BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan kerja magang di PT Lemo Utama, penulis ditempatkan di departemen IT dan diberikan tugas utama sebagai *software developer.* Namun sebagai sebuah departemen baru, tidak jarang penulis juga membantu pekerjaan-pekerjaan dalam menjalankan fungsi departemen sebagai *IT Support*. Departemen ini berada di bawah pengawasan langsung Darmawan Sukmadjaja selaku *General Manager* tanpa ada manajer khusus departemen IT. Selama proses kerja magang berlangsung, penulis diberi pelatihan dasar mengenai informasi dan struktur *database* perusahaan serta diarahkan oleh pembimbing lapangan akan aplikasi yang harus dirancang. Beliau menjabarkan permasalahan yang ada kepada penulis, membantu analisa permasalahan dan berdiskusi akan solusi yang harus diwujudkan penulis dalam bentuk aplikasi. Penulis juga bekerja bersama Andri Setiawan yang bertugas sebagai *IT Support*.

3.2 Tugas yang Dilakukan

Selama melakukan kerja magang di PT Lemo Utama, penulis mendapatkan tugas-tugas termasuk didalamnya merancang dan membuat aplikasi *Order Information Tracking*. Adapun rincian akan tugas yang dilakukan penulis selama masa magang dua bulan adalah,

1. Mempelajari sistem kerja perusahaan, jaringan dan struktur *database* perusahaan,
2. Mempelajari pembuatan aplikasi berbasis *database* MySQL,
3. Melakukan perbaikan jaringan berupa penggantian *router* dan kabel UTP yang rusak atau usang,
4. Memasang dan memperbaharui *antivirus* di seluruh komputer perusahaan serta melakukan pemulihan terhadap data-data yang terinfeksi virus.

5. Membuat, mengimplementasi dan menguji aplikasi sederhana untuk mencetak surat tagihan pelanggan per order,
6. Membuat, mengimplementasi dan menguji aplikasi sederhana untuk mencetak surat tagihan pelanggan per kelompok order,
7. Melakukan konfigurasi untuk *printer sharing*,
8. Membuat dan mengimplementasi aplikasi *Order Information Tracking*.

Pada minggu-minggu pertama pelaksanaan kerja magang, penulis diminta mempelajari pembuatan aplikasi berbasis *database* dengan koneksi ODBC yang dapat menghasilkan *Excel file*. Selama proses belajar tersebut, penulis juga diberikan tugas-tugas sederhana yang berkaitan dengan jaringan dan fungsi *IT Support*. Penulis kemudian diberikan tugas untuk membuat aplikasi untuk melihat daftar order ekspor dan impor yang tersimpan di *database*, menentukan prioritas order import, dan realisasi order serta menghasilkan *Excel file* yang memuat data-data tersebut. Penulis secara berkala setiap harinya melaporkan aktifitas dan perkembangan tugas yang dilakukan kepada pembimbing lapangan. Sementara pada hari Jumat untuk setiap minggunya penulis merangkum dan mengulas kembali pekerjaan yang telah dilakukan pada minggu tersebut.

Dalam laporan kerja magang ini, pembahasan difokuskan pada pembuatan aplikasi *Order Information Tracking* karena memiliki bobot terbesar dimana merupakan tugas utama yang diberikan pembimbing lapangan kepada penulis.

3.3 Uraian Pelaksanaan Kerja Magang

Secara garis besar pelaksanaan kerja magang ini dapat dikelompokkan menjadi beberapa pokok bahasan yaitu perancangan dan pembangunan aplikasi termasuk pengujian dan implementasi aplikasi, masalah yang ditemukan, dan solusi yang diberikan terhadap masalah.

3.3.1 Perancangan dan Pembangunan Aplikasi

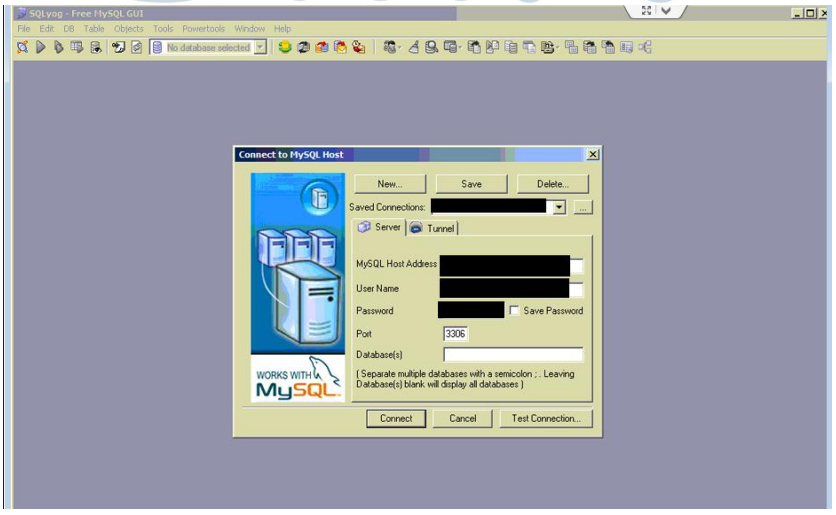
Dalam proses perancangan aplikasi *Order Information Tracking*, pertama-tama penulis harus mempelajari dasar-dasar MySQL, *data manipulation language*

dan *data definition language*, cara membangun koneksi antara aplikasi dan *database* dengan ODBC, memahami fungsi-fungsi *tools* dan *Excel object library* pada Visual Basic serta struktur *database* dan alur kerja sistem di PT Lemo Utama sebelum benar-benar dapat membangun aplikasi *Order Information Tracking* yang diinginkan. Dalam proses merancang dan membangun aplikasi tersebut terdapat kendala-kendala yang kemudian akan dibahas pada sub bab selanjutnya.

3.3.1.1 *Database* dan MySQL

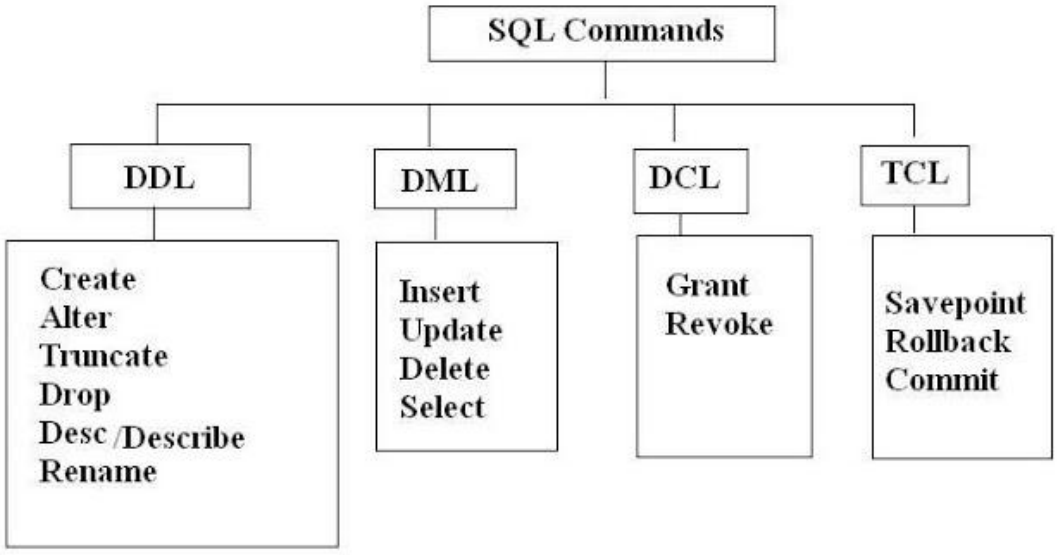
PT Lemo Utama menggunakan MySQL sebagai Database Management System (DBMS) di perusahaan tersebut. MySQL merupakan *open source relational* DBMS yang berbasis SQL. Dalam *database* dikenal banyak permodelan seperti *hierarchical model*, *network model*, *object model*, dan *relational database*. Model relasi merupakan permodelan yang paling sering digunakan dalam *database*. Model ini menggunakan sekumpulan tabel berdimensi dua yang disebut sebagai relasi dimana tiap relasi tersusun atas baris dan atribut.

MySQL tidak memiliki GUI sehingga biasanya diakses melalui command line namun terdapat aplikasi "*front end*" yang memberikan fitur management MySQL dengan GUI. Selama proses magang, penulis menggunakan aplikasi SQLyog untuk memudahkan proses pembuatan *query* dan pembelajaran struktur *database* perusahaan.



Gambar 3.1 SQLyog

Dalam SQL terdapat bahasa-bahasa instruksi yang digunakan untuk membuat dan menghapus *database*, menambah, menghapus, memanipulasi data, dan menampilkan data dari *database*. Bahasa instruksi ini dikelompokkan menjadi beberapa sub bahasa yaitu *data definition language*, *data manipulation language*, *data control language*, dan *transaction control language* seperti dijelaskan dibawah ini.



Gambar 3.2 Pengelompokan SQL Command

A. Data Definition Language

Data Definition Language (DDL) merupakan bahasa dalam *database management system* yang digunakan untuk mendefinisikan objek-objek ke dalam *database* baik menambah, menghapus atau memodifikasi objek tersebut. Objek berbeda dengan data informasi yang tersimpan dalam *database*.

B. Data Manipulation Language

Berbeda dengan DDL, *Data Manipulation Language* (DML) merupakan bahasa yang digunakan untuk memanipulasi data dalam tabel di *database* yang dipilih. Bentuk manipulasi yang dapat dilakukan seperti menambahkan sebaris data kedalam tabel, mengganti data pada atribut

tertentu di baris tertentu, menghapus baris dan menampilkan data dari tabel atau gabungan beberapa tabel.

Bahasa yang umum digunakan adalah DDL dan DML khususnya. Adapun *data control language* (DCL) digunakan untuk melakukan kontrol terhadap data dan *database* seperti pemberian dan pembatasan akses. Sementara itu, *transaction control language* (TCL) digunakan untuk mengatur proses transaksi yang terjadi di *database* seperti menyimpan, mengembalikan/membatalkan perubahan yang terjadi di *database*.

Dalam pembuatan aplikasi *Order Information Tracking*, penulis hanya menggunakan DML. Penulis tidak menggunakan DDL dikarenakan aplikasi ini hanya mengambil data yang sudah ada dan tersimpan dalam tabel tabel di *database* perusahaan. Perusahaan sudah memiliki aplikasi utama dan sistem yang berjalan dimana data-data dari sistem tersebut disimpan dalam *database* perusahaan. Penulis kemudian hanya perlu mengambil, mengolah data dan menampilkan informasi yang dibutuhkan melalui aplikasi *Order Information Tracking*.

