BAB IV
IMPLEMENTASI SISTEM

Dalam bab ini dijelaskan tentang implementasi teknologi Active Server Pages yang diaplikasikan dalam suatu *web page* yang bersifat interaktif, dan mudah dalam pemeliharaannya. Aplikasi ini merupakan salah satu contoh aplikasi e-commerce yang banyak terdapat di dalam pemrograman internet (*web programming*) dan merupakan salah satu bidang yang paling populer yang banyak dilakukan oleh masyarakat sekarang ini, terutama masyarakat teknologi informasi.

4.1 TEKNOLOGI YANG DIGUNAKAN DALAM APLIKASI

Aplikasi e-commerce dengan menggunakan Active Server Pages ini merupakan aplikasi yang didesain sedemikian rupa agar dapat dimengerti dan dijalankan secara mudah oleh *client*, *user-friendly*, dan dapat diakses dalam waktu yang relatif singkat. Jadi, semua aspek mengenai masalah kecepatan eksekusi, redundansi, dan *interface* telah dijadikan bahan pertimbangan pada saat pendesainan aplikasi ini. Adapun aplikasi transaksi *online* ini menggunakan antara lain:

- Microsoft Windows 98 sebagai Sistem Operasi (*Operating System*).
- Microsoft Personal Web Server sebagai *Web Server*.
- Microsoft Access sebagai *Database Server*.
- Macromedia DreamWeaver sebagai perangkat lunak perancangan dan desain sistem, dengan menggunakan kombinasi ActiveX control, dan tag HTML.
- Power Builder dari PowerSoft sebagai perangkat lunak perancangan aplikasi *back-end*.
- Termasuk penggunaan fasilitas ODBC sebagai pengakses basis data universal yang dikembangkan oleh Microsoft.

Untuk menjalankan semua aplikasi di atas, termasuk dengan aplikasi transaksi *online* ini, *server* yang dibangun masih menggunakan teknologi *single-tier*. Pengertian *single-tier* ini adalah semua jenis *server*, baik *web server*, *database server*, maupun *application server* terletak pada komputer yang sama. Dengan demikian, semua program dan data akan disimpan di dalam satu komputer. Prinsip ini sama dengan prinsip yang dibentuk oleh sebuah *mainframe*, dan memiliki sejumlah kekurangan, seperti membutuhkan sumber daya yang besar. Untuk lebih mempercepat pengaksesan, mengurangi kebutuhan sumber daya yang besar, serta memudahkan pemeliharaan sistem yang ada, ada baiknya sistem dapat dikembangkan dengan menggunakan teknologi *two-tier* atau bahkan *three-tier*. Teknologi *two-tier* telah diterapkan di dalam sistem *client/server*, yang memisahkan kebutuhan dan fungsi antara *client* dan *server*. Sedangkan teknologi *three-tier* telah membagi kebutuhan antara *web server*, *database server*, dan *application server* ke dalam tiap komputer yang berbeda. Teknologi *three-tier* memungkinkan penggunaan *resource* yang tidak terlalu besar, dan peng-*update*-an di antara ketiganya menjadi lebih mudah.

4.1 IMPLEMENTASI BASIS DATA

Dari hasil pemetaan terhadap *ER-Diagram*, yang telah dijabarkan dengan *mapping diagram*, dan keterangan pada kamus data yang ada pada bab IV, maka dapat dijabarkan sejumlah tabel yang menjadi struktur basis data yang digunakan di dalam aplikasi transaksi *online*, dan yang juga merupakan detail basis data yang akan dipergunakan dalam pembuatan aplikasi tersebut, antara lain:

❖ Tabel Barang

Tabel ini digunakan untuk menyimpan data produk yang terdiri dari *field* kode barang, kode kategori barang, nama barang, jumlah barang, berat barang, harga beli, harga jual, keterangan barang, dan file gambar yang berguna untuk menentukan lokasi di mana file gambar produk berada. Tabel Barang memiliki *field* kode_barang sebagai *primary key* dan kode_kategori sebagai *foreign key*.

Primary key adalah sebuah key yang bersifat unik yang terdapat di dalam sebuah tabel yang digunakan sebagai pembeda antara tiap *record* yang ada. Sedangkan *foreign key* merupakan sebuah *field* yang didapatkan dari sebuah tabel lain, dimana *foreign key* pada tabel A merupakan *primary key* pada tabel B.

Struktur dari tabel Barang ini adalah:

Tabel 4.1. Tabel Barang

Field	**Tipe**	**Ukuran**	**Keterangan**
Kode_Barang	Numeric		Kode barang
Kode_Kategori	Numeric		Kode kategori
Nama_Barang	Text	30	Nama barang
Jumlah_Barang	Numeric		Jumlah barang
Berat_Barang	Numeric		Berat barang
Harga_Beli	Currency		Harga pokok
Harga_Jual	Currency		Harga jual
Keterangan_barang	Memo		Info barang
File_Gambar	Text	30	Path file

❖ Tabel Cargo

Tabel ini digunakan untuk menyimpan data cargo yang terdiri dari *field* kode cargo, nama cargo, nama kontak cargo, alamat cargo, nomor telepon, nomor faks, kota, propinsi, dan keterangan singkat mengenai cargo. Tabel Cargo menggunakan kode_cargo sebagai *primary key*.

Struktur dari tabel Cargo ini adalah:

Tabel 4.2. Tabel Cargo

Field	**Tipe**	**Ukuran**	**Keterangan**
Kode_Cargo	Numeric		Kode cargo
Nama_Cargo	Text	30	Nama cargo
Contact_Cargo	Text	30	Nama kontak
Alamat_Cargo	Text	50	Alamat cargo
Telp_Cargo	Text	10	Nomor telepon
Fax_Cargo	Text	10	Nomor faks
City_Cargo	Text	30	Kota cargo
State_Cargo	Text	30	Propinsi cargo
Keterangan_Cargo	Memo		Info cargo

❖ Tabel Customer

Tabel ini digunakan untuk menyimpan data pelanggan yang terdiri dari *field* kode pelanggan, nama pelanggan, nama kontak pelanggan, alamat pelanggan, nomor telepon pelanggan, nomor faks pelanggan, kota pelanggan,

propinsi, keterangan barang, dan password yang digunakan untuk verifikasi pada saat pembelian barang di *web site.* *Primary key* dari tabel Customer adalah kode_customer yang berupa alamat e-mail, karena alamat e-mail untuk tiap pelanggan pasti tidak ada yang sama.

Struktur dari tabel Customer ini adalah:

Tabel 4.3. Tabel Customer

Field	Tipe	Ukuran	Keterangan
Kode_Customer	Text	50	Kode pelanggan
Nama_Customer	Text	30	Nama pelanggan
Contact_Customer	Text	30	Nama kontak
Alamat_Customer	Text	50	Alamat pelanggan
Telp_Customer	Text	10	Nomor telepon
Fax_Customer	Text	10	Nomor faks
City_Customer	Text	30	Kota pelanggan
State_Customer	Text	30	Nama propinsi
Keterangan_Cust	Memo		Info pelanggan
Password_Cust	Text	10	Password

❖ Tabel Detail_Beli

Tabel ini digunakan untuk menyimpan detail pembelian barang dari supplier. Adapun *field* yang terdapat di dalam tabel ini adalah nomor faktur, kode barang, jumlah pembelian, dan nilai sub total pembelian barang. *Primary key* dari tabel ini adalah gabungan dari no_faktur dan kode_barang, yang masing-masing merupakan *primary key* dari tabel Faktur dan tabel Barang.

Struktur dari tabel Detail_Beli ini adalah:

Tabel 4.4. Tabel Detail_Beli

Field	Tipe	Ukuran	Keterangan
No_Faktur	Numeric		Nomor faktur
Kode_Barang	Numeric		Kode barang
Quantity	Numeric		Jumlah beli
Subtotal	Currency		Sub total beli

❖ Tabel Detail_Jual

Tabel ini digunakan untuk menyimpan detail penjualan barang ke pelanggan. Adapun *field* yang terdapat di dalam tabel ini adalah nomor nota, kode barang, jumlah penjualan, dan nilai sub total penjualan barang. *Primary key* dari

tabel ini adalah gabungan dari kode_barang dan nomor_nota yang merupakan *primary key* dari tabel Barang dan tabel Nota Transaksi.

