then

    fnMessage('Untuk database ini belum mendukung fasilitas Force Check.', mWARN);
    TableList := TStringList.Create;
    DM.DB.GetTableNames(TableList);
    ProgressBar1.Max := TableList.Count;

    for i := 0 to TableList.Count - 1 do
    begin
      TableName := TableList[i];
      IndexName := 'IDX_' + TableName;
//      Label2.Caption := 'Force checking: ' + TableName + ' - IndexName + ' ...';
      Application.ProcessMessages;
      case ConnectionType of
        scSQLServer:
          begin
            fnSQLAdd(DM.QryTemp1, 'SELECT name FROM sysindexes WHERE name = '
+ QuotedStr(IndexName), True);
            fnSQLOpen(DM.QryTemp1);

            if not DM.QryTemp1.IsEmpty then
            begin
              fnSQLAdd(DM.Cmd1, Format('DROP INDEX %s.%s', [TableName,
IndexName]));
              fnExecSQL(DM.Cmd1);
            end;
          end;
      end;
      ProgressBar1.StepIt;
    end;
  end;

  TableList := TStringList.Create;
  DM.DB.GetTableNames(TableList);

  ProgressBar1.Position := 0;
  if dm.fnCheckDBStructure(ProgressBar1) then
    fnMessage('Struktur database telah selesai diperbarui.', mINFO)
  else
    fnMessage('Struktur database gagal diperbarui.', mERROR);
  ProgressBar1.Position := ProgressBar1.Max;
end;

procedure TFCheckDB.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FCheckDB := nil
end;

end.
```

UOperator.pas

```
unit UOperator;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, BaseGrid, AdvGrid, DBAdvGrid;

type
```

```
  TFOperator = class(TForm)
    Label2: TLabel;
    Label1: TLabel;
    StgMain: TAdvStringGrid;
    GrbMain: TGroupBox;
    GrbNav: TGroupBox;
    BtnClose: TButton;
    Button1: TButton,
    Button2: TButton,
    Button3: TButton,
    Button4: TButton,
    GrbSearch: TGroupBox;
    btnSearch: TButton;
    ESearch: TEdit;
    ComboBox1: TComboBox;
    procedure Setting;
    procedure SearchData;
    procedure FillLevel;

    procedure FormCreate(Sender: TObject);
    procedure suitempBtnCloseClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormResize(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure suitempbtnSearchClick(Sender: TObject);
    procedure suitempButton1Click(Sender: TObject);
    procedure suitempButton2Click(Sender: TObject);
    procedure suitempButton3Click(Sender: TObject);
    procedure suitempComboBox1KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure suitempESearchKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure suitempButton4Click(Sender: TObject);
    procedure suitempGrbMainResize(Sender: TObject);
    procedure CmbLocChange(Sender: TObject);
    procedure suitempComboBox1Change(Sender: TObject);
  private
    procedure EnableButton;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FOperator: TFOperator;

implementation

uses UDM, UFunc, UMain, UOperatorAdd;
```

```
{$R *.dfm}

const
  ColNum = 0;
  ColCode = 1;
  ColName = 2;
  ColLevel = 3;
  ColId = 4;

procedure TFOperator.Setting;
begin
  try
    if windowState <> wsMaximized then
      GrbSearch.Top := 25
    else
      GrbSearch.Top := 10;

    GrbSearch.Width := ClientWidth - GrbSearch.Left - 10;

    GrbNav.Top := GrbSearch.Top + GrbSearch.Height + 10;
    GrbNav.Left := ClientWidth - GrbNav.Width - 10;
    GrbNav.Height := ClientHeight - GrbNav.Top - 10;

    GrbMain.Top := GrbSearch.Top + GrbSearch.Height + 10;
    GrbMain.Width := GrbNav.Left - GrbMain.Left - 10;
    GrbMain.Height := ClientHeight - GrbMain.Top - 10;