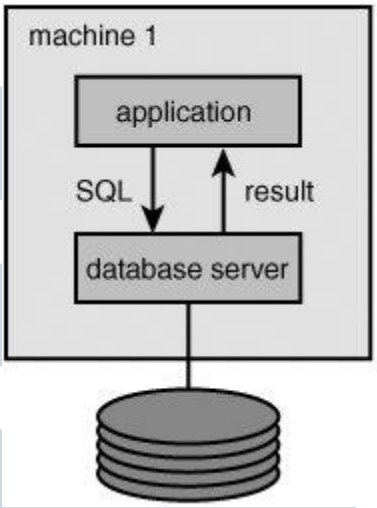
3.3.1.2 Model Hubungan Aplikasi dan *Database Server*

Dalam perancangan aplikasi, penulis mempelajari beberapa model arsitektur yang digunakan dalam relasi antara aplikasi dengan *database server*. Adapun arsitektur tersebut antara lain adalah Monolithic Architecture, *Client/Server Architecture* dan *Internet architecture*. Penulis menerapkan C*lient/Server Architecture* pada aplikasi *Order Information Tracking* karena sangat sesuai dengan peruntukan aplikasi itu sendiri. Penjelasan dari masing-masing arsitektur adalah sebagai berikut.

A. Monolithic Architecture

Arsitektur ini merupakan model arsitektur yang paling sederhana. Antara aplikasi dan *database* berjalan dalam satu mesin yang sama (Gambar 3.3), mungkin dalam sebuah mainframe ataupun PC. Aplikasi

akan memberikan SQL *statement/query* kepada *database server* untuk kemudian diinterpretasikan dan dijalankan. Hasil dari *query* tersebut kemudian dikirimkan kembali ke aplikasi.



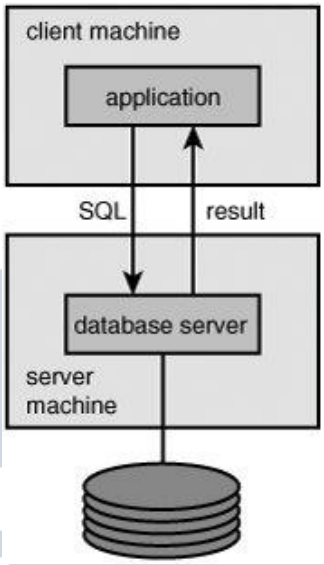
Gambar 3.3 *Monolithic Architecture*

Model ini tidak dapat diadopsi pada aplikasi *Order Information Tracking* karena aplikasi ditujukan untuk digunakan oleh banyak pihak dengan *database* terpusat sehingga aplikasi tidak bisa berada dalam satu mesin bersama dengan *database*.

B. Client/Server Architecture

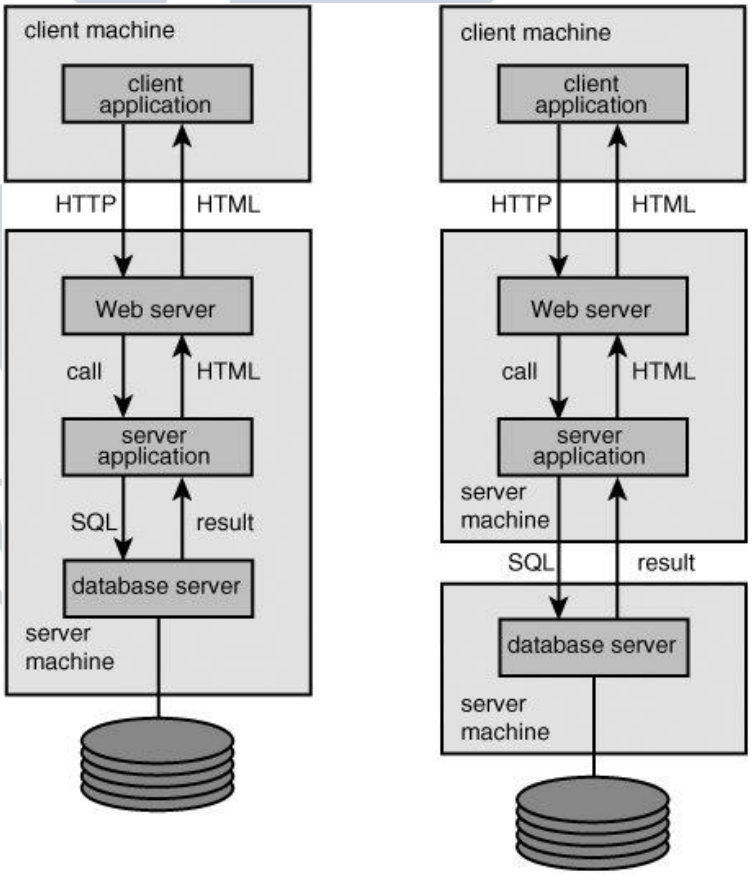
Client/Server merupakan model yang umum diterapkan saat ini pada berbagai macam aplikasi. Seperti terlihat pada Gambar 3.4, komputer dimana aplikasi berjalan disebut sebagai klien berada di mesin yang berbeda dengan komputer *database server*. Jalur komunikasi terjadi dalam jaringan LAN ataupun WAN.

Model ini sangat sesuai untuk diterapkan pada aplikasi yang dibuat karena aplikasi dapat diletakkan di komputer pengguna terpisah dari komputer *server*. Selain itu juga dimungkinkan penggunaan lebih dari satu aplikasi/user melalui banyak komputer yang terhubung ke *server*.



Gambar 3.4 *Client/Server Architecture*

C. Internet Architecture



Gambar 3.5 *Internet Architecture*

Arsitektur ini mirip dengan *client/server* namun letak perbedaannya adalah pembagian aplikasi menjadi dua bagian utama. Aplikasi berada di komputer klien dalam bentuk *user interface*, contohnya melalui web browser, sementara sebagian aplikasi yang berhubungan dengan *database* berada di komputer *server*. Kedua aplikasi ini kemudian dinamakan *client application* dan *server application*. Aplikasi di pihak klien dapat saja tidak memberikan *sql query* tetapi hanya memanggil aplikasi *server* dimana *query* sebenarnya berada. Aplikasi *server* dan *database* juga dapat berada di mesin yang berbeda.

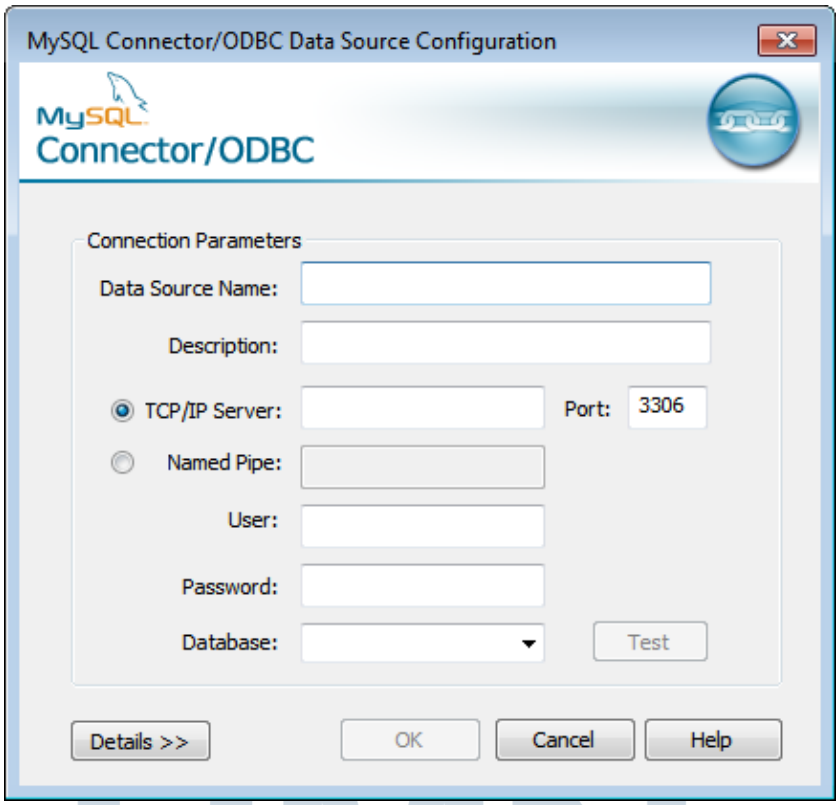
Penggunaan model ini pada aplikasi *order information trucking* sangat dimungkinkan dan dapat memberikan kelebihan dimana aplikasi dapat diakses dari komputer manapun. Namun karena penggunaan aplikasi yang hanya ditujukan bagi penggunaan oleh karyawan internal, model *client/server* dirasa lebih tepat.

3.3.1.3 *Open Database Connection* (ODBC)

ODBC merupakan sebuah standar konektivitas yang menyediakan API untuk menjalankan dan menghubungkan aplikasi dengan *Database Management System* dalam hal ini MySQL. ODBC dibangun oleh Microsoft agar semua aplikasi dapat melakukan koneksi ke *database server* dalam lingkungan yang heterogen. Artinya melalui ODBC, setiap aplikasi tidak membutuhkan pengaturan yang berbeda-beda jika hendak menggunakan *database management system* yang berbeda misalnya Oracle. Hal ini dapat terwujud selama terdapat ODBC client, ODBC *server* dan ODBC *driver*.

ODBC *Client* menggunakan *commands/query* untuk meminta data dari atau untuk mengirim data ke *server* DBMS. Namun, DBMS tidak akan memahami permintaan ODBC *Client* sebelum permintaan tersebut melewati ODBC *Driver*. Oleh karena itu, dibutuhkan *Driver* yang sesuai dengan DBMS. ODBC *Driver* ini adalah perangkat lunak yang berada di *front-end*. ODBC *Driver* selanjutnya menerjemahkan perintah ke dalam format yang dapat dimengerti

ODBC *Server*. ODBC *Server* mengirimkan jawaban kembali ke ODBC *driver*, yang kemudian akan diterjemahkan ke dalam format yang dimengerti ODBC *Client*.



Gambar 3.6 *ODBC Data Source Configuration* untuk MySQL

ODBC *Data Source* digunakan untuk mengatur koneksi antara ODBC Client dan ODBC *server* dengan ODBC *driver* yang tepat. Konfigurasi ODBC *Data Source* dapat dilihat pada Gambar 3.6. Melalui konfigurasi ini, aplikasi klien akan dapat mengetahui alamat dimana *server database* berada melalui IP *server*. Selanjutnya kolom *user* dan *password* memungkinkan aplikasi klien memiliki otoritas untuk mengakses *database* yang dipilih. *Data Source Name* merupakan nama alias yang digunakan aplikasi untuk memanggil *Data Source configuration* agar dapat menjalin koneksi dengan *server*.