Struktur dari tabel Detail_Jual ini adalah:

Tabel 4.5. Tabel Detail_Jual

Field	Tipe	Ukuran	Keterangan
Kode_Barang	Numeric		Kode barang
Nomor_Nota	Numeric		Nomor nota
Quantity	Numeric		Jumlah jual
Subtotal	Currency		Sub total jual

❖ Tabel Faktur

Tabel ini digunakan untuk menyimpan *header* pembelian barang dari *supplier*. Adapun *field* yang terdapat di dalam tabel ini adalah nomor faktur, kode supplier, tanggal transaksi pembelian, dan nilai total pembelian barang. *Primary key* dari tabel ini adalah no_faktur, sedangkan yang menjadi *foreign key* pada tabel ini adalah kode_supplier.

Struktur dari tabel Faktur ini adalah:

Tabel 4.6. Tabel Faktur

Field	Tipe	Ukuran	Keterangan
No_Faktur	Numeric		Nomor faktur
Kode_Supplier	Numeric		Kode supplier
Tanggal_Beli	Date/time		Tanggal beli
Total	Currency		Total pembelian

❖ Tabel Kategori

Tabel ini digunakan untuk menyimpan kategori barang yang dijual oleh Toko Vektor Komputer. Adapun *field* yang terdapat di dalam tabel ini adalah kode kategori, nama kategori, dan keterangan dari kategori yang ada. *Primary key* dari tabel ini adalah kode_kategori.

Struktur dari tabel Kategori ini adalah:

Tabel 4.7. Tabel Kategori

Field	Tipe	Ukuran	Keterangan
Kode_Kategori	Numeric		Kode kategori
Nama_Kategori	Text	30	Nama kategori
Keterangan	Memo		Info kategori

❖ Tabel Kirim

Tabel ini digunakan untuk menyimpan tarif pengiriman barang antar kota per *cargo*. Adapun *field* yang terdapat di dalam tabel ini adalah kode cargo, kode kota tujuan (dari kode tujuan dapat diketahui nama kota tempat layanan), dan biaya pengiriman antar kota tersebut. *Primary key* dari tabel ini adalah gabungan dari kode_cargo dan kode_tujuan.

Struktur dari tabel Kirim ini adalah:

Tabel 4.8. Tabel Kirim

Field	**Tipe**	**Ukuran**	**Keterangan**
Kode_Cargo	Numeric		Kode cargo
Kode_Tujuan	Numeric		Kode tujuan
Harga	Currency		Tarif kirim

❖ Tabel Kota

Tabel ini digunakan untuk menyimpan nama kota yang dilayani oleh perusahaan pengiriman barang. Adapun *field* yang terdapat di dalam tabel ini adalah kode tujuan, nama kota, nama propinsi, dan keterangan mengenai kota tersebut. *Primary key* dari tabel ini adalah kode_tujuan.

Struktur dari tabel Kota ini adalah:

Tabel 4.9. Tabel Kota

Field	**Tipe**	**Ukuran**	**Keterangan**
Kode_Tujuan	Numeric		Kode kota
Nama_Kota	Text	30	Nama kota
Propinsi	Text	30	Nama propinsi
Keterangan	Memo		Info kota

❖ Tabel Nota_Transaksi

Tabel ini digunakan untuk menyimpan *header* penjualan barang ke pelanggan. Adapun *field* yang terdapat di dalam tabel ini adalah nomor nota, kode pelanggan, kode cargo, tanggal transaksi penjualan, dan nilai total transaksi penjualan barang. *Primary key* dari tabel ini adalah nomor_nota. Sedangkan kode_customer dan kode_cargo merupakan *foreign key*.

Struktur dari tabel Nota_Transaksi ini adalah:

Tabel 4.10. Tabel Nota_Transaksi

Field	Tipe	Ukuran	Keterangan
Nomor_Nota	Numeric		Nomor nota jual
Kode_Customer	Text	50	Kode pelanggan
Kode_Cargo	Numeric		Kode cargo
Tanggal_Jual	Date/time		Tanggal jual
Total	Currency		Total penjualan

❖ Tabel Supplier

Tabel ini digunakan untuk menyimpan informasi mengenai data supplier. Adapun *field* yang terdapat di dalam tabel ini adalah kode supplier, nama supplier, nama kontak supplier, alamat supplier, nomor telepon supplier, nomor faks supplier, kota, propinsi atau negara bagian, negara asal supplier, dan keterangan mengenai supplier. *Primary key* dari tabel ini adalah kode_supplier.

Struktur dari tabel Supplier ini adalah:

Tabel 4.11. Tabel Supplier

Field	Tipe	Ukuran	Keterangan
Kode_Supplier	Numeric		Kode supplier
Nama_Supplier	Text	30	Nama supplier
Contact_Supplier	Text	30	Nama kontak
Alamat_Supplier	Text	50	Alamat supplier
Telp_Supplier	Text	10	Nomor telepon
Fax_Supplier	Text	10	Nomor faks
City_Supplier	Text	30	Kota supplier
State_Supplier	Text	30	Nama propinsi
Country_Supplier	Text	30	Negara supplier
Keterangan_Sup	Memo		Info supplier

4.2 IMPLEMENTASI APLIKASI BACK-END

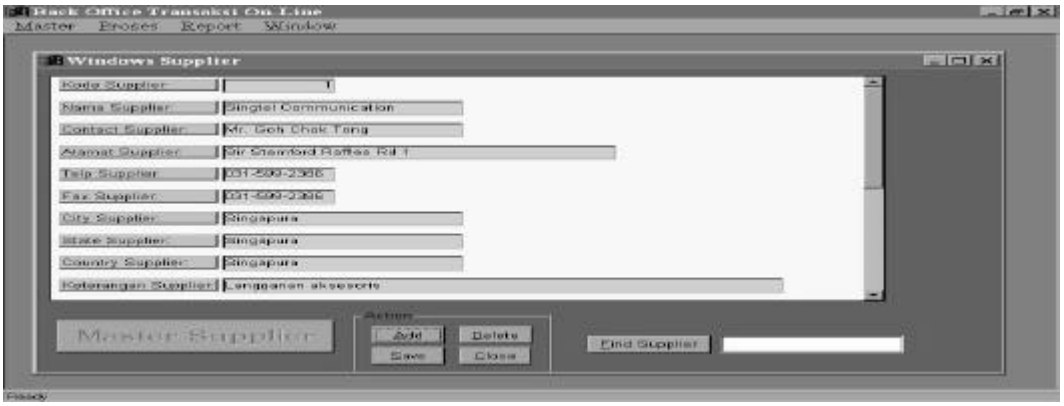
Pada bagian ini akan dibahas mengenai implementasi yang dilakukan pada aplikasi *back-end*. Aplikasi *back-end* ini bersifat lokal dan *offline*, dibuat dengan menggunakan perangkat lunak Microsoft Access sebagai file basis data, dan sebagai antar-muka-nya digunakan Power Builder. Untuk aplikasi *back-end* ini terdiri dari dua menu utama, yaitu:

1. MENU MASTER

Menu ini terdiri dari sub menu supplier, sub menu kategori, sub menu barang, sub menu cargo, dan sub menu kota propinsi.

a. SUB MENU SUPPLIER

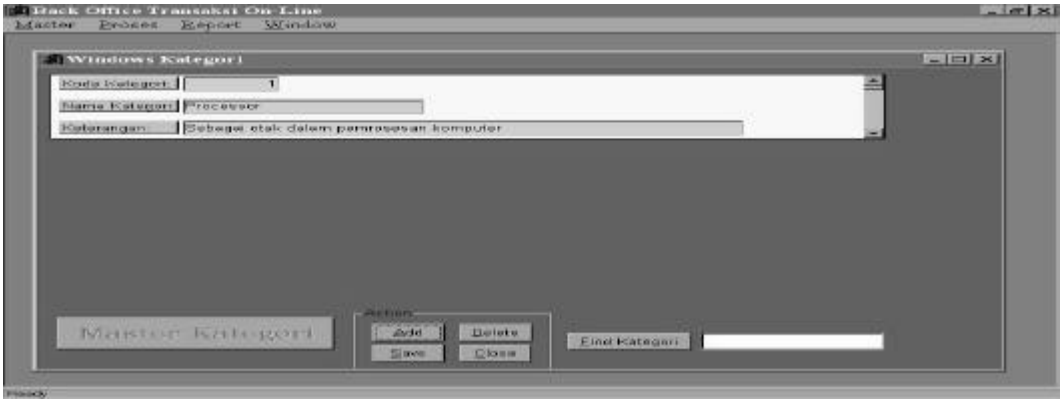
Sub menu ini berfungsi untuk memasukkan data supplier yang menyuplai produk-produk yang dijual oleh Toko Vektor Komputer.