    StgMain.ColWidths[ColCode] := Round(20 * StgMain.Width / 100);
    StgMain.ColWidths[ColName] := Round(20 * StgMain.Width / 100);
    StgMain.ColWidths[ColLevel] := Round(30 * StgMain.Width / 100);
    StgMain.ColWidths[ColId] := 0;
  except
  end
end;

procedure TFOperator.SearchData;
var i: integer;
  KeyWord: string;
  LastRow: integer;
begin
  try
    LastRow := StgMain.Row;
    EnableButton;
    fnClearGrid(StgMain);

    KeyWord := Trim(ESearch.Text);
    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM operator ' +
      'LEFT JOIN privilege ON privilege.privilege_id = operator.privilege_id '#13 +
      'WHERE (operator_name LIKE "%' + KeyWord + '%" '#13 +
```

```
    'OR operator_user LIKE "%' + KeyWord + '%" OR privilege_name LIKE "%' +
KeyWord + '%")', True);

   if ComboBox1.Text <> 'All' then
     fnSQLAdd(dm.QryTemp1, 'AND privilege_name = ' +
QuotedStr(ComboBox1.Text));

   fnSQLAdd(dm.QryTemp1, ORDER BY operator_name');
   fnSQLOpen(dm.QryTemp1);

   if dm.QryTemp1.RecordCount > 1 then
    StgMain.RowCount := dm.QryTemp1.RecordCount + 1
   else
    StgMain.RowCount := 2;

   for i := 1 to dm.QryTemp1.RecordCount do
   begin
    StgMain.Cells[ColNum, i] := Format('%d.', [i]);
    StgMain.Cells[ColCode, i] := dm.QryTemp1.FieldByName('operator_user').AsString;
    StgMain.Cells[ColName, i] :=
dm.QryTemp1.FieldByName('operator_name').AsString;
    StgMain.Cells[ColLevel, i] :=
dm.QryTemp1.FieldByName('privilege_name').AsString;
    StgMain.Cells[ColId, i] := dm.QryTemp1.FieldByName('operator_id').AsString;

    dm.QryTemp1.Next;
   end;

   if LastRow >= StgMain.RowCount then LastRow := StgMain.RowCount-1;
   StgMain.Row := LastRow;
  except
   on E: Exception do fnCheckNetwork(E, Self);
  end
end;

procedure TFOperator.FormCreate(Sender: TObject);
begin
  WindowState := wsMaximized;
  FillLevel;
end;

procedure TFOperator.suitempBtnCloseClick(Sender: TObject);
begin
  Close
end;

procedure TFOperator.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FOperator := nil
end;

procedure TFOperator.FormResize(Sender: TObject);
begin
  Setting
end;

procedure TFOperator.FormActivate(Sender: TObject);
begin
  Setting;
  SearchData
end;

procedure TFOperator.FillLevel;
begin
  try
    ComboBox1.Items.Clear;
    ComboBox1.Items.Add('All');

    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM privilege '+
     ' ORDER BY privilege_name', True);
    fnSQLOpen(dm.QryTemp1);

    while not dm.QryTemp1.Eof do
    begin
     ComboBox1.Items.Add(dm.QryTemp1.FieldByName('privilege_name').AsString);

     dm.QryTemp1.Next;
    end;

    ComboBox1.ItemIndex := 0
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

procedure TFOperator.suitempbtnSearchClick(Sender: TObject);
begin
  SearchData;
  if dm.QryTemp1.IsEmpty then
  begin
    fnMessage(SearchNotFound, mINFO);
  end
end;

procedure TFOperator.suitempButton1Click(Sender: TObject);
begin
  prCreateForm(TFOperatorAdd, FOperatorAdd, '');

  if Assigned(FOperatorAdd) then
  begin
    FOperatorAdd.FormType := 'Add';
    FOperatorAdd.ShowModal;
    SearchData;
  end
end;