3.3.1.4 Visual Basic

Dalam proses pembuatan aplikasi ini, penulis menggunakan *development tools* Visual Basic 2005 Express Edition. Pada tahap awal, penulis belajar menggunakan tools-tools yang biasanya digunakan dalam pembuatan aplikasi sederhana seperti data gridview dan timer. Selain itu juga penulis belajar

menggunakan beberapa *library* yang digunakan untuk ODBC *Connection* dan pembuatan objek-objek Microsoft Excel.

Visual Basic merupakan sebuah *software development tools* untuk membuat aplikasi pada platform windows berbasis .NET Framework. Visual Basic saat ini telah sepenuhnya merupakan pemrograman aplikasi berorientasi objek.

Terdapat beberapa *command* dan pengetahuan dasar yang dibutuhkan sebelum memulai pemrograman di Visual Bassic. Dimulai dengan perintah "New", merupakan perintah yang digunakan agar *compiler* membuat *instance* dari suatu kelas. Untuk mendeklarasikan sebuah variabel, digunakan perintah "Dim" yang berasal dari kata *dimension*. Dengan perintah ini, sebuah variabel baru dapat dideklarasikan dan diisi dengan nilai. Berikut contoh penggunaan "Dim".

```
Dim nilai As Integer
nilai = 10
```

Pada contoh tersebut, nilai merupakan variabel yang hendak dideklarasikan sebagai kelas tipe data integer. Penggunaan perintah "dim" dan "new" untuk mendeklarasikan variabel "winForm" sebagai tipe "Form" dapat dilihat pada contoh dibawah.

```
Dim winForm As System.Windows.Forms.Form = New System.Windows.Forms.Form()
```

Perintah selanjutnya adalah "Sub" atau *subroutine* yang digunakan untuk mendeklarasi suatu fungsi berisi sekelompok baris program dimana pada akhir baris, fungsi ini tidak mengembalikan nilai apapun kepada pemanggilnya.

```
Private Sub Load (ByVal object As System.Object)
.....< code is here >.....
End Sub
```

Adapun perintah "Function" menyerupai perintah "Sub" namun memiliki perbedaan dimana "Function" akan mengembalikan suatu nilai kepada pemanggilnya. Pada akhirnya nilai yang dikembalikan oleh "Function" juga mungkin tidak berisi apa-apa. Dibawah ini merupakan contoh penggunaan "Function".

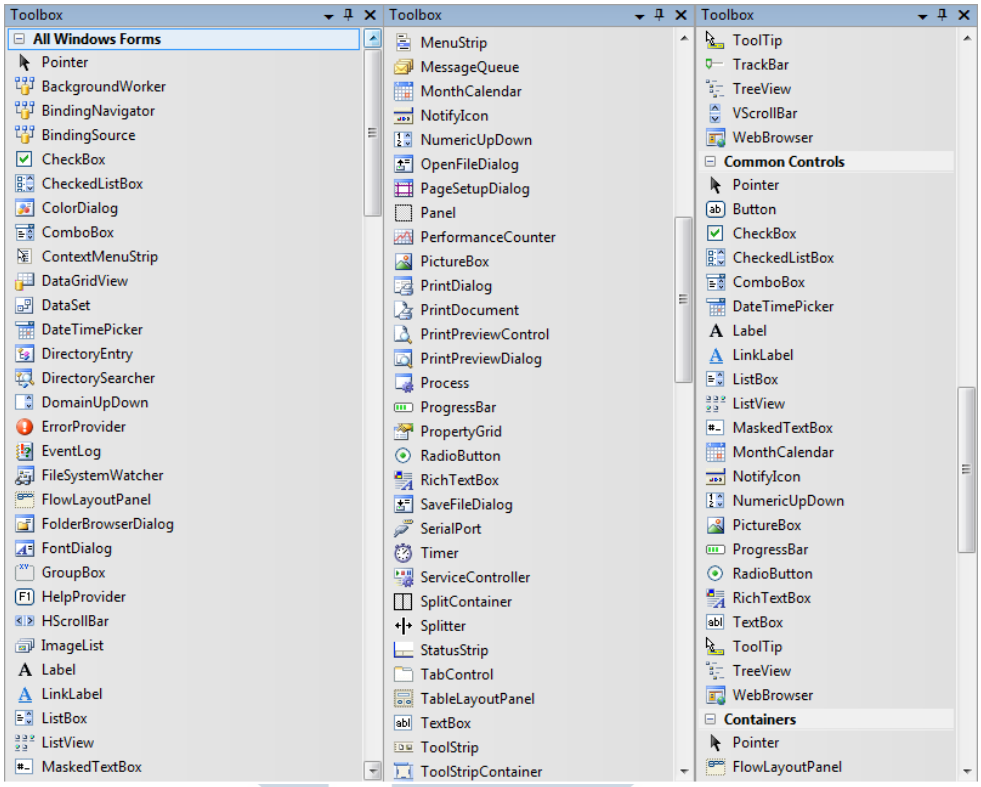
```
Public Function Add (ByVal ParamArray values () As Integer) As Long
        Dim result As Long = 0
        Result = values(0) + values(1)
        Return result
End Function
```

TABEL 3.1

Keywords dalam Visual Basic

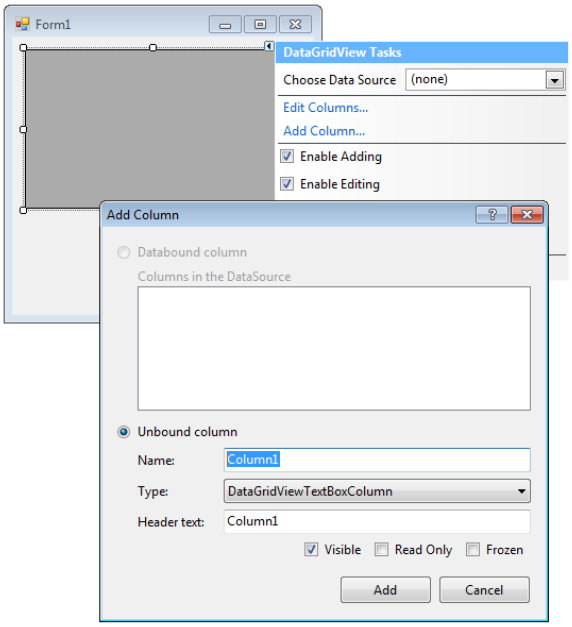
Keyword	Description
Namespace	A collection of classes that provide related capabilities. For example, the "System.Drawing" namespace classes associated with graphics.
Class	A definition of an object. Includes properties (variables) and methods, which can be subs or functions.
Instance	When a class is created, the resulting object is an instance of the class's definition. Not a keyword in Visual Basic.
Method	A generic name for a named set of commands. In Visual Basic, both subs and functions are types of methods. Not a keyword in Visual Basic.
Sub	A method that contains a set of commands, allows data to be transferred as parameters, and provides scope around local variables and commands.
Function	A method that contains a set of commands, return a value, allows data to be transferred as parameters, and provides scope around local variables and commands.
Return	Ends the currently executing sub or function. Combined with a return value for functions.
Dim	Declares and defines a new variable.
New	Creates an instance of an object.
Nothing	Used to indicate that a variable has no value. Equivalent to other languages and databases.
Me	A reference to the instance of the object within which a method is executing.
Console	A type of application that relies on a command-line interface. Console applications are commonly used for simple test frame. Also refers to a command window to and from which application can read and write text data.
Module	A code block that isn't a class but which can contain sub and functions methods. Not frequently used for application development, but is part of a console application.

Sumber : Professional Visual Basic 2008, hal.6



Gambar 3.7 *Toolbox* dalam Visual Basic 2005

DataGridView merupakan tools yang digunakan untuk menampilkan data dalam bentuk grid atau tabel. DataGridView merupakan tools yang mulai ada seletah .NET Framework 2.0. Kelas DataGridView memungkinkan kostumisasi sel, baris dan kolom yang ditampilkan aplikasi.



Gambar 3.8 Menambahkan Kolom pada DataGridView

Dalam aplikasi yang dibuat oleh penulis, DataGridView digunakan sebagai media komunikasi utama dengan pengguna untuk menampilkan data yang didapatkan dari *database*. Beberapa command digunakan antara lain

1. <dgvname>.Rows.Clear(), untuk menghapus seluruh baris yang ada dalam DataGridView dengan nama <dgvname>,
2. <dgvname>.Rows.Add(), digunakan untuk menambah baris kosong pada DatGridView
3. <dgvname>.Item(kolom,baris).Value, digunakan untuk merujuk nilai yang ada pada DataGridView dengan sel dengan indeks baris dan kolom yang ditunjuk. Dapat juga digunakan untuk mengisi nilai pada sel tersebut.
4. <dgvname>.Rows(baris).DefaultCellStyle.BackColor = color.Red, digunakan untuk memberikan warna pada satu blok sel di indeks baris yang ditunjuk, dalam contoh ini yaitu warna merah.

