

BAB II

LANDASAN TEORI

Dalam merancang dan mengimplementasikan sistem ini serta pembuatan tugas akhir ini di gunakan beberapa landasan teori, yaitu :

2.1 Visual Basic for Application (VBA)

Visual Basic for Application atau lebih dikenal dengan VBA adalah bahasa pemrograman yang *syntax*-nya hampir menyerupai Visual Basic dan memiliki beberapa tambahan kemampuan untuk membantu programmer dalam mengembangkan aplikasi yang membutuhkan kondisi khusus dengan lebih mudah. VBA didisain agar aplikasi dapat dikustomisasi