procedure TFOperator.suitempButton2Click(Sender: TObject);
begin
  if StgMain.Cells[ColCode, 1] = '' then
  begin
    fnMessage(NoDataToEdit, mINFO);
    Exit
  end;

  prCreateForm(TFOperatorAdd, FOperatorAdd, '');

  if Assigned(FOperatorAdd) then
  begin
    FOperatorAdd.OperatorId := StgMain.Cells[ColId, StgMain.Row];
    FOperatorAdd.FormType := 'Edit';
    FOperatorAdd.ShowModal;
    SearchData;
  end;
end;

procedure TFOperator.suitempButton3Click(Sender: TObject);
begin
  if StgMain.Cells[ColCode, 1] = '' then
  begin
    fnMessage(NoDataToDelete, mWARN);
    Exit;
  end;

  if UpperCase(StgMain.Cells[ColCode, StgMain.Row]) = 'MASTER' then
  begin
    fnMessage('Operator "Master" tidak dapat dihapus.', mWARN);
    Exit;
  end;

  fnSQLADd(dm.QryTemp1, 'SELECT * FROM transaction_history WHERE
operator_id = :paramOpId ', True);
  fnSQLParamByName(dm.QryTemp1, 'paramOpId', StgMain.cells[ColId,
StgMain.Row]);
  fnSQLOpen(dm.QryTemp1);

  if not dm.QryTemp1.IsEmpty then
  begin
    fnMessage(DataCannotDelete, mERROR);
    Exit
  end;

  if fnMessage('Anda yakin akan menghapus Operator '"+StgMain.Cells[ColName,
StgMain.Row]+'" ?', mCONFIRM) = IDYES then
  begin
    dm.DB.BeginTrans;

    fnSQLAdd(dm.QryTemp1, 'DELETE FROM operator WHERE operator_id
= :paramId', True);
    fnSQLParamByName(dm.QryTemp1, 'paramId', StgMain.Cells[ColId, StgMain.Row]);
    fnExecSQL(dm.QryTemp1);

    dm.DB.CommitTrans;

    SearchData;

    fnMessage(DataDeleted, mINFO);
  end
end;

procedure TFOperator.suitempComboBox1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then
    Perform(WM_NEXTDLGCTL, 0, 0);
end;

procedure TFOperator.suitempESearchKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then
    btnSearch.Click;
end;

procedure TFOperator.suitempButton4Click(Sender: TObject);
begin
  try
    DM.RvReport.SetParam('parLevel', ComboBox1.Text);
    DM.RvReport.SetParam('parKataKunci', ESearch.Text);
    fnLoadPrint('HOperatorB', 'DvStg', StgMain, 1, StgMain.RowCount - 1);
    fnMessage(DataPrinted, mINFO);
  except
    on E: Exception do fnMessage(ErrorPrinting+#13'Bug: '+E.Message, mWARN);
  end
end;

procedure TFOperator.suitempGrbMainResize(Sender: TObject);
begin
  Setting
end;
```

```
procedure TFOperator.CmbLocChange(Sender: TObject);
begin
  FillLevel;
  SearchData;
end;

procedure TFOperator.EnableButton;
begin
end;

procedure TFOperator.suitempComboBox1Change(Sender: TObject);
begin
  SearchData;
  if DM.QryTemp1.IsEmpty then
    fnMessage('Data tidak ditemukan', mINFO);
end;

end.
```

UOperatorAdd.pas

```
unit UOperatorAdd;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DBCtrls;

type
  TFOperatorAdd = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    lblNote1: TLabel;
    lblNote2: TLabel;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    CmbLevel: TComboBox;
    BtnCancel: TButton;
    BtnSave: TButton;
    procedure suitempBtnCancelClick(Sender: TObject);
    procedure suitempBtnSaveClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
    procedure suitempEdit1KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormResize(Sender: TObject);
  private
    procedure Setting;
    { Private declarations }
  public
    FormType: string;
    OperatorId: string;
    { Public declarations }
  end;

var
  FOperatorAdd: TFOperatorAdd;

implementation

uses UDM, UFunc, UMain, UOperator;