Selain DataGridView, penulis juga menggunakan *Timer Tools* yang dapat digunakan baik untuk penghitungan waktu maju ataupun mundur. Untuk mengaktifkan *timer*, digunakan *command* <Timername>.Enabled = True dan <Timername>.Enabled = False untuk menghentikan kembali timer. Setelah *timer* diaktifkan, *timer* akan menjalankan *timer event* atau *tick* yaitu satu blok *subroutine* yang akan dijalankan tiap siklus *timer*. Siklus ini dapat diatur tiap beberapa mili dettik atau lebih. Contoh, terdapat variabel second dengan nilai integer 60 sebagai berikut dengan siklus *timer* 1 detik

```
Private Sub <Timername>_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles <Timername>.Tick
        If second = 0 Then
                <Timername>.Enabled = False
                MsgBox("Waktu Habis")
        Else
                Second = Second – 1
        End If
End Sub
```

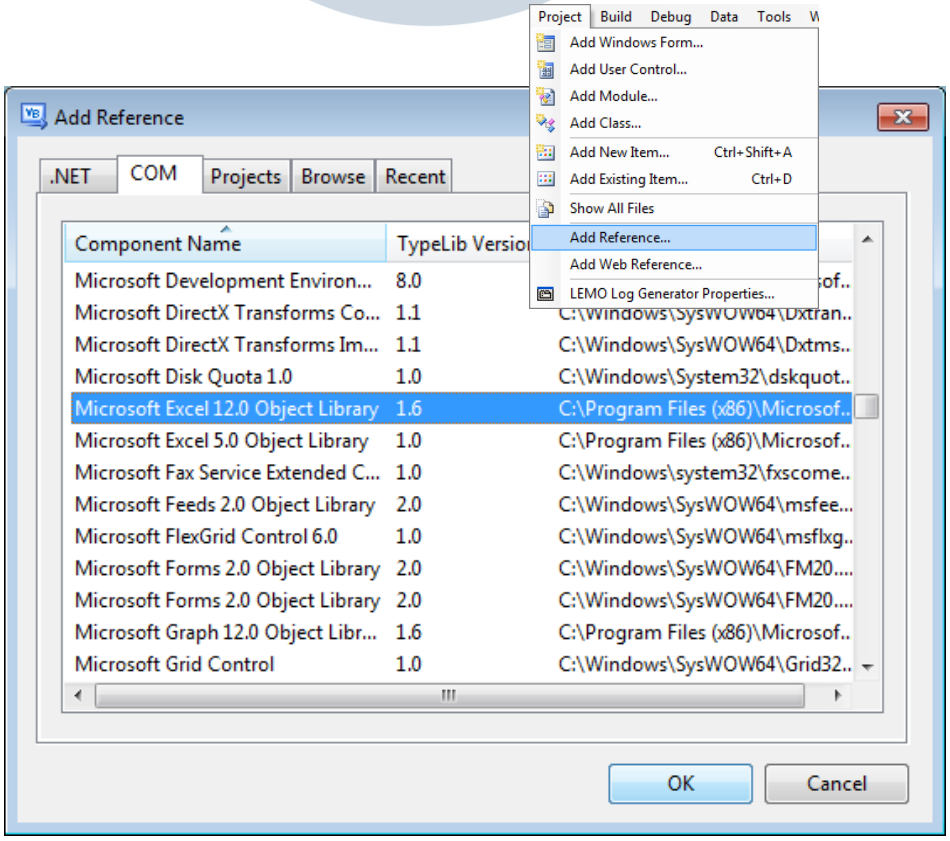
Apabila <Timername>.Enabled diberi nilai True maka *timer* akan aktif dan untuk setiap siklus *tick*, method <Timername>_Tick akan dijalankan. Setiap detiknya nilai second akan dikurangi hingga akhirnya second = 0 dan *timer* dinonaktifkan.

Pada contoh ini *timer* dibuat untuk menampilkan pesan “Waktu Habis” setelah 60 detik.

3.3.1.5 *Component Object Model* (COM)

Dalam aplikasi *Order Information Tracking* terdapat fitur automasi pembuatan *file* Microsoft Office Excel. Aplikasi tersebut akan membuat *Excel file* berisi tabel berisi informasi order yang ada dengan format yang telah diatur. Pembuatan *Excel file* melalui aplikasi yang dibuat dimungkinkan dengan adanya *Component Object Model* (COM) pada Visual Basic. Salah satu komponen yang terdaftar dalam COM adalah *Office Object Model*. Komponen ini tidak hanya mendukung automasi *Excel file* tetapi juga *Word file* dan *Outlook file*.

Untuk menggunakan *Office Object Model* ini, pertama-tama perlu disertakan *reference Office Object Library* bersangkutan ke dalam *project* yang sedang dibangun dalam hal ini *Excel Object Library* agar dapat menggunakan kelas-kelas objek *Excel*.



Gambar 3.9 Menambahkan *Excel Object Library Reference*

Beberapa kelas objek yang perlu diperhatikan untuk membuat *Excel file* antara lain,

1. Microsoft.Office.Interop.Excel.Application
2. Microsoft.Office.Interop.Excel.Workbook
3. Microsoft.Office.Interop.Excel.Worksheet
4. Microsoft.Office.Interop.Excel.Range

Objek-objek ini secara langsung melambangkan tampilan Microsoft Excel pada umumnya. Objek aplikasi merupakan aplikasi *Excel* itu sendiri sementara Workbook adalah lembar kerja berisi kumpulan Worksheet pada *Excel*. Objek *Range* merepresentasikan sel dimana nilai dapat diisikan dalam satu sel ataupun kumpulan sel.

A. Application Object

Microsoft.Office.Interop.Excel.Application merupakan kelas dari objek *Application* yang mendefinisikan atribut dan *methods* yang dapat dilakukan terhadap dan oleh objek *Application*. Objek ini memungkinkan dilakukannya pengaturan terhadap aplikasi *Excel* dan objek Workbook yang sedang dibuka oleh objek tersebut. Berikut adalah perintah yang dibutuhkan untuk mendeklarasikan sebuah variabel sebagai objek dari kelas Application.

```
Dim contohApp As Excel.Application

contohApp = New Excel.Application Class
```

B. Workbook Object

Microsoft.Office.Interop.Excel.Workbook merupakan kelas yang memuat atribut dan *methods* untuk objek Workbook *Excel*. Objek Workbook berjalan diatas objek aplikasi dan memungkinkan dibuatnya objek Worksheet. Secara *default* untuk setiap Workbook yang dibuat akan memiliki tiga lembar Worksheet. Berikut adalah perintah yang dibutuhkan untuk mendeklarasikan sebuah variabel sebagai objek dari kelas Workbook diatas objek *Application* yang telah dibuat sebelumnya.

```
Dim contohBook As Excel.Workbook

Dim misValue As Object = System.Reflection.Missing.Value

contohBook = contohApp.Workbook.Add(misValue)
```

C. Worksheet Object

Microsoft.Office.Interop.Excel.Worksheet merupakan kelas dari objek Worksheet *Excel*. Worksheet merupakan lembar kerja tersusun atas kolom dan baris dimana *user* melakukan aktifitas. Seperti telah disebutkan sebelumnya, setelah objek Workbook dibuat, secara *default* aka nada tiga lembar Worksheet yang tersedia dengan nama "Sheet1", "Sheet2", "Sheet3". Berikut adalah perintah yang dibutuhkan untuk mendeklarasikan sebuah variabel sebagai objek dari kelas Worksheet diatas objek Workbook yang telah dibuat sebelumnya dan memilih Worksheet sebagai lembar kerja aktif.

```
Dim contohSheet As Excel.Worksheet

contohSheet = contohBook.Sheets("Sheet1")
```

D. Range Object

Microsoft.Office.Interop.Excel.Range merupakan kelas dari objek Range *Excel*. Objek ini merupakan objek yang akan sering digunakan dalam manipulasi pembuatan *Excel file*. Dalam memanipulasi suatu bagian dalam Worksheet, bagian tersebut harus dinyatakan sebagai objek *Range* terlebih dahulu. Bagian dari objek *Range* adalah sebuah sel, baris, kolom atau kelompok sel. Objek *Range* memiliki banyak *method* yang digunakan antara lain untuk memasukkan nilai atau formula, mengubah format sel, memberikan *border*, melakukan fungsi *merge* dan lain-lain. Berikut adalah beberapa perintah yang digunakan sebagai *method* dari objek *Range*.

```
1.  Dim contohRange As Excel.Range
2.  contohRange = contohSheet.Range("A1","B1")
3.  contohRange.BorderAround()
4.  contohRange.Merge()
```

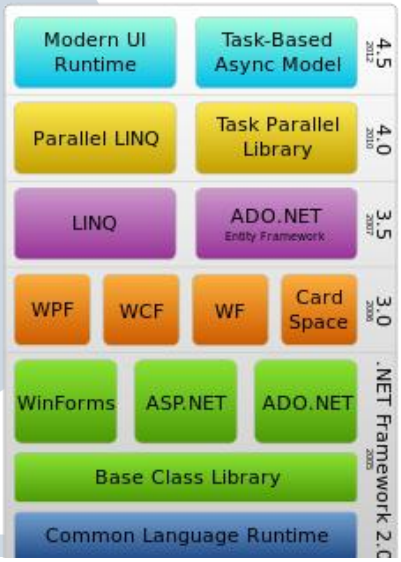
```
5.  contohRange.FormulaR1C1 = “Test”
6.  contohRange.Font.Bold = True
7.  contohRange.Font.Size = “18”
8.  contohRange.HorizontalAlignment = 3
9.  contohRange. VeritcalAlignment = 1
10. contohRange.ColumnWidth = 11
```

Perintah 1 digunakan untuk mendeklarasikan variabel tersebut ke dalam objek *Range*. Perintah 2 digunakan untuk memilih blok sel, dalam hal ini blok yang dipilih berada pada baris 1 kolom A dan B. Perintah ke 3 dan 4 digunakan untuk memberikan garis batas di sekeliling sel dan menggabungkan sel-sel dalam *range* menjadi satu sel. FormulaR1C1 pada perintah ke 5 digunakan untuk memasukkan *formula* atau nilai ke dalam sel sementara perintah 6 dan 7 digunakan untuk memanipulasi *atribut font* yaitu memberikan efek *bold* dan mengatur ukuran *font*. Untuk mengatur posisi tulisan di dalam sel, digunakan perintah 8 dan 9 baik dalam dimensi vertikal ataupun horizontal. Untuk posisi horizontal, terdapat beberapa nilai indeks yang bisa digunakan dari 1 hingga 8 dimulai dengan posisi *general, left, center, right, fill, justify, center across selection*, dan *distributed* berturut-turut. Sementara untuk posisi vertikal, dapat digunakan nilai indeks 1 hingga 5 dimulai dengan posisi *top, center, bottom, justify,* dan *distributed.*

3.3.1.6 Microsoft .Net Framework

Microsoft .NET Framework merupakan sebuah *framework* yang menyediakan *programming model* untuk aplikasi yang berjalan di *platform operating system* Windows. Baik untuk aplikasi *development tools* ataupun aplikasi lainnya yang berjalan di *environment* Windows membutuhkan .Net Framework. .Net Framework menyediakan *library* utama yang dibutuhkan untuk membuat aplikasi di Windows yang dapat digunakan oleh banyak bahasa

pemrograman atau *development tools* berbeda salah satunya Visual Basic. *Library* tersebut menyediakan pengaturan *user interface, data access, database connectivity*, kriptografi, pengembangan aplikasi berbasis web, algoritma numerik, dan komunikasi dalam jaringan. Selain *library* utama, .Net Framework juga menyediakan *Common Language Runtime* (CLR), aplikasi *virtual machine* yang mengatur dan menyediakan layanan keamanan, manajemen memori, dan *exception handling* bagi aplikasi. Semua aplikasi yang dibangun diatas .Net Framework akan diawasi oleh CLR untuk menjamin hal-hal tersebut. Hal ini memungkinkan pembuat program untuk lebih fokus pada inti aplikasi tanpa perlu mengatur *management memory* dsb.