Gambar 4.1. Master Supplier

b. SUB MENU KATEGORI

Sub menu ini berfungsi untuk menampung semua data mengenai kategori dari barang-barang yang akan dijual.



Gambar 4.2. Master Kategori

c. SUB MENU BARANG

Sub menu ini berfungsi untuk menampung semua data barang yang dijual.

Gambar 4.3. Master Barang

d. SUB MENU CARGO

Sub menu ini berfungsi untuk menyimpan data semua nama cargo yang melayani tiap daerah propinsi.

Gambar 4.4. Master Cargo

e. SUB MENU KOTA PROPINSI

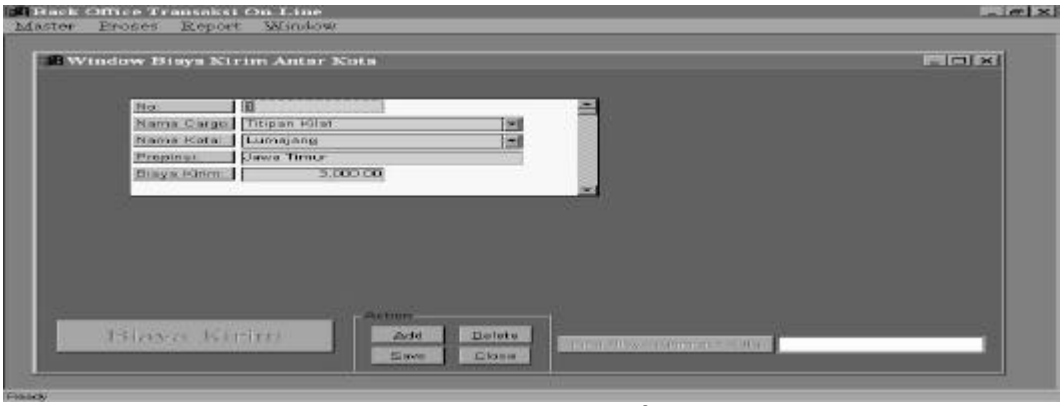
Sub menu ini berfungsi untuk menyimpan nama-nama kota yang telah mendapatkan perjanjian pengiriman barang dengan cargo yang ada.

Gambar 4.5. Master Kota Propinsi

2. MENU PROSES

a. SUB MENU TARIF KOTA

Sub menu ini digunakan untuk menyimpan tarif kiriman antar kota per jenis cargo.



Gambar 4.6 Menu Tarif Kota

b. SUB MENU TRANSAKSI INPUT

Sub menu ini digunakan untuk menyimpan semua jenis transaksi yang dilakukan dengan supplier.

Gambar 4.7 Menu Transaksi Input

4.3 IMPLEMENTASI ACTIVEX DATA OBJECT (ADO)

ActiveX Data Objects (ADO) merupakan tampilan yang lebih bersahabat (*friendly*) dari OLE-DB. Ini berarti, pengembang tidak perlu mengetahui segala hal tentang OLE-DB secara keseluruhan, karena ADO akan menghilangkan semua kompleksitas pemrograman tersebut dan memberikan programer suatu cara yang mudah dalam mengakses data pada setiap *data store* (*data store* mengacu pada setiap bentuk basis data). ADO merupakan suatu cara mendapatkan data dari dan ke dalam suatu *data store*, memudahkan pengembang dalam membaca suatu *record*, menemukan suatu *record*, dan melakukan *update* data. Sedangkan *record* merupakan sekumpulan data yang mengandung informasi dari tiap *field* (baris), sesuai dengan entry data yang di-*input*-kan.

Di bawah ini akan dijelaskan implementasi yang dilakukan terhadap ADO dalam kaitannya dengan aplikasi e-commerce yang memanfaatkan teknologi Active Server Pages.

4.3.1 CONNECTION OBJECT DAN RECORDSET

Untuk menggunakan ADO, yang pertama kali perlu diketahui adalah penggunaan ***connection object***. *Connection object* adalah suatu cara yang digunakan ADO untuk menyimpan informasi mengenai koneksi terhadap *data store*. Suatu *connection object* dapat dikodekan dengan cara berikut:

```
<%
Dim objconn
Set objconn = server.createobject ("adodb.connection")
%>
```

, sedangkan untuk menutup suatu *connection objec*t digunakan perintah **close**, yaitu:

```
objconn.close
```

Perlu diperhatikan, bahwa perintah tersebut belum benar-benar menghilangkan penggunaan *object* dari *memory*, sehingga untuk melepaskan *memory* menjadi suatu *free space*, *connection object* perlu diatur dengan perintah **nothing**, yaitu:

```
Set objconn = nothing
```

Hal lain yang perlu diketahui dalam ADO adalah yang disebut ***recordset***. *Recordset* adalah sekumpulan *record*, atau boleh juga dikatakan bahwa *recordset* adalah *record-record* itu sendiri. Sebuah *record* menyimpan sejumlah informasi yang berkaitan. Sehingga lebih tepat dikatakan bahwa *recordset* adalah sekumpulan *record* yang saling berelasi untuk tiap *field*-nya.

Secara umum, terdapat beberapa tipe dari *recordset*, yang kerap kali disebut dengan nama ***cursor type***, yaitu:

- *Updateable* dan non-*updateable recordset*, didefinisikan apakah pada *recordset* tersebut dapat atau tidak dapat dilakukan peng-*update*-an data.
- *Scrollable* dan *non-scrollable recordset*, menentukan apakah di dalam *recordset* tersebut dapat dilakukan *scroll* (pergeseran *record* ke *record* sebelumnya, sesudahnya atau hanya dibatasi untuk *move forward* saja).
- *Keyset* dan *non-keyset recordset*, untuk mengambil data di dalam suatu basis data, dimana suatu *keyset* digunakan untuk mengambil ***key*** yang unik dan menerima data yang aktual sesuai dengan *request* terhadap *record* tersebut.

- *Dynamic* dan *static recordset*, menentukan *record* mana yang tersedia di dalam *recordset* pada waktu yang dikehendaki. Sebuah *static recordset* hanya mengandung *record-record* yang telah ada pada saat *recordset* diciptakan, sedangkan *dynamic recordset* hanya mengatur sebagian dari *recordset* yang ada. Ini berarti bahwa jika ada *record* baru yang ditambahkan, dihapus, atau diubah oleh orang lain pada saat melakukan akses basis data, maka hasil perubahan yang terjadi akan langsung tersedia pada saat itu juga.