{$R *.dfm}

procedure TFOperatorAdd.Setting;
begin
  try
    GroupBox1.Top := 5;
    BtnSave.Top := GroupBox1.Top + GroupBox1.Height + 5;
    BtnCancel.Top := BtnSave.Top;
    LblNote1.Top := btnSave.Top + BtnSave.Height + 5;
    lblNote2.Top := lblNote1.Top + lblNote1.Height + 5;
    ClientHeight := lblNote2.Top + lblNote2.Height + GroupBox1.Left;
    ClientWidth := (GroupBox1.Left * 2) + GroupBox1.Width
  except
  end
end;

procedure TFOperatorAdd.suitempBtnCancelClick(Sender: TObject);
begin
  Close
end;

procedure TFOperatorAdd.suitempBtnSaveClick(Sender: TObject);
var PrivId: string;
begin
  try
    PrivId := fnFindValue('privilege', 'privilege_id', 'privilege_name', CmbLevel.Text);

    if (PrivId = '') or (Edit1.Text = '') or (Edit2.Text = '')
      or (Edit3.Text = '') or (CmbLevel.Text = '') then
    begin
      fnMessage(DataNotComplete, mWARN);
      Exit;
    end;

    if FormType <> 'Edit' then
    begin
      fnSQLAdd(dm.QryTemp1, 'SELECT * FROM operator WHERE ' +
        'operator_user = :paramUserName '+
        'AND operator_id <> :paramId', True);
      fnSQLParamByName(dm.QryTemp1, 'paramId', Trim(OperatorId));
      fnSQLParamByName(dm.QryTemp1, 'paramUserName', Trim(Edit1.Text));
      fnSQLOpen(dm.QryTemp1);

      if not dm.QryTemp1.IsEmpty then
      begin
        fnMessage(DataAlreadyUsed, mWARN);
        Exit;
      end;

      fnStartTransaction;

      fnSQLAdd(dm.QryTemp1, 'INSERT INTO operator (operator_id, '+
        'operator_user, operator_name, operator_pass, privilege_id) ' +
        'VALUES ( paramId, :paramUserName, :paramName, :paramPass, '+
        ':paramPrivId)', True);
      fnSQLParamByName(dm.QryTemp1, 'paramId', fnGenerateId);
      fnSQLParamByName(dm.QryTemp1, 'paramUserName', Trim(Edit1.Text));
      fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit2.Text));
      fnSQLParamByName(dm.QryTemp1, 'paramPass', fnRC2Encrypt(Trim(Edit3.Text)));
      fnSQLParamByName(dm.QryTemp1, 'paramPrivId', PrivId);
      fnExecSQL(dm.qryTemp1);

      fnCommit;

      fnMessage(DataSaved, mINFO);
    end
    else
    begin
      fnSQLAdd(dm.QryTemp1, 'SELECT * FROM operator WHERE ' +
        'operator_user = :paramUserName '+
        'AND operator_id <> :paramId', True);
      fnSQLParamByName(dm.QryTemp1, 'paramId', Trim(OperatorId));
      fnSQLParamByName(dm.QryTemp1, 'paramUserName', Trim(Edit1.Text));
      fnSQLOpen(dm.QryTemp1);

      if not dm.QryTemp1.IsEmpty then
      begin
        fnMessage(DataAlreadyUsed, mWARN);
        Exit;
      end;

      fnStartTransaction;

      fnSQLAdd(dm.QryTemp1, 'UPDATE operator SET operator_name = :paramName,
operator_user = :paramUserName, ' +
        'operator_pass = :paramPass, privilege_id = :paramPrivId '+
        ' WHERE operator_id = :paramId', True);
      fnSQLParamByName(dm.QryTemp1, 'paramId', OperatorId);
      fnSQLParamByName(dm.QryTemp1, 'paramName', Trim(Edit2.Text));
      fnSQLParamByName(dm.QryTemp1, 'paramUserName', Trim(Edit1.Text));
      fnSQLParamByName(dm.QryTemp1, 'paramPass', fnRC2Encrypt(Trim(Edit3.Text)));
      fnSQLParamByName(dm.QryTemp1, 'paramPrivId', PrivId);
      fnExecSQL(dm.qryTemp1);

      fnCommit;

      fnMessage(DataEdited, mINFO);
    end;