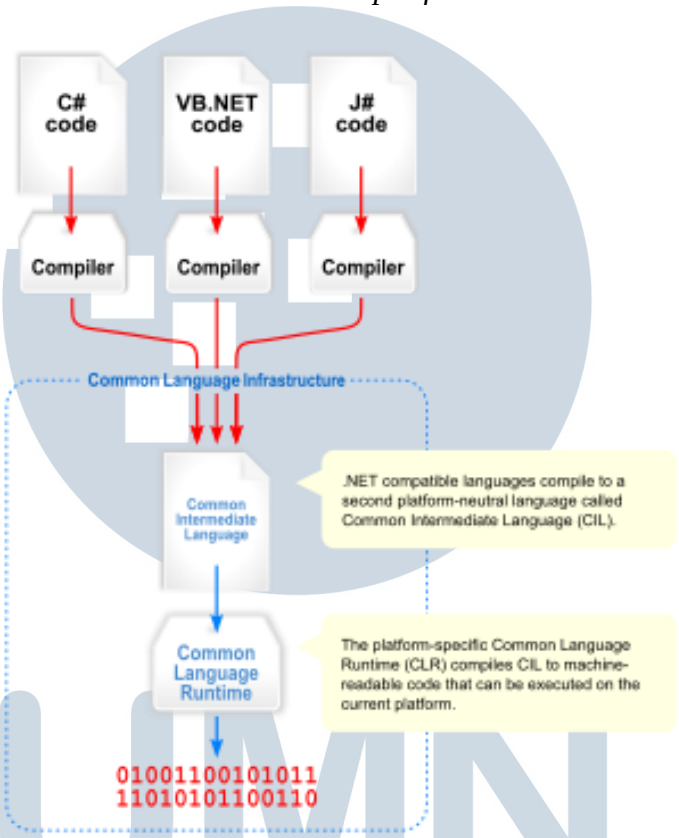


Gambar 3.10 Perkembangan .Net Framework

Windows XP saat pertama hadir masih menggunakan .Net Framework versi 1.0 dan mulai menggunakan versi 2.0 pada Windows XP SP2. Versi 2.0 akan turut disertakan dalam paket instalasi Visual Studio 2005. Hingga saat ini .Net Framework sudah mencapai versi 4.5.1 untuk Windows 8.1.

Interoperability merupakan model yang diusung oleh .Net Framework sebagai sebuah *framework* untuk *development tools application*. Dalam penggunaan sebuah aplikasi, tidak jarang terjadi juga interaksi dengan aplikasi lain seperti yang ada pada aplikasi *Order Information Tracking*. Aplikasi ini melibatkan interaksi dengan aplikasi lain yaitu Microsoft Excel baik Excel 2003, 2007 ataupun 2010. Melalui design interoperability, .Net Framework menyediakan kemampuan untuk tetap dapat menggunakan berbagai fungsi aplikasi baik untuk versi lama ataupun baru. Akses untuk *Component Object Model* (COM) terdapat dalam System.Runtime.InteropServices dan System.EnterpriseServices.

Dalam arsitektur .Net Framework dikenal istilah *Common Language Interface.* (CLI) bertujuan untuk menyediakan platform bahasa pemrograman yang tidak terikat pada satu aplikasi *development tools* dan juga CLR. CLI *code* ditempatkan di CLI *assemblies* yang disimpan dalam *Portable Executable Format.* Format ini umum dalam Windows *platform* untuk berkas DLL dan EXE.



Gambar 3.11 Gambaran *Common Language Interface* (CLI)

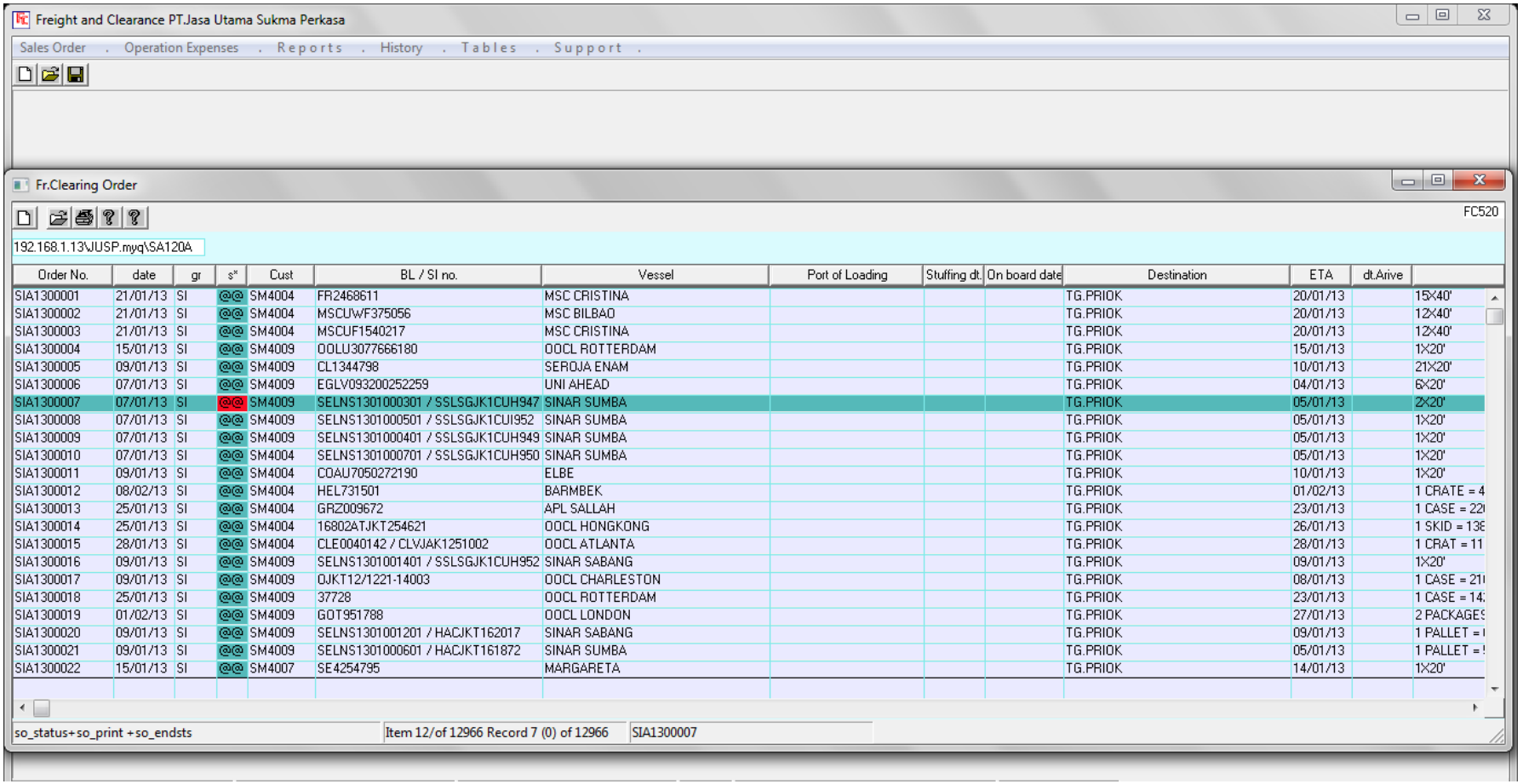
3.3.1.7 Alur Kerja dan Sistem Perusahaan

Alur kerja perusahaan mulai dari penerimaan order hingga pencetakan tagihan dapat dilihat pada lembar lampiran. Alur kerja dimulai ketika perusahaan mendapatkan *Delivery Order* (DO) dari *shipper*, perusahaan yang hendak melakukan impor ataupun ekspor barang. DO merupakan surat yang diterbitkan oleh pihak *shipping* (perusahaan penyedia jasa pengiriman ekspor/impor) sebagai tanda bukti pengambilan kontainer/peti kemas kosong dari DEPO (tempat penumpukan kontainer kosong) yang akan digunakan untuk pengiriman ekspor

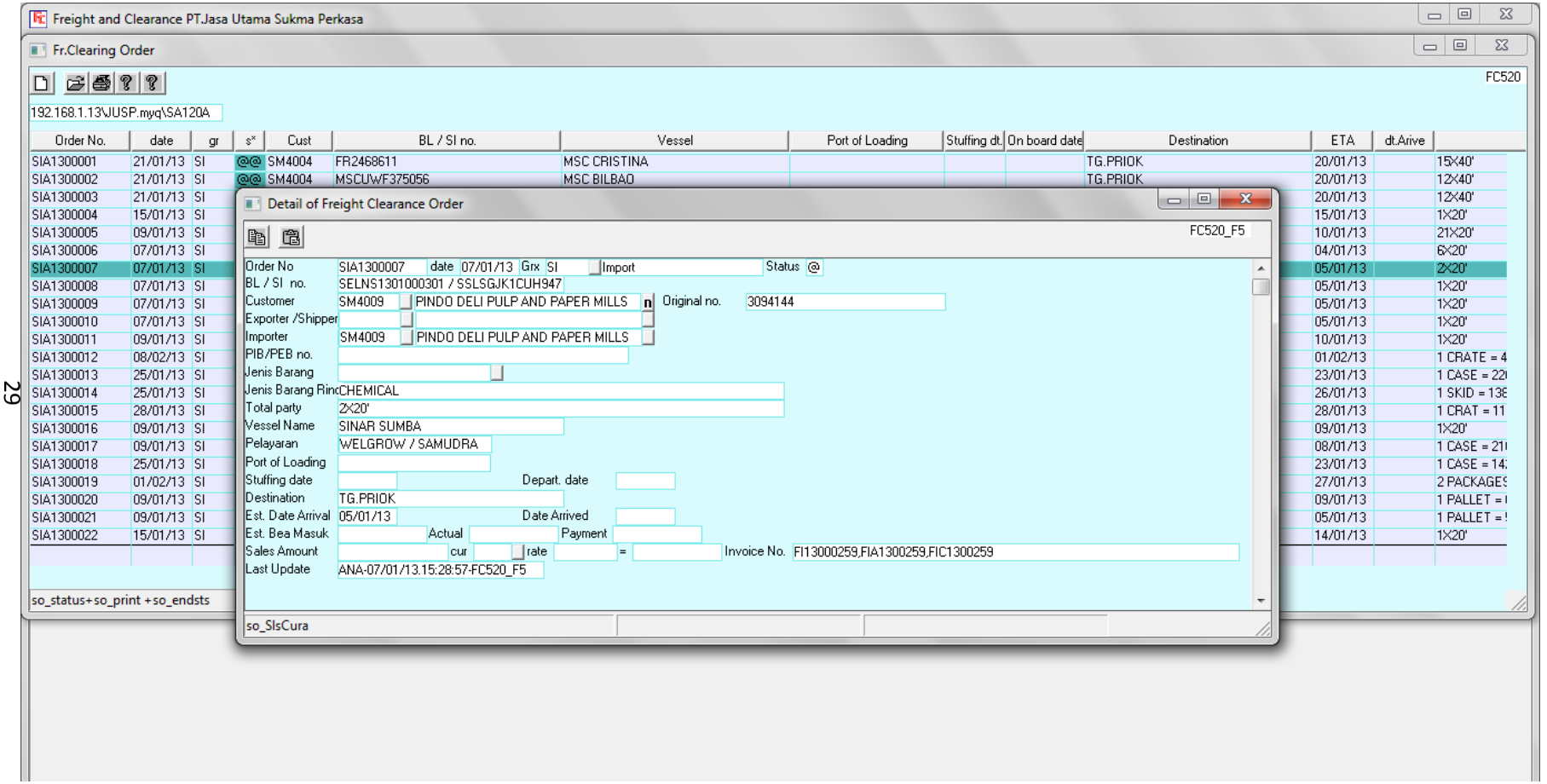
atau pengiriman kontainer berisi barang dari UTPK (Unit Tempat Penitipan Peti Kemas) ke gudang *shipper* dalam alur impor.