Sedangkan untuk aplikasi di dalam *web*, terdapat yang *disebut* ***ADO Recordset Type*** yang terdiri dari empat tipe, yaitu:

- *Forward Only*, merupakan tipe *default* dan suatu *non-scrollable recordset*.
- *Static*, hampir sama dengan *recordset Forward Only*, dengan perbedaan *scroll* sehingga dapat dilakukan pergeseran ke *record* yang sebelumnya (*move backward*).
- *Dynamic*, yaitu suatu *recordset* yang sangat dinamis, dimana dapat dilakukan penambahan, perubahan, dan penghapusan yang dilakukan oleh *user* lain. *Recordset* ini juga bersifat *scrollable*.
- *Keyset*, hampir sama dengan *dynamic recordset*, dengan pengecualian tidak dapat melihat *record* yang ditambahkan oleh *user* lain, walaupun perubahan pada *record* tersebut dapat dilihat setelah peng-*update*-an. Untuk setiap *record* yang telah terhapus akan menjadi tidak terakses (*inaccessible*).

Bagian terakhir dari *recordset* yang perlu diketahui adalah yang disebut ***Locking***, yang terdiri dari empat tipe sebagai berikut:

- *Read-only*, yang merupakan *default* dan tidak dilakukan prosedur *locking* karena perubahan data telah tidak dapat dilakukan.
- *Pessimistic*, yaitu *locking* yang sangat protektif, dimana pada *record* akan dilakukan *locking* sesaat setelah dilakukan perubahan data.
- *Optimistic*, dimana pada *record* hanya dilakukan *locking* bila data tersebut akan di-*update.*

- *Optimistic batch*, membolehkan perubahan terhadap beberapa *record* dan melakukan *update* secara sekaligus.

Jadi, apabila ingin melihat suatu *recordset* secara mendetail, maka di dalam suatu baris perintah *recordset* terdiri dari beberapa bagian, yaitu:

`Recordset.Open Source, ActiveConnection, CursorType, LockType, Options`

, dimana semua perintah tersebut dapat diperinci sebagai berikut:

- *Source* adalah darimana data tersebut tersebut berasal, *source* dapat berupa nama suatu tabel pada basis data, atau suatu *query*.
- *ActiveConnection*, identik dengan koneksi terhadap suatu *data store* atau basis data.
- *CursorType*, atau tipe *recordset*, merupakan konstanta yang telah ditentukan sebelumnya yang memberikan informasi yang terdiri dari:
  - *AdOpenDynamic*, memberikan suatu *recordset* yang *scrollable* dan *full dynamic*.
  - *AdOpenForwardOnly*, memberikan suatu *recordset* yang *updateable* dan *non scrollable*.
  - *AdOpenKeyset*, memberikan suatu *recordset* yang *scrollable.*
  - *AdOpenStatic*, memberikan suatu *recordset* yang bersifat *read-only* dan *scrollable*.

- *LockType*, terdiri dari konstanta-konstanta sebagai berikut:
  - *AdLockBatchOptimistic*, memberikan *lock* optimistik untuk *batch update.*
  - *AdLockOptimistic*, memberikan suatu *recordset* yang *updateable*, dimana *lock* akan ditempatkan sesaat sebelum dilakukan *update.*
  - *AdLockPessimistic*, memberikan suatu *recordset* yang *updateable*, dimana *lock* akan dilakukan sesaat setelah terjadi pengeditan.
  - *AdLockReadOnly*, memberikan suatu *recordset* yang bersifat *read-only*, dan tidak memperbolehkan *update.*

- *Options*, menspesifikasikan properti *source* yang diinterpretasikan, terdiri dari:
  - *AdCmdText*, mengindikasikan source menyimpan teks berisikan perintah, seperti perintah SQL.
  - *AdCmdTable*, mengindikasikan *source* menyimpan nama tabel.
  - *AdCmdStoreProc*, mengindikasikan *source* menyimpan nama dari suatu *stored procedure* atau *query*.
  - *AdCmdUnknown*, mengindikasikan *source* menyimpan suatu tipe yang tidak diketahui.

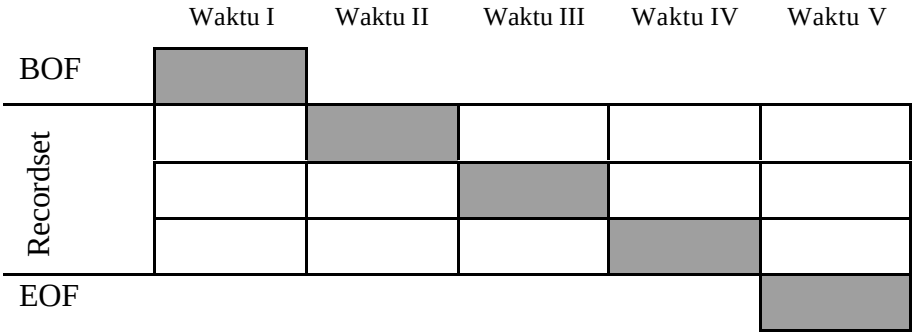
Di bawah ini adalah contoh suatu perintah *recordset* untuk membuka basis data, yaitu:

```
<%
     dim rsuser
     dim objConn
     set rsuser = server.CreateObject ("adodb.recordset")
     set objConn = server.CreateObject ("adodb.connection")
     objConn.Open "dsn=skripsiku"
     rsuser.Open "customer", objConn, 0, 2, 2
     if not rsuser.eof then
          'lakukan satu atau beberapa perintah di sini
     end if
%>
```

Segmen 4.1. Membuka File Basis Data

4.3.2 BEGIN OF FILE (BOF) DAN END OF FILE (EOF)

Di dalam pemrograman basis data, baik untuk aplikasi biasa maupun aplikasi *web*, selalu mempunyai ciri khusus, yaitu digunakannya tabel-tabel sebagai penyimpan data, dan adanya relasi di antara tabel-tabel yang ada untuk menghindari redundansi data. Setiap tabel memiliki jumlah *record* yang berbeda, tergantung dari jumlah data yang dimasukkan ke dalam tabel tersebut. Sebuah struktur tabel dapat digambarkan sebagai berikut:



Gambar 4.8. Struktur Tabel Dengan Pointer Bergerak Ke EOF

Pada saat pertama kali membuka suatu *recordset*, maka *current record* adalah terletak pada *record* pertama, dan bila dilakukan pemindahan *pointer* dengan menggunakan perintah *MoveNext*, maka *current record* tersebut akan berpindah pula. Bila *pointer* terus digerakkan hingga di akhir tabel, maka sesungguhnya *pointer* tersebut akan menunjuk kepada suatu *current record*, yang bukan merupakan suatu *record* yang valid. Dikatakan demikian karena *pointer* tersebut tidak menunjuk kepada suatu nilai tertentu yang mewakili isi *record*. Keadaan inilah yang disebut bahwa *pointer* tersebut telah menunjuk kepada bagian akhir dari suatu file (*end of file* / EOF). Properti EOF perlu diset untuk mengindikasikan bahwa *pointer* telah bergerak ke akhir dari suatu recordset. Hal yang sama terjadi pada bagian awal file (*begin of file* / BOF). Bila *pointer* terus diarahkan ke bagian awal file, maka pada akhirnya *pointer* tersebut akan menunjuk pada sebuah tempat sebelum *record* pertama terletak di dalam tabel. Oleh karena itulah, maka bila telah berada pada *record* terakhir, status EOF harus diset ke "**True**" sambil mengarahkan *current record* sesudah *record* yang terakhir.

Untuk memeriksa posisi EOF dapat dilihat pada segmen berikut:

```
<%
     dim rsusers
     dim sql
     set rsusers = server.CreateObject ("adodb.recordset")
     sql = "select * from customer "
     rsusers.Open sql, "dsn=skripsiku"
     if not rsusers.EOF then
          'lakukan satu atau beberapa perintah di sini
     end if
%>
```

Segmen 4.2. Memeriksa Keadaan End Of File

Segmen ini menjelaskan bahwa selama keadaan *pointer* pada *recordset* belum mencapai bagian akhir file, maka akan dilakukan suatu aksi yang terdapat di dalam *looping* **if … then**. Bila tabel tersebut merupakan suatu *recordset* kosong, maka segmen di atas tidak akan pernah menjalankan suatu aksi.