    Close
  except
    on E: Exception do fnCheckNetwork(E, Self)
  end
end;

procedure TFOperatorAdd.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
  FOperatorAdd := nil
end;

procedure TFOperatorAdd.FormActivate(Sender: TObject);
begin
  try
    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM privilege '+
      ' ORDER BY privilege_name', True);
    fnSQLOpen(dm.QryTemp1);

    CmbLevel.Items.Clear;
    while not dm.QryTemp1.Eof do
    begin
      CmbLevel.Items.Add(dm.QryTemp1.FieldByName('privilege_name').AsString);
      dm.QryTemp1.Next;
    end;
    CmbLevel.ItemIndex := 0;

    if FormType = 'Edit' then
    begin
      fnSQLAdd(dm.QryTemp1, 'SELECT * FROM operator ' +
        'LEFT JOIN privilege ON privilege.privilege_id = operator.privilege_id ' +
        'WHERE operator_id = :paramId ', True);
      fnSQLParamByName(dm.QryTemp1, 'paramId', OperatorId);
```

```
    fnSQLOpen(dm.qryTemp1);

    Edit1.Text := dm.QryTemp1.FieldByName('operator_user').AsString;
    Edit2.Text := dm.QryTemp1.FieldByName('operator_name').AsString;
    Edit3.Text := '';

    Caption := 'Ubah Operator';
    CmbLevel.ItemIndex :=
CmbLevel.Items.IndexOf(dm.QryTemp1.FieldByName('privilege_name').AsString);

    if dm.QryTemp1.FieldByName('operator_user').AsString = 'master' then
    begin
      CmbLevel.Enabled := False;
      Edit1.Enabled := False;
      Edit2.Enabled := False;
      Edit3.SelectAll;
      Edit3.SetFocus
    end;
  end
  except
    on E: Exception do fnCheckNetwork(E, Self)
  end
end;

procedure TFOperatorAdd.suitempEdit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then
    Perform(WM_NEXTDLGCTL, 0, 0);
end;

procedure TFOperatorAdd.FormResize(Sender: TObject);
begin
  Setting
end;

end.
```

Ucompany.pas

```
unit UCompany;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TFCompany = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    lblNote1: TLabel;
    lblNote2: TLabel;
    BtnCancel: TButton;
    BtnNext: TButton;
    GroupBox1: TGroupBox;
    EName: TEdit;
    EAddress: TEdit;
    ETelp: TEdit;
    EContact: TEdit;
    ENpwp: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure suitempENameKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure suitempBtnCancelClick(Sender: TObject);
    procedure suitempBtnNextClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    Save: boolean;
    { Private declarations }
  public
    Canceled: boolean;
    procedure CleanUp;
    { Public declarations }
  end;

var
  FCompany: TFCompany;

implementation

uses UDM, UFunc, UMain;

{$R *.dfm}

procedure TFCompany.FormCreate(Sender: TObject);
begin
  Canceled := False;
  Save := False;
  EName.Clear;
  EAddress.Clear;
  ETelp.Clear;
  EContact.Clear;
  try
    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM company', True);
    fnSQLOpen(dm.QryTemp1);

    if not dm.QryTemp1.IsEmpty then
    begin
      EName.Text := dm.QryTemp1.FieldByName('company_name').AsString;
      EAddress.Text := dm.QryTemp1.FieldByName('company_addr').AsString;
      ETelp.Text := dm.QryTemp1.FieldByName('company_telp').AsString;
      EContact.Text := dm.QryTemp1.FieldByName('company_contact').AsString;
      ENpwp.Text := dm.QryTemp1.FieldByName('company_npwp').AsString;