Ketika PT Lemo Utama telah menerima DO dari *shipper* maka perusahaan harus segera mengirim truk untuk menyelesaikan order. DO dapat diterima baik oleh *admin* ekspor, impor ataupun oleh bagian *trucking*. Permasalahan muncul ketika *admin* impor maupun ekspor dan *trucking* tidak saling berkoordinasi sehingga order yang diterima terlalu banyak. Hal ini akan menyulitkan bagian *trucking* untuk menentukan prioritas pelaksanaan order sementara unit truk yang tersedia terbatas. Apabila pada akhirnya pelaksanaan order tidak selesai pada waktunya, perusahaan akan mendapatkan denda dari pihak pelabuhan karena barang terlalu lama menumpuk di pelabuhan, peti kemas kosong terlambat dikembalikan ke DEPO, atau bahkan terlambat memuat peti kemas ke UTPK yang berakibat keterlambatan waktu berangkat kapal.

Untuk itulah perusahaan membutuhkan sebuah sistem yang dapat menampilkan seluruh order aktif yang ada, tanggal terima dan *deadline* order, serta status *realisasi* order agar *admin* impor, ekspor dan pihak *trucking* dapat saling berkooordinasi. PT Lemo Utama sebenarnya telah memiliki aplikasi utama yang digunakan mulai dari melakukan prosess *input* order impor dan ekspor, *update* status pelaksanaan order, hingga pencetakan surat tagihan namun aplikasi tersebut memiliki tampilan yang terlalu *detail* dan sulit digunakan untuk melihat status realisasi order. Dalam sebuah order, terdapat istilah partai yang berisi kode untuk mengindikasikan jumlah dan tipe peti kemas. Kode 6x40’ misalnya, menandakan bahwa order tersebut berisi 6 peti kemas dengan panjang 40 kaki. Untuk setiap peti kemas 40 kaki harus menggunakan sebuah truk dimana pelaksanaan order akan tergantung ketersediaan truk dan kebijakan bagian *trucking* sehingga dari 6 suborder tersebut mungkin saja hanya 2 yang baru terlaksana pada satu hari dan sisanya dilaksanakan di hari selanjutnya. Dalam aplikasi yang dimiliki perusahaan, status tiap-tiap suborder tersebut tidak dapat diketahui karena status yang tertera hanya mengindikasikan apakah semua suborder belum terlaksana atau semua suborder sudah terlaksana.



Gambar 3.12 *Main Menu* Aplikasi *Freight Clearing Order*



Gambar 3.13 *Sub Menu* Aplikasi *Freight Clearing Order*

Seperti terlihat pada kedua gambar (Gambar 3.12 dan Gambar 3.13) bahwa aplikasi tidak menampilkan status untuk tiap suborder seperti berapa jumlah peti kemas yang telah diangkut atau belum. Hal inilah yang sangat menyulitkan pihak *admin* impor ataupun ekspor karena tanpa koordinasi dari bagian *trucking*, mereka tidak dapat benar-benar mengetahui status pelaksanaan order. Terlebih lagi order yang ditampilkan bukan hanya memuat order aktif tetapi juga order lain yang telah selesai terlaksana sehingga dibutuhkan ketelitian yang lebih bagi *admin* untuk mengetahui order yang telah selesai atau belum.

3.3.1.8 Struktur *Database* Aplikasi

Aplikasi yang dibuat penulis menggunakan *database* MySQL sesuai dengan *database* yang digunakan oleh perusahaan. Aplikasi ini akan memanfaatkan *database* perusahaan yang sudah ada sehingga tidak diperlukan lagi pembuatan *database* baru atau tabel baru diluar *database* perusahaan. Hal ini dikarenakan aplikasi akan menggunakan data yang sebelumnya telah diinput dari aplikasi utama perusahaan. Aplikasi *Order Information Tracking* hanya melakukan proses mengambil dan menampilkan data tanpa menyimpan ataupun merubah data kembali ke *database*.

TABEL 3.2

Struktur Tabel ar010a

Field	Type
mp_nocust	char(6)
mp_grcust	char(3)
mp_loc	char(1)
mp_area	char(3)
mp_slsman	char(3)
mp_inst	char(5)
mp_name	char(25)
mp_adr1	char(25)
mp_adr2	char(25)
mp_adr3	char(25)
mp_phone	char(25)

Struktur Tabel ar010a (Lanjutan)

mp_nofax	char(25)
mp_email	char(33)
mp_person	char(25)
mp_bank	char(15)
mp_pkp	char(1)
mp_npwp	char(20)
mp_terms	decimal(3,0)
mp_credit	decimal(11,0)
mp_t_inv	decimal(11,0)
mp_t_do	decimal(11,0)
mp_t_pmt	decimal(11,0)
mp_t_xpmt	decimal(11,0)
mp_t_cost	decimal(11,0)
mp_hdate	date
mp_htime	char(8)
mp_huser	char(3)
mp_hprogr	char(8)

TABEL 3.3

Struktur Tabel ta110a

Field	Type
ta_noso	char(13)
ta_dtso	date
ta_dt_f	date
ta_dt_t	date
ta_dtend	date
ta_sales	char(3)
ta_cust	char(6)
ta_origin	char(15)
ta_servcd	char(4)
ta_loadcd	char(3)
ta_dari	char(6)
ta_ke	char(6)
ta_depo	char(6)
ta_area	char(3)
ta_ritcd	char(4)
ta_vessel	char(50)
ta_nabar	char(20)
ta_ujalan	decimal(6,0)
ta_ujalan2	decimal(6,0)
ta_ujalan3	decimal(6,0)

Struktur Tabel ta110a (Lanjutan)

ta_ujalan4	decimal(6,0)
ta_ujalan5	decimal(6,0)
ta_ujalan6	decimal(6,0)
ta_ujalan7	decimal(6,0)
ta_ujalan8	decimal(6,0)
ta_ujalan9	decimal(6,0)
ta_hkom	decimal(6,0)
ta_qbbm	decimal(5,0)
ta_qbbm2	decimal(5,0)
ta_qbbm3	decimal(5,0)
ta_qbbm4	decimal(5,0)
ta_qbbm5	decimal(5,0)
ta_qbbm6	decimal(5,0)
ta_qbbm7	decimal(5,0)
ta_qbbm8	decimal(5,0)
ta_qbbm9	decimal(5,0)
ta_qbbm10	decimal(5,0)
ta_qrit	decimal(3,0)
ta_qritx	decimal(3,0)
ta_qrita	decimal(3,0)
ta_qty	decimal(10,3)
ta_qtya	decimal(10,3)
ta_uom	char(3)
ta_hgsat	decimal(12,2)
ta_uprice	decimal(12,2)
ta_qfak	decimal(10,3)
ta_hfak	decimal(10,0)
ta_status	char(1)
ta_endsts	char(1)
ta_print	decimal(1,0)
ta_huser	char(3)
ta_hdate	date
ta_htime	char(8)
ta_hprogr	char(8)

TABEL 3.4

Struktur Tabel ta120a

Field	Type
tr_notrx	char(10)
tr_dttrxx	date
tr_dttrx	date
tr_tmtrx	char(8)

Struktur Tabel ta120a (Lanjutan)

tr_dtend	date
tr_tmend	char(8)
tr_noso	char(13)
tr_cust	char(6)
tr_sales	char(3)
tr_servcd	char(10)
tr_dari	char(6)
tr_ke	char(6)
tr_depo	char(6)
tr_area	char(3)
tr_ritcd	char(4)
tr_rit	decimal(1,0)
tr_loadcd	char(3)
tr_nabar	char(20)
tr_qty	decimal(10,3)
tr_uom	char(3)
tr_hgsat	decimal(12,2)
tr_uprice	decimal(12,2)
tr_vessel	char(50)
tr_contnr	char(60)
tr_cont_dtin	date
tr_cont_dtin2	date
tr_cont_jmin	decimal(5,2)
tr_cont_jmin2	decimal(5,2)
tr_cont_dtout	date
tr_cont_dtout2	date
tr_cont_dtout3	date
tr_cont_jmout	decimal(5,2)
tr_cont_jmout2	decimal(5,2)
tr_cont_jmout3	decimal(5,2)
tr_ref1	char(10)
tr_ref2	char(10)
tr_nopin	char(5)
tr_nopol	char(10)
tr_supir	char(4)
tr_ujalan	decimal(8,0)
tr_ujalan2	decimal(8,0)
tr_ujalan3	decimal(8,0)
tr_ujalan4	decimal(8,0)
tr_ujalan5	decimal(8,0)
tr_ujalan6	decimal(8,0)
tr_ujalan7	decimal(8,0)
tr_ujalan8	decimal(8,0)
tr_ujalan9	decimal(8,0)
tr_sujln2	decimal(8,0)
tr_sujln3	decimal(8,0)
tr_sujln4	decimal(8,0)
tr_supv	char(6)
tr_nofak	char(10)
tr_dtfak	date
tr_qfak	decimal(7,3)
tr_hgfak	decimal(10,0)
tr_hsup	decimal(10,0)

Struktur Tabel ta120a (Lanjutan)

tr_adjkom	decimal(10,0)
tr_per	decimal(2,0)
tr_week	decimal(2,0)
tr_ajalan	decimal(4,0)
tr_dtbbm	date
tr_qbbm	decimal(3,0)
tr_qbbm2	decimal(3,0)
tr_qbbm3	decimal(3,0)
tr_qbbm4	decimal(3,0)
tr_qbbm5	decimal(3,0)
tr_qbbm6	decimal(3,0)
tr_qbbm7	decimal(3,0)
tr_qbbm8	decimal(3,0)
tr_qbbm9	decimal(3,0)
tr_qbbm10	decimal(3,0)
tr_hkom	decimal(6,0)
tr_skomisi	decimal(9,0)
tr_sqbbm	decimal(5,0)
tr_sqbbm2	decimal(5,0)
tr_shbbm	decimal(11,0)
tr_qbbma	decimal(5,0)
tr_hbbma	decimal(11,0)
tr_hkenek	decimal(9,0)
tr_statux	char(1)
tr_status	char(1)
tr_endsts	char(1)
tr_arsts	char(1)
tr_print	decimal(1,0)
tr_hdate	date
tr_htime	char(8)
tr_huser	char(3)
tr_hprogr	char(8)

TABEL 3.5

Struktur Tabel sj120a

Field	Type
so_noso	char(10)
so_blno	char(33)
so_origin	char(22)
so_name	char(33)
so_stuffdt	char(66)
so_party	char(66)
so_vessel	char(33)
so_destina	char(33)
so_noor	char(10)

Struktur Tabel sj120a (Lanjutan)

so_jum	char(10)
so_dr	char(22)
so_ke	char(22)
so_cont	char(33)
so_seal	char(33)
so_oper	char(10)
so_ksg	char(22)
so_dtsj	date
so_nosj	char(10)
so_ket	char(33)

Aplikasi ini membutuhkan akses ke empat tabel diatas yaitu tabel ar010a, ta110a, ta120a, dan sj120a. Tabel ar010a, ta110a, dan ta120a berada pada satu *database* yang sama sementara tabel sj120a berada pada *database* terpisah. Untuk itu aplikasi ini akan membangun dua jalur komunikasi saat mengakses seluruh tabel. Tabel ar010a merupakan tabel berisi nama perusahaan pelanggan dan kode pelanggan. Tabel ta110a berisi daftar order atau transaksi yang diterima perusahaan sementara tabel ta120a merupakan tabel yang berisi rincian dari tiap order di tabel ta110a. Adapun tabel sj120a berisi informasi perkapalan tiap order termasuk tanggal Tila/SP2/SPPB, surat perintah pengeluaran barang dari pelabuhan untuk segera mengambil barang. Jika pada hari ke tiga sejak diterbitkannya SPPB, peti kemas masih atau belum berada di pelabuhan untuk dikirim maka perusahaan pendapatkan denda 200% per hari dari nilai biaya angkut saat itu.

3.3.1.9 Order Information Tracking

A. Penggunaan dan Peranan Aplikasi

Aplikasi ini ditujukan untuk digunakan oleh beberapa divisi dalam departemen operasional yaitu *admin* impor, ekspor dan *trucking*. Tujuan utama dari dibuatnya aplikasi ini adalah untuk mempermudah koordinasi antar divisi tersebut dimana untuk setiap order baru yang diterima dan diinput ke sistem oleh masing-masing *admin* akan dapat dilihat melalui

aplikasi. Begitu juga dengan order yang telah terlaksana atau selesai. Dengan demikian baik *admin* impor dan ekspor dapat mengetahui jumlah order yang sedang dijalankan *trucking* dan memperkirakan apakah masih memungkinkan untuk dilakukan penambahan order baru berdasarkan tanggal *deadline* tiap order dan jumlah order yang ada. Bagi pihap *trucking*, aplikasi ini juga membantu dalam hal menentukan prioritas order.

Aplikasi ini pada akhirnya akan menghasilkan *Excel file* berisi salinan terhadap data yang ditampilkan di antarmuka aplikasi. Jumlah *Excel file* yang dihasilkan ada dua, *file* pertama menyimpan order yang diterima PT Lemo Utama di kantor pusat dan *file* kedua menyimpan order yang diterima di kantor pegangsaan. Data order yang diterima pihak perusahaan yang berlokasi di pegangsaan tidak turut ditampilkan di antarmuka aplikasi namun hanya ditulis ke dalam *Excel file* kedua. Pada Gambar 3.14 dan Gambar 3.15 ditunjukkan file Lemo_Order.xlsx dan PGS_Order.xlsx.

Order By Day

No.	TGL	TILA	IMP/EXP	Customer	TOI/TOE	Container	Partay	Realisasi	Outstanding
1	16 . 1 . 2014	-	424 EXP	JU QQ INDAH KIAT PULP PAP	TOE3000477	Pendek 20"	1	0	1
2	16 . 1 . 2014	-	437 EXP	JU QQ INDAH KIAT PULP PAP	TOE3000490	Pendek 20"	1	0	1
3	15 . 1 . 2014	15 . 1 . 2014	128 A	JU QQ PINDO DELI PULP PAP	TOI3000336	Pendek 20"	15	3	12
4	14 . 1 . 2014	14 . 1 . 2014	1068 A	JU QQ INDAH KIAT PULP PAP	TOI3000323	Panjang 40"	25	19	6

Gambar 3.14 File Lemo_Order.xlsx

Order By Day

No.	TGL	IMP/EXP	Customer	TOI/TOE	Partay	Qty. Rit	Realisasi	Outstanding
1	15 . 1 . 2014	PGS0660	ROLITRANS INTERNATIONAL	TOP1000456	Panjang 40"	1	0	1
2	13 . 1 . 2014	PGS0647	KOMPONINDO BETONJAYA	TOP1000430	Panjang 40"	3	2	1

Gambar 3.15 File PGS_Order.xlsx

B. Alur Proses dalam Aplikasi

Secara garis besar, aplikasi ini berjalan dengan sangat sederhana dimulai dari membuat *Excel file* kosong, membaca *database*, menuliskan hasil pembacaan tabel baris demi baris ke datagridview pada tampilan aplikasi dan *Excel file* secara bersamaan, dan terakhir menyimpan *Excel file*. Proses ini dilakukan berulang-ulang saat timer habis dan diulang kembali.

Untuk penjelasan lebih detail tentang jalannya proses dalam aplikasi akan diuraikan sebagai berikut

1. Pada saat aplikasi dijalankan, timer akan aktif dan menghitung mundur berdasarkan waktu yang ditetapkan. Secara default, timer diatur pada rentang waktu satu jam.

2. Bersamaan dengan aktifnya timer, aplikasi akan membuka *Excel file* dengan nama Lemo_Order.xlsx atau membuat *file* baru jika *file* belum pernah dibuat atau tidak ditemukan. Semua sel dalam *Excel file* tersebut akan dihapus sehingga tampak seperti *file* baru. Untuk setiap *overwrite* yang dilakukan pada file, aplikasi akan memunculkan *dialog box* untuk meminta persetujuan pengguna. Agar *dialog box* tidak muncul, DisplayAlerts harus dinonaktifkan terlebih dahulu. Berikut perintah yang digunakan untuk membuat *Excel file* pada Visual Basic.

```
xlApp = New excel.ApplicationClass
xlWorkBook = xlApp.Workbooks.Add(misValue)
xlWorkSheet = xlWorkBook.Sheets("sheet1")

My.Computer.FileSystem.CreateDirectory _
("C:\LEMO\")
xlApp.DisplayAlerts = False
xlWorkSheet.SaveAs("C:\LEMO\Lemo_Order.xlsx")
xlApp.DisplayAlerts = True
xlWorkSheet.Cells.Clear()
```

Gambar 3.16 Membuat *Excel file*

3. Setelah *Excel file* dibuat, aplikasi akan membuat *header* pada *file* dan mengatur format worksheet seperti lebar tiap kolom, penjajaran dan ukuran tulisan.

4. *Excel file* telah siap dan langkah selanjutnya adalah melakukan koneksi ke *database* dan melakukan proses pengambilan data dengan perintah SELECT.

```
koneksi.Open()
cmd = koneksi.CreateCommand
cmd.CommandText = "SELECT d.mp_name,c.ta_dtso,c.ta_qrit,
c.ta_ritcd,c.tr_ritcd,c.tr_nabar,c.ta_cust,c.ta_noso,
c.ta_origin,c.tr_statux,c.tr_status,c.jumlah from
(select b.ta_dtso,b.ta_qrit,b.ta_ritcd,a.tr_ritcd,
a.tr_nabar,b.ta_cust,b.ta_noso,b.ta_origin,a.tr_statux,
a.tr_status,count(*) as jumlah from ta120a a,ta110a b
where b.ta_origin NOT like '%PG%' AND a.tr_noso=b.ta_noso
group by b.ta_noso,a.tr_status order by b.ta_dtso desc,
b.ta_noso asc) c, ar010a d where c.ta_cust=d.mp_nocust"
hasil = cmd.ExecuteReader
```

Gambar 3.17 *Query* Pengambilan Data

5. Data hasil pembacaan akan ditampung dalam variabel objek odbcDataReader yaitu “hasil”. Data yang didapatkan berbentuk tabel berisikan list order beserta informasi nama pelanggan, tanggal order, dan jumlah realisasi order. Data dalam “tabel virtual” tersebut akan dibaca satu per satu dari baris teratas hingga paling bawah. Dalam setiap siklus pembacaan, untuk satu baris dalam tabel yang terbaca dan memenuhi syarat *if else*, akan ditampilkan dalam baris aktif di DataGridView dan ditulis ke dalam baris aktif di *Excel file*. Setelah itu dibuat baris baru dibawah baris yang terisi baik di DataGridView ataupun worksheet dan dijadikan sebagai posisi baris aktif yang baru. Proses ini diulang terus menerus hingga tidak ada lagi data tersisa. Adapun angka total realisasi per order didapatkan dari penjumlahan detail order pada tabel ta120a. Untuk baris dengan nilai ta_noso (kode order) dan ta_origin (kode ekspor/impor) yang sama dengan baris sebelumnya, menunjukkan bahwa baris tersebut berasal dari order yang sama. Dengan

demikian order tidak perlu dicatat pada baris baru melainkan hanya memperbaharui status realisasi order (+1 atau -1).

```
If hasil.HasRows = True Then
    dgv.Rows.Clear()
    While hasil.Read
        If (hasil.Item("ta_dtso").year.ToString.Substring(0, 4).Equals
            dgv.Rows.Add()
            kol1 = "a" & baris + 4
            kol2 = "j" & baris + 4
            chartRange = xlWorkSheet.Range(kol1, kol2)
            chartRange.Borders.Value = True

            dgv.Item(0, baris).Value = baris + 1
            xlWorkSheet.Cells(baris + 4, 1) = baris + 1

            dgv.Item(1, baris).Value = hasil.Item("mp_name")
            xlWorkSheet.Cells(baris + 4, 5) = hasil.Item("mp_name")
```

Gambar 3.18 Pembuatan *Header* pada *Excel File*

6. Masih dalam satu siklus pembacaan, aplikasi akan membuat *nested connection* untuk mengakses tabel sj120a pada *database* lain. Hal ini dilakukan untuk mendapatkan informasi mengenai tanggal Tila/SP2/SPPB untuk masing-masing order.

```
If tanggaltila.AddDays(3).CompareTo(Now) <= 0 Then
    dgv.Rows(baris).DefaultCellStyle.BackColor = Color.Red
    chartRange.Interior.Color = RGB(255, 0, 0)
ElseIf tanggaltila.AddDays(2).CompareTo(Now) <= 0 Then
    dgv.Rows(baris).DefaultCellStyle.BackColor = Color.Yellow
    chartRange.Interior.Color = RGB(255, 255, 0)
ElseIf tanggaltila.CompareTo(Now) <= 0 Then
    dgv.Rows(baris).DefaultCellStyle.BackColor = Color.Green
    chartRange.Interior.Color = RGB(0, 255, 0)
Else
    dgv.Rows(baris).DefaultCellStyle.BackColor = Color.White
    chartRange.Interior.Color = RGB(255, 255, 255)
End If
```

Gambar 3.19 Penentuan Prioritas Order

Seperti dapat dilihat pada gambar, untuk order yang belum terselesaikan hingga hari ke tiga setelah terbitnya SPPB akan diberi warna merah baik pada DataGridView ataupun *Excel file*. Sementara order yang belum terselesaikan hingga hari ke dua akan diberi warna kuning dan order yang belum terselesaikan hingga hari pertama setelah tanggal Tila/SP2/SPPB akan diberi warna

hijau. Saat sebuah order sudah mendapat warna merah berarti perusahaan sudah pasti akan mendapat denda dari pelabuhan.

7. Langkah terakhir adalah menyimpan *Excel file* dan melakukan perintah releaseObject. Perintah ini perlu dilakukan agar aplikasi *Excel* dapat benar-benar ditutup dan tidak menyebabkan tertinggalnya objek atau proses yang berjalan.

```
Private Sub releaseObject(ByVal obj As Object)
    Try
        System.Runtime.InteropServices.Marshal.ReleaseComObject(obj)
        obj = Nothing
    Catch ex As Exception
        obj = Nothing
    Finally
        GC.Collect()
    End Try
End Sub
```

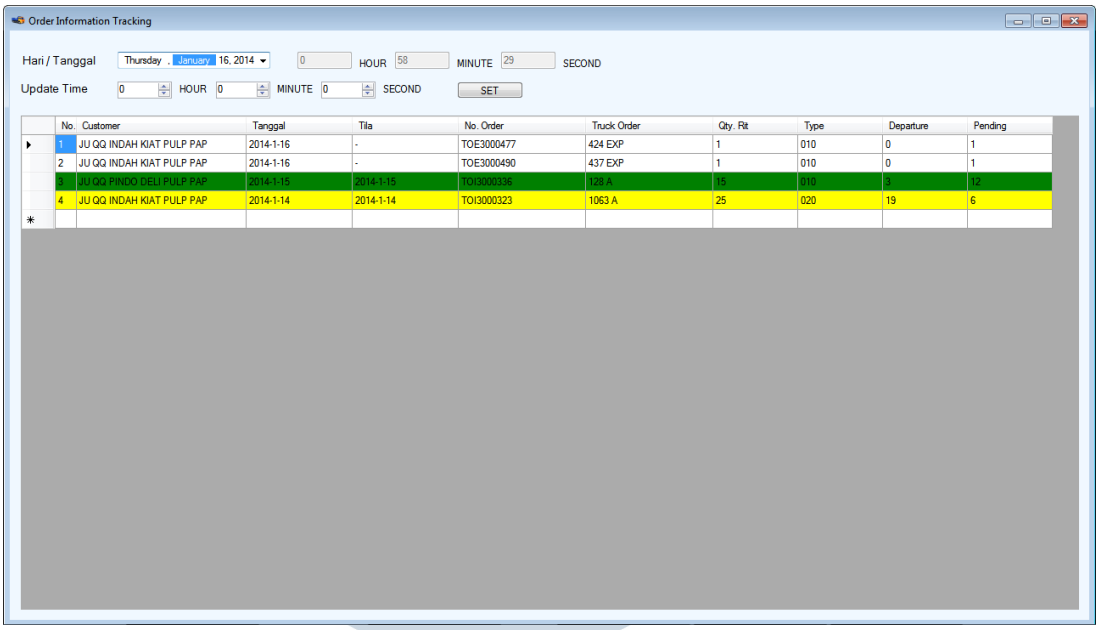
Gambar 3.20 Fungsi Sub realeaseObject

8. Seluruh langkah tersebut diulang untuk membuat *file* PGS_Order.xlsx dimana perbedaan terletak pada proses penyaringan *if else*. Setelah timer *timeout* maka secara otomatis seluruh proses diulang kembali dan timer dikembalikan ke nilai awal.

C. Antar Muka Aplikasi

Perancangan antarmuka aplikasi dilakukan di awal perancangan aplikasi. Untuk merancang antarmuka yang baik, penulis menyesuaikan dengan kebutuhan perusahaan dan pengguna dalam hal ini *admin* impor, ekspor, dan *trucking*. Sesuai dengan kebutuhan tersebut, antar muka aplikasi dibuat sesederhana mungkin agar pengguna tidak kesulitan dalam melihat informasi. Informasi ditampilkan dalam bentuk tabel dan kolom dimana penentuan kolom-kolom ini juga sudah didiskusikan dengan pengguna dan pembimbing lapangan agar kolom yang digunakan benar-benar merupakan kolom yang dibutuhkan dalam koordinasi antar divisi. Adapun pengaturan yang diberikan kepada pengguna hanya berupa

penentuan *timer interval*. Gambaran antarmuka aplikasi ini dapat dilihat pada gambar berikut. Adapun data yang ditunjukkan merupakan data *dummy* untuk menjaga kerahasiaan perusahaan.



Gambar 3.21 Antar Muka *Order Information Tracking*

D. Pengujian dan Implementasi Aplikasi

Sebelum dilakukan implementasi aplikasi *Order Information Tracking*, penulis melakukan pengujian akan keandalan data yang ditampilkan oleh aplikasi. Dalam tahap awal dilakukan pengecekan silang antara status order yang ditampilkan pada aplikasi dengan status order sebenarnya sekaligus menguji apakah order yang ditampilkan benar-benar masih berjalan atau sebenarnya sudah selesai.

Pengujian juga dilakukan untuk mengetahui apakah proses yang berjalan mengalami *crash* khususnya pada *nested connection* dan pengulangan proses saat *timer timeout*. Dalam proses pengujian ini, penulis mendapatkan permintaan untuk menambah fitur pada aplikasi. Fitur tersebut adalah untuk menentukan tingkat prioritas order berdasarkan

warna. Pada tahap awal perancangan aplikasi, fitur ini belum dimasukkan dalam *user requirements*.

Setelah melakukan serangkaian pengujian dan perbaikan, penulis melakukan pengarahan kepada *admin* impor, ekspor dan *trucking* terkait cara penggunaan aplikasi.

3.3.2 Permasalahan dan Solusi

Dalam proses pembuatan aplikasi mulai dari perancangan, pengujian dan implementasi, didapatkan beberapa kendala. Kendala yang ditemui dan solusi yang dilakukan antara lain,

1. Penulis menemukan kesulitan dalam membuat *Excel file* dimana untuk menjalankan aplikasi, direktori dan *Excel file* kosong harus dibuat secara manual terlebih dahulu. Masalah ini muncul karena penggunaan perintah *open* dalam inisiasi *Excel file*. Permasalahan dapat diselesaikan dengan mengganti perintah *open* menjadi *save as* seperti ditunjukkan pada Gambar 3.17.

```
xlApp = New excel.ApplicationClass
xlWorkBook = xlApp.Workbooks.Open("C:\LEMO\LEMO_Order.xlsx")
xlWorkSheet = xlWorkBook.Sheets("sheet1")
xlWorkSheet.Cells.Clear()
```

Gambar 3.22 Perintah Open untuk Membuka *File*

2. Dalam mengakses tabel sj120a, aplikasi tidak bisa mengeksekusi perintah SELECT karena objek koneksi sedang digunakan dan belum ditutup sementara akes ke tabel sj120a dibutuhkan dalam tiap siklus proses dimana siklus membutuhkan koneksi yang terus tersambung. Untuk mengatasi permasalahan ini, aplikasi harus membuat dua koneksi dimana koneksi kedua berjalan di dalam proses dimana koneksi pertama tetap terhubung.

3. Permasalahan lainnya muncul setiap kali aplikasi membuat *Excel file* dimana pada *task manager*, aplikasi *Excel* masih menunjukkan status

running meskipun *file* telah ditutup. Pada daftar *process* di task manager juga menunjukkan adanya proses-proses *Excel* dari inisiasi *file* sebelumnya yang masih berjalan. Penulis kemudian mendapatkan bahwa hal tersebut disebabkan karena objek yang belum dimatikan meskipun *Excel file* sudah ditutup atau bahkan aplikasi *Order Information Tracking* ditutup. Untuk itu perlu ditambahkan fungsi releaseObject yang dapat dilihat pada gambar 3.21.

4. Pada tahap implementasi, ditemukan permasalahan dimana aplikasi yang telah dibuat tidak dapat dibuka di PC *admin* impor, ekspor maupun *trucking* dan selalu menunjukkan pesan error. Inti permasalahan kemudian diketahui dan ternyata disebabkan oleh versi .Net Framework yang terinstal pada PC masing-masing *admin*. Dalam aplikasi ini penulis menggunakan *Component Object Model* (COM) Microsoft *Excel* 12.0 *Object library* untuk membuat *Excel file* dengan format .xlsx yang hanya didukung oleh .Net Framework dengan versi diatas 2.0. Sebagai solusi mengatasi masalah tersebut maka dilakukan pembaharuan .Net Framework pada tiap PC dengan versi terbaru.