Apabila *recordset* tidak kosong, dalam pengertian ada data yang berada di dalam tabel tersebut dan telah dimasukkan ke dalam *recordset*, maka dapat dilakukan suatu pengambilan nilai *field* dari *recordset* tersebut. Contoh berikut ini akan mengambil nilai *field* **nama_cust** dari *recordset* **rsusers** :

```
<%
     nama_customer = rsusers ("nama_cust"))
%>
```

Segmen 4.3. Mengambil Field Dari Suatu Recordset

4.3.3 PERPINDAHAN POINTER DI DALAM RECORDSET

Untuk berpindah dari satu *record* ke *record* lainnya, dapat digunakan beberapa *method*. *Method* **MoveNext** digunakan untuk bergerak ke *record* berikutnya sebanyak satu *record*, **MovePrevious** digunakan untuk bergerak ke *record* sebelumnya sebanyak satu *record*, **MoveFirst** digunakan untuk memindahkan *pointer* ke awal *record*, dan **MoveLast** untuk memindahkan *pointer* ke *record* yang terakhir. Perpindahan antar *record* biasanya digunakan untuk mencari suatu nama yang spesifik, kemudian berdasarkan hasil pencarian tersebut diproses suatu aksi, atau melakukan suatu *looping* untuk menghasilkan suatu aksi. Segmen berikut ini menggunakan *method* **MoveNext** untuk memindahkan *pointer* dari awal hingga akhir *recordset*.

```
<%
     while not objRec.EOF
          'lakukan satu atau beberapa perintah di sini
          objRec.MoveNext
     wend
%>
```

Segmen 4.4. Memindahkan Pointer Ke Record Berikutnya.

4.3.4 PENAMBAHAN DATA PADA TABEL BASIS DATA

Untuk menambah data, maka langkah-langkah yang perlu diperhatikan adalah membuka koneksi dengan ADO dan menyimpan semua data-data tersebut ke dalam *record* yang baru. Segmen berikut ini akan memperlihatkan prosedur penambahan data pada tabel "**customer**".

```
<%
      dim rsuser
      dim objConn
      set rsuser = server.CreateObject ("adodb.recordset")
      set objConn = server.CreateObject ("adodb.connection")
      objConn.Open "dsn=skripsiku"

      rsuser.Open "customer",objConn,0,2,2
          rsuser.AddNew
          rsuser("kode_customer") = Request.Form ("Email")
          rsuser("nama_customer") = Request.Form ("login")
          rsuser("contact_customer") = Request.Form ("contact")
          rsuser("alamat_customer") = Request.Form ("address")
          rsuser("telp_customer") = Request.Form ("phone")
          rsuser("fax_customer") = Request.Form ("fax")
          rsuser("city_customer") = Request.Form ("city")
          rsuser("state_customer") = Request.Form ("state")
          rsuser("keterangan _customer") = Request.Form ("ket")
          rsuser("password_customer") = Request.Form ("password")
          rsuser.Update

     rsUser.Close
     objConn.Close
     set rsUser = nothing
     set objconn = nothing
%>
```

Segmen 4.5. Penambahan Data Pada Basis Data

Setelah dilakukan operasi ***AddNew*** untuk menyediakan *record* baru pada tabel, maka pengisian data dapat dilakukan. Setelah semua data telah terisi dengan benar, maka pada tabel tersebut akan dilakukan operasi ***update*** untuk *record* yang baru.

4.3.5 PENGGUNAAN KOMPONEN AD ROTATOR

Komponen ***Ad Rotator*** merupakan salah satu komponen dalam ADO yang memanfaatkan *server* dalam menciptakan *object*, yang dalam hal ini berupa suatu tempat yang berisikan sekaligus beberapa iklan yang dapat ditampilkan secara bergantian untuk menghemat penggunaan *space* pada *web site*, sekaligus untuk memperindah tampilan, dan mengurangi kejenuhan dari *client* yang sering memasuki *web site* yang sama. Adapun cara kerja dari komponen *Ad Rotaror* ini adalah dengan melakukan rotasi terhadap beberapa iklan yang ada secara sekaligus dengan memanfaatkan suatu file penjadwalan (*schedule file*). File penjadwalan ini berupa suatu file teks dengan dua bagian, dimana bagian pertama berisi parameter untuk menampilkan semua iklan yang ada, dan bagian kedua berisi parameter yang spesifik untuk tiap iklan yang akan mengalami rotasi tersebut. Secara singkat, penggunaan dari komponen *Ad Rotator* dapat dilihat pada segmen berikut ini:

```
<%
     dim VarIklan, ObjIklan
     set ObjIklan = server.CreateObject ("mswc.adrotator")
     VarIklan = ObjIklan.GetAdvertisement ("iklan.txt")
     Response.Write varIklan
%>
```

Segmen 4.6. Penggunaan Komponen Ad Rotator

4.4 IMPLEMENTASI ASP PADA HALAMAN HTML

Secara fisik, suatu file ASP hampir sama dengan file HTML. Perbedaan yang spesifik di antara keduanya adalah dalam lingkup pemrograman, dimana pada file Active Server Pages, biasanya terdapat pemrograman *web* yang tidak dimungkinkan tercipta di dalam file HTML. Sisi lainnya yang cukup berperanan penting adalah proses dalam menampilkan hasil eksekusi file, dimana pada file HTML, semua kode HTML langsung diterjemahkan di *client*, sedangkan pada file ASP, semua kode HTML beserta *scripting* yang ada akan dieksekusi terlebih dahulu di *server* untuk kemudian dikirimkan ke *client* dalam bentuk HTML.

4.4.1 MENAMBAHKAN DATA PADA BASIS DATA

Untuk menjadi anggota (*member*) di dalam *web site* ini, maka *client* perlu mengisikan beberapa informasi yang berkaitan dengan data pribadi *client*. Hal terpenting yang harus diisikan dengan benar adalah *identifier* (*login*) dan password karena kedua inputan inilah yang memegang peranan paling penting di dalam proses transaksi selanjutnya. Penambahan data pada basis data dapat dilihat pada contoh berikut, dimana akan dilakukan registrasi anggota baru di dalam *web site* seperti yang terdapat di dalam file **registered.asp**, yaitu:

```
<%
     dim rsuser
     dim objConn
     set rsuser = server.CreateObject ("adodb.recordset")
     set objConn = server.CreateObject ("adodb.connection")
     objConn.Open "dsn=skripsiku"

     rsuser.Open "customer",objConn,0,2,2
          rsuser.AddNew
          rsuser("kode_customer") = Request.Form ("Email")
          rsuser("nama_customer") = Request.Form ("login")
          rsuser("contact_customer") = Request.Form ("contact")
          rsuser("alamat_customer") = Request.Form ("address")
          rsuser("telp_customer") = Request.Form ("phone")
          rsuser("fax_customer") = Request.Form ("fax")
          rsuser("city_customer") = Request.Form ("city")
          rsuser("state_customer") = Request.Form ("state")
          rsuser("keterangan_customer")=Request.Form ("ket")
          rsuser("password_customer") = Request.Form ("password")
     rsuser.Update

          rsUser.Close
          objConn.Close
          set rsUser = nothing
          set objconn = nothing
%>
```

Segmen 4.7. Registrasi anggota baru

4.4.2 MEMERIKSA EKSISTENSI CLIENT DI DALAM WEB SITE

Suatu aplikasi *web* merupakan aplikasi yang *connectionless*. Pengertian *connectionless* di sini adalah, selama terjadinya proses (baik *browsing* maupun transaksi), maka sebenarnya *server* tidak pernah melakukan *tracking* terhadap seorang *client* dari satu *request* ke *request* lainnya. Setiap *request* yang datang

dari seorang *client* selalu diperlakukan oleh *server* layaknya *client* tersebut adalah *user* yang tidak dikenal sebelumnya. Tujuan dari hal tersebut di atas adalah untuk mengehemat penggunaan *memory*, karena dengan demikian tidak perlu lagi disediakan *memory* untuk mengingat setiap proses pada tiap *user*. Sebagai penggantinya, maka digunakanlah suatu variabel yang disebut ***session***, dimana satu *session* melambangkan satu *client*. Sebuah *session* bermula pada saat *client* melakukan *request* terhadap halaman pertama dari file ASP di dalam aplikasi.

Ada banyak kegunaan dari variabel *session*. Variabel *session* biasanya disimpan di dalam file global.asa dengan tujuan agar setiap terjadi *request* baru, maka akan terjadi pula sebuah *session* baru. Penggunaan variabel *session* juga menghemat penggunaan *memory* karena variabel *session* hanya dipakai secukupnya saja, demikian pula dengan penggunaan *memory*-nya. Kegunaan dari suatu variabel *session* antara lain untuk menyimpan suatu variabel global, maupun menyimpan suatu *identifier* tertentu yang akan terus dipakai selama terjadinya *request* pada aplikasi. Ini. Dua buah segmen di bawah ini memperlihatkan cara memeriksa eksistensi *client* pada *web site.*

```
<%
     dim loginname
     loginname = Request("loginname1")

     dim rsusers
     dim sql
     set rsusers = server.CreateObject ("adodb.recordset")
     sql = "select * from customer where
            kode_customer = '"+(loginname)+"'"
     rsusers.Open sql, "dsn=skripsiku"

     if not rsusers.EOF then
          'data yang dicari ada pada tabel
     end if
%>
```

Segmen 4.8. Memeriksa Eksistensi Client Melalui Pencarian Pada Basis Data

Di dalam segmen 4.8, terjadi runtunan proses, dimana pada saat *client* mengisikan suatu nama *client*, maka inputan tersebut akan disimpan di dalam suatu variabel yang bernama **loginname**. Kemudian selanjutnya dilakukan suatu koneksi basis data oleh ADO di dalam tabel "**customer**" pada file "**skripsiku**"

untuk membandingkan nama yang di-*input* dengan data pada basis data ADO. Sedangkan untuk segmen yang menggunakan suatu variabel *session* dapat dilihat pada contoh berikut ini:

```
<%
     if session("realname")="Mr.X" or session("realname")=" " then
          Response.Write "Maaf, Anda Belum Melakukan Login"
     End if
%>
```

Segmen 4.9. Memeriksa Eksistensi Client Dengan Variabel Session

Di dalam segmen ini, diperlihatkan bahwa di dalam global.asa telah disimpan suatu variabel *session* yang disebut "**realname**", dimana nilai variabel ini telah disetting dengan nilai "**Mr.X**". Jika *session* "**realname**" tersebut dibandingkan dengan nama *client* pada saat melakukan transaksi adalah sama, atau "**realname**" tersebut tidak berisikan suatu nilai sama sekali, berarti dapat dikatakan bahwa seorang *client* belum melakukan suatu *login*. Dengan demikian, dapat dilihat dengan jelas bahwa kegunaan variabel *session* sangat besar artinya dalam memeriksa eksistensi *client*, maupun untuk memeriksa hal-hal lainnya, seperti status pembayaran, dan lain-lain.

Setelah eksistensi seorang *client* telah dapat terdeteksi, maka langkah berikutnya adalah mengambil nilai "**password**" yang di-*input* oleh *client* tersebut. Password merupakan suatu kata kunci yang rahasia dan tidak boleh diketahui oleh orang lain, karena pasangan dari *identifier* seorang *client* dengan passwordnya adalah bersifat unik. Nama *login* yang diinput oleh seorang *client* merupakan alamat dari e-mail, karena alamat e-mail adalah unik, dan antara satu *client* dengan client lainnya tidak mungkin memiliki alamat e-mail yang sama. Password berisikan kata kunci sebanyak maksimal 10 huruf, dan pada saat pengisian password tidak dapat dilihat oleh *client* lain. Segmen berikut ini menjelaskan proses pengambilan nilai password yang diinput oleh seorang *client*:

```
<%
     dim password
     password = Request("password1")
%>
```

Segmen 4.10. Mengambil Nilai Password Yang Diinputkan

4.4.3 MELAKUKAN VALIDASI E-MAIL ADDRESS DAN PASSWORD

Bila *identifier* dan password seorang *client* telah terisi semua, maka langkah berikutnya adalah menguji kebenaran dari inputan *client* tersebut. *Identifier* seorang *client* yang berupa alamat e-mail akan dicocokkan dengan data yang terdapat di dalam basis data, sehingga dengan demikian perlu dilakukan suatu koneksi terhadap ADO dan membandingkan inputan dengan *recordset* nama yang ada di dalam basis data, seperti yang terdapat dalam file **checklogin.asp**.

```
<%
     dim loginname
     dim password
     loginname = Request("loginname1")
     password = Request("password1")

     dim rsusers
     dim sql
     set rsusers = server.CreateObject ("adodb.recordset")
     sql = "select * from customer where
           kode_customer = '"+(loginname)+"'"
     rsusers.Open sql, "dsn=skripsiku"

     if not rsusers.EOF then
          if UCase(password) = UCase(rsusers("password_customer"))
          then
               session("realname") = loginname
               session("realpassword") = password
               Response.Write "Login dan Password Anda Cocok"
          else
               Response.Write ("Password Tidak Cocok")
               Response.Write ("Harap Lakukan Login Ulang")
          end if
     else
          Response.Write  ("Login Name Tidak Ada Dalam Database ")
          Response.Write  ("Re-login atau Registrasi Member Baru")
     end if

     rsusers.Close
     set rsusers = nothing
%>
```

Segmen 4.11. Validasi Identifier dan Password

Apabila *identifier* yang diinputkan terdapat di dalam *recordset*, maka langkah berikutnya adalah membandingkan password yang diisi *client* tersebut sesuai dengan data password yang terdapat di dalam *recordset*, seperti yang ditunjukkan dalam segmen 4.11. Bila cocok, maka berarti *client* boleh melanjutkan transaksi. Bila tidak cocok, maka *client* harus mengisi ulang hingga benar, dan bila *identifier* yang diisikan tidak sama dengan yang terdapat di dalam

basis data, maka *client* diberikan kesempatan untuk melakukan *login* ulang atau registrasi sebagai *member* baru. Login yang tepat perlu dilakukan, karena transaksi hanya dapat dilanjutkan dengan pembayaran apabila *client* tersebut telah menjadi anggota (*member*) dalam aplikasi ini.

4.4.4 MENYIMPAN IDENTIFIER DAN PASSWORD DALAM SESSION

Setelah pengisian *identifier* dan password telah cocok, maka nilai dari kedua variabel tersebut akan disimpan di dalam variabel *session*. Nilai dari kedua variabel ini akan terus disimpan hingga transaksi selesai dilakukan, atau waktu yang digunakan untuk melakukan *browsing* telah habis, atau *session* yang dilakukan telah selesai. Untuk penyimpanan *identifier* dan password ke dalam variabel *session* dapat dilihat pada segmen 4.12.

```
<%
     session("realname") = loginname
     session("realpassword") = password
%>
```

Segmen 4.12. Menyimpan ID dan Password Ke Dalam Variabel Session

4.4.5 MENAMPILKAN MENU UTAMA

Pada saat seorang *client*, baik yang telah terdaftar maupun yang belum terdaftar masuk ke dalam *web site*, maka *client* tersebut sudah diberikan beberapa fasilitas yang berhubungan dengan kegiatan di dalam *web site* tersebut. Beberapa fasilitas tersebut antara lain kemampuan untuk melakukan *browsing*, registrasi, login, mencari kategori dan detail produk, serta melihat daftar *cargo* dan kota-kota yang dilayani oleh *cargo* yang ada. Menu utama yang terdapat di dalam file default.asp didesain dengan menggunakan Macromedia DreamWeaver yang menyediakan kemudahan desain *web* dalam bentuk tabel yang terbentuk oleh baris dan kolom, sehingga dengan demikian desain *web* akan kelihatan lebih rapi dan terstruktur.

4.4.6 MENAMPILKAN COUNTER PADA WEB PAGE

Untuk mengetahui jumlah kunjungan yang dilakukan oleh *client* terhadap *web page*, maka dapat dibuat suatu ***counter*** yang nilainya akan bertambah sebanyak satu (*increment*) untuk setiap kunjungan baru. Fungsi *counter* ini memanfaatkan object **FileSystemObject** yang terdapat pada komponen *scripting* (**scripting objects**). Di dalam suatu aplikasi komersial, fungsi *counter* biasanya ditujukan untuk mengetahui seberapa aktif *client* yang mengunjungi suatu *web page*, sekaligus mengetahui seberapa populer suatu *web site* di dalam internet. Fungsi *counter* di dalam Active Server Pages memanfaatkan suatu variabel global yang didefinisikan di dalam file **global.asa**, dan dideklarasikan di dalam prosedur **Application_On_Start** yang akan dieksekusi pertama kali pada saat aplikasi dijalankan. Nilai dari *counter* diambil dari suatu file teks yang terdapat di dalam server, yaitu isi dari file teks itu sendiri, untuk kemudian disimpan di dalam variabel global dan ditampilkan ke *client*. Segmen berikut ini akan menjelaskan prosedur pembuatan *counter* di dalam **global.asa**.

```
Dim fsVisitors   'FileSystemObject object
Dim fileVisitors 'TextStream object
Set fsVisitors =
     Server.CreateObject ("Scripting.FileSystemObject")
Set fileVisitors =
     fsVisitors.OpenTextFile("C:\Inetpub\wwwroot\Skripsiku\
     visitors.txt")
'Read counter value from text file
Application("counter") = fileVisitors.ReadLine
fileVisitors.Close
```

Segmen 4.13. Pendefinisian Fungsi Counter di Dalam Application_On_Start

Fungsi *counter* baru akan bertambah nilainya (mengalami *increment*) bila ada seorang *client* yang mengunjungi *web page*. Karena setiap *client* diwakili oleh suatu *session*, maka penambahan nilai *counter* hanya dilakukan pada variabel *session*, yang dalam hal ini didefinisikan di dalam prosedur **Session_On_Start**. Penambahan *counter* dapat dilihat pada segmen berikut ini:

```
‘lakukan penguncian variabel global
Application.Lock
‘menambahkan nilai counter
Application("counter") = CStr((Application("counter")) + 1)
Application.Unlock

Dim fsVisitors   'FileSystemObject object
Dim fileVisitors 'TextStream object
Set fsVisitors =
     Server.CreateObject("Scripting.FileSystemObject")
Set fileVisitors =
     fsVisitors.CreateTextFile("C:\Inetpub\wwwroot\Skripsiku\
     visitors.txt", true)
'menuliskan nilai counter pada  file teks
fileVisitors.Writeline(Application("counter"))
fileVisitors.Close
```

Segmen 4.14. Penambahan Nilai Counter Pada Session_On_Start

4.4.7 MENAMPILKAN KATEGORI PRODUK

Bila seorang *client*, baik yang sudah terdaftar maupun yang belum terdaftar mengakses *web page*, maka setiap *client* akan dihadapkan pada suatu fasilitas untuk melihat kategori produk yang ditawarkan. Dan dari kategori tersebut, juga dapat dispesifikasikan tipe-tipe produk yang dijual sesuai dengan kategorinya. Setiap *client*, walaupun belum meregistrasikan dirinya, tetap dapat melihat-lihat produk yang ditawarkan dengan tujuan apabila *client* tersebut tertarik dengan produk yang ditawarkan, maka untuk selanjutnya *client* tersebut dapat tergerak untuk melakukan registrasi dengan tujuan akhir agar *client* tersebut juga melakukan transaksi di dalam *web page* tersebut. Barulah pada saat hendak melakukan pembayaran, dilakukan suatu rutin untuk memeriksa status *client*. Bila belum terdaftar, maka *client* tidak dapat meneruskan transaksi.

Pada saat seorang *client* telah mengakses *web page*, maka *client* akan mendapatkan salam pembuka dari *server*. Setelah itu *client* dapat melakukan *browsing* untuk melihat kategori produk yang ditawarkan dalam *web page*. Segmen berikut ini menjelaskan bagaimana cara mengakses semua kategori produk yang ada di basis data, dan ditampilkan dalam bentuk *drop-down* sehingga dapat dipilih oleh *client*.

```
<FORM Action="kategori.asp" method="get" id=form1 name=form1>
<SELECT id=select1 name=kategori1 style="HEIGHT: 22px; WIDTH:
175px">
<%
     dim objConn
     dim objRec
     set objConn = server.CreateObject ("adodb.connection")
     set objRec = server.CreateObject ("adodb.recordset")

     objConn.Open "dsn=skripsiku"
     objRec.Open "kategori",objconn

     while not objRec.EOF
          Response.Write "<option>"
          Response.Write objRec ("nama_kategori")
          Response.Write "</option>"
          objRec.MoveNext
     wend
     Response.Write "<p>"

     objRec.Close
     objConn.Close
     set objRec = nothing
     set objconn = nothing

%>
</SELECT>
<INPUT type="submit" value="Pilih" id=submit1 name=submit1>
</FORM>
```

Segmen 4.15. Memilih Kategori Produk Dari Basis Data

Di sini diperlihatkan, bahwa untuk dapat menampilkan kategori produk yang ada, maka pertama kali yang dilakukan adalah membuat suatu koneksi dengan basis data yang bersesuaian (dalam hal ini adalah tabel "**kategori**"), sesudah itu dilakukan proses *looping* untuk mengambil tiap nama kategori yang ada untuk kemudian ditampilkan. Untuk kategori yang dipilih *client* akan dikirim ke *page* yang membutuhkan, seperti yang ditunjukkan dalam *form action* (dalam contoh di atas, *form action* mengarah ke kategori.asp). Setelah itu, hasil *request* akan disimpan di dalam variabel, sebagaimana yang tertera pada segmen di bawah ini:

```
<%
     dim nama_kat
     nama_kat = trim (Request.QueryString ("kategori1"))
%>
```

Segmen 4.16. Mengambil Dan Menyimpan Hasil Request

4.4.8 MENAMPILKAN PRODUK BERDASARKAN KATEGORI

Pada tahap berikutnya, *client* dapat memilih produk yang diinginkan, sesuai dengan kategori produk tadi. Untuk setiap kategori dapat terdiri dari beberapa jenis produk.

```
<FORM Action="Detail.asp" method="get" id=form1 name=form1>
<SELECT  id=select1  name=produk1  style="HEIGHT:  22px;  WIDTH:
175px">
<%

     dim objrec
     set objrec = server.CreateObject ("adodb.recordset")

     strquery = "select barang.nama_barang from barang,kategori
     where [barang].[kode_kategori] = [kategori].[kode_kategori]
     and [kategori].[nama_kategori]='"+(nama_kat)+"' "

     Response.Write "<p>"

     objrec.Open strquery, "dsn=skripsiku"

     while not objRec.EOF
          Response.Write "<option>"
          Response.Write objRec ("nama_barang")
          Response.Write "</option>"
          objRec.MoveNext
     wend
     Response.Write "<p>"

     objrec.Close
     set objrec = nothing
%>
</SELECT>
<INPUT type="submit" value="Detail" id=submit1 name=submit1>
</FORM>
```

Segmen 4.17. Menampilkan Produk Berdasarkan Kategori

Di dalam segmen ini, terdapat suatu variabel yang bernama **strquery** yang menampung data hasil suatu ***query*** pada basis data. Suatu *query* merupakan suatu perintah standar pada pemrograman basis data, yang digunakan untuk memilih dan menampilkan data-data tertentu sesuai dengan yang dikehendaki. Kegunaan *query* sangat banyak, antara lain untuk memudahkan *browsing* data pada basis data yang kompleks, maupun mengurangi kompleksitas itu sendiri, dan yang terutama adalah dapat memilih data beserta *field-field* yang ada sesuai dengan kebutuhan.

4.4.9 MENGARAHKAN CLIENT DENGAN REDIRECT METHOD

Sebuah *web page* yang baik biasanya terdiri dari beberapa file, baik file HTML maupun file ASP. Khusus untuk file ASP, karena file ini memiliki fasilitas pemrograman *web* dan menggunakan banyak variabel, baik variabel lokal maupun global, maka dalam hal ini cara pengaksesan di dalam ASP juga harus diperhatikan. Nilai variabel di dalam ASP biasanya dapat diambil dari nilai suatu variabel sebelumnya, maupun dari *response* (hasil *request*) terhadap ***form*** atau ***querystring***. Dengan demikian, perlu diatur suatu cara agar sebuah *web page* selalu diakses melalui gerbang utama. Pengertian gerbang utama di sini adalah suatu *page* yang dideklarasikan di awal session sebagai suatu *starter page*. Untuk mencegah terjadinya kemungkinan seorang *client* mengakses *page* lain sebelum mengakses *starter page*, maka dapat digunakan metode ***redirect*** yang berfungsi untuk mengarahkan seorang *client* ke *page* tertentu yang dikehendaki oleh administrator. Fungsi lain dari metode *redirect* adalah mengarahkan *client* ke suatu page lain bila suatu saat terjadi renovasi file pada *server*, sehingga dengan demikian seluruh aplikasi dapat dialihkan sementara hingga perbaikan file selesai dilakukan. Untuk memastikan aplikasi dan semua variabel yang ada berjalan dengan baik, maka metode *redirect* didefinisikan di awal *session* di dalam global.asa.

```
STARTPAGE = "/Skripsiku/Default.asp"
CURRENTPAGE = Request.ServerVariables ("SCRIPT_NAME")
IF STRCOMP(STARTPAGE,CURRENTPAGE,1) THEN
     Response.Redirect ("http://centaur" & startpage)
     Session.Timeout = 20
     session("Start") = Now
END IF
```

Segmen 4.18. Mengarahkan Client ke Page Tertentu

Pertama kali, akan diciptakan suatu variabel yang mendefinisikan *path* bagi *starter page*. Sesudah itu, nilai variabel akan dibandingkan dengan *page* pertama yang diakses oleh *client*. Bila hasil perbandingan ternyata tidak sama, maka *client* akan diarahkan langsung ke *starter page*.

4.4.10 MELIHAT BARANG YANG TELAH DIBELI

Untuk setiap produk yang ingin dibeli dan telah dimasukkan di dalam *cart* dapat dilihat beberapa *property* produk tersebut, antara lain: nama produk, jumlah stok produk, berat produk, harga jual, dan informasi singkat mengenai produk tersebut. Nama produk merupakan identifikasi produk yang dipilih, jumlah stok digunakan untuk menghitung parameter jumlah barang yang dibeli, dimana jumlah pembelian adalah maksimal sama dengan jumlah stok produk yang dipilih, berat produk digunakan untuk mempertimbangkan biaya pengiriman barang per kilogram per jasa pengiriman. Setiap jasa pelayanan pengiriman barang memperlakukan biaya pengiriman yang berbeda-beda. Harga jual digunakan untuk memberikan informasi harga kepada *client* agar dapat menghitung besarnya jumlah biaya yang harus dikeluarkan untuk tiap produk dan nantinya akan ditambahkan dengan besarnya biaya pengiriman, sedangkan informasi singkat berisikan informasi sekilas mengenai produk yang ditawarkan. Status barang yang telah dibeli dapat dilihat pada akhir transaksi, dimana setelah terjadi pembayaran dengan menggunakan kartu kredit yang dianggap valid, maka akan terjadi pengurangan stok barang di basis data.

4.4.11 VALIDASI KARTU KREDIT

Setelah selesai melakukan transaksi, maka tahap akhir yang perlu dilakukan oleh seorang *client* adalah membayar produk yang dibelinya dengan menggunakan kartu kredit. Di dalam aplikasi yang dibuat ini, pembayaran dengan kartu kredit dapat dilakukan dengan tujuh jenis kartu kredit yang berbeda, antara lain adalah Visa, Master Card, American Express, Diners Club, Discover, enRoute, dan JCB. Ketujuh jenis kartu kredit tersebut mempunyai ciri khusus yang membedakannya satu dengan yang lain. Misalnya, untuk jenis Visa memiliki digit sebanyak 13 atau 16, dan menggunakan angka 4 sebagai digit awal, jenis Master memiliki digit 16, dengan awalan 51, 52, 53, 54, dan 55, jenis Amrican Express memiliki panjang 15 digit, dengan digit awal 34, atau 37, jenis Diners Club memiliki panjang 14 digit dengan digit awal 300, 301, 302, 303, 304,

305, 36, dan 38. Demikian pula untuk jenis kartu kredit lainnya yang juga memiliki ciri-ciri khusus yang berbeda satu dengan lainnya. Untuk melakukan pengecekan dengan kartu kredit ini, maka digunakan dua buah script, yaitu dengan nama **checkcc.asp** dan **checkcc.inc**. Di dalam checkcc.asp terdapat prosedur untuk melakukan validasi kartu kredit, sedangkan file checkcc.inc merupakan suatu *file include* yang lebih menyerupai sebuah unit yang berisikan rutin-rutin pengecekan. Rutin dari file checkcc.inc dapat dilihat pada segmen berikut ini:

```
<%
function trimtodigits(tstring)
'removes all chars except of 0-9
     s=""
     ts=tstring
     for x=1 to len(ts)
          ch=mid(ts,x,1)
          if asc(ch)>=48 and asc(ch)<=57 then
          s=s & ch
          end if
     next
  trimtodigits=s
end function


function checkcc(ccnumber,cctype)
     ctype=ucase(cctype)
     select case ctype
     case "V"
          cclength="13;16"
          ccprefix="4"
     case "M"
          cclength="16"
          ccprefix="51;52;53;54;55"
     case "A"
          cclength="15"
          ccprefix="34;37"
     case "C"
          cclength="14"
          ccprefix="300;301;302;303;304;305;36;38"
     case "D"
          cclength="16"
          ccprefix="6011"
     case "E"
          cclength="15"
          ccprefix="2014;2149"
     case "J"
          cclength="15;16"
          ccprefix="3;2131;1800"
     case else
          cclength=""
          ccprefix=""
```

```
end select

prefixes=split(ccprefix,";",-1)
lengths=split(cclength,";",-1)
number=trimtodigits(ccnumber)

prefixvalid=false
lengthvalid=false

for each prefix in prefixes
     if instr(number,prefix)=1 then
          prefixvalid=true
     end if
next

for each length in lengths
     if cstr(len(number))=length then
          lengthvalid=true
     end if
next

result=0

if not prefixvalid then
     result=result+1
end if

if not lengthvalid then
     result=result+2
end if

qsum=0

for x=1 to len(number)
     ch=mid(number,len(number)-x+1,1)
     'response.write ch
     if x mod 2=0 then
          sum=2*cint(ch)
          qsum=qsum+(sum mod 10)
          if sum>9 then
               qsum=qsum+1
          end if
     else
          qsum=qsum+cint(ch)
     end if
next

'response.write qsum
if qsum mod 10<>0 then
     result=result+4
end if

if cclength="" then
     result=result+8
end if
```

```
     checkcc=result

end function
%>
```

Segmen 4.19. Validasi Kartu Kredit

4.4.12 MELAKUKAN UPDATE DATA DALAM TABEL

Setelah validasi kartu kredit selesai dilakukan, berarti *client* telah selesai melakukan pembelian sejumlah *unit* barang dan barang siap dikirimkan dengan menggunakan *cargo* yang dipilih. Untuk proses akhir, secara otomatis *server* akan mengadakan peng-*update*-an barang, dimana akan terjadi pengurangan stok untuk tiap barang yang dibeli dan nilai yang baru akan disimpan di dalam basis data. Kemudian hasil dari proses transaksi akan disimpan dalam bentuk file di *server*, sehingga untuk selanjutnya file tersebut akan diproses dan produk yang dibeli siap untuk dikirimkan sesuai dengan jenis *cargo* yang dipilih.