      BtnNext.Caption := 'Simpan';
    end
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

procedure TFCompany.suitempENameKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_RETURN then
    Perform(WM_NEXTDLGCTL, 0, 0);
end;

procedure TFCompany.suitempBtnCancelClick(Sender: TObject);
begin
//  if fnMessage('Apakah Anda yakin ingin keluar aplikasi ?', mCONFIRM) = IDYES
then
//  begin
//    Canceled := True;
//    Close
//  end
//  else
//  begin
//    Canceled := True;
//    Close
//  end;
  Canceled := True;
  Close;
end;

procedure TFCompany.suitempBtnNextClick(Sender: TObject);
var id: string;
begin
  Save := False;
  try
    if (EName.Text = '') or (EAddress.Text = '') then
    begin
      fnMessage(DataNotComplete, mWARN);
      Save := False;
      Exit;
    end;

    fnSQLAdd(dm.QryTemp1, 'SELECT * FROM company ', True);
    fnSQLOpen(dm.QryTemp1);

    if dm.QryTemp1.IsEmpty then
    begin
      fnStartTransaction;

      fnSQLAdd(dm.Cmd1, 'INSERT INTO company (company_name, ' +
        'company_addr, company_npwp, company_telp, company_contact) ' +
        'VALUES
(:paramName, :paramAddress, :paramNPWP, :paramTelp, :paramContact)');
      fnSQLParamByName(dm.Cmd1, 'paramName', Trim(EName.Text));
      fnSQLParamByName(dm.Cmd1, 'paramAddress', Trim(EAddress.Text));
      fnSQLParamByName(dm.Cmd1, 'paramNPWP', Trim(ENpwp.Text));
      fnSQLParamByName(dm.Cmd1, 'paramTelp', Trim(ETelp.Text));
      fnSQLParamByName(dm.Cmd1, 'paramContact', Trim(EContact.Text));
      fnExecSQL(dm.Cmd1);

      Save := True;
      fnCommit;
    end
    else
    begin
      id := dm.QryTemp1.FieldByName('company_name').AsString;

      fnStartTransaction;

      fnSQLAdd(dm.Cmd1, 'UPDATE company SET company_name = :paramName, ' +
        'company_addr = :paramAddress, company_npwp = :paramNpwp, company_telp
= :paramTelp, '+
        'company_contact = :paramContact WHERE company_name = :paramId');
      fnSQLParamByName(dm.Cmd1, 'paramId', id);
      fnSQLParamByName(dm.Cmd1, 'paramName', Trim(EName.Text));
      fnSQLParamByName(dm.Cmd1, 'paramAddress', Trim(EAddress.Text));
      fnSQLParamByName(dm.Cmd1, 'paramNpwp', Trim(ENpwp.Text));
      fnSQLParamByName(dm.Cmd1, 'paramTelp', Trim(ETelp.Text));
      fnSQLParamByName(dm.Cmd1, 'paramContact', Trim(EContact.Text));
      fnExecSQL(dm.Cmd1);

      Save := True;
      fnCommit;

      fnMessage('Profil Perusahaan telah disimpan.', mINFO)
    end;
```

```
//    if Save then
//    begin
//      ModalResult := mrOK;
//      Close
//    end
//    else EName.SetFocus
  except
    on E: Exception do fnCheckNetwork(E, Self);
  end
end;

procedure TFCompany.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if (not Save) and (not Canceled) then
    CanClose := False
  else
    if Canceled then
      ModalResult := mrCancel
    else
      ModalResult := mrOK
end;

procedure TFCompany.CleanUp;
begin
  FreeAndNil(FCompany);
end;

procedure TFCompany.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FCompany := nil
end;

end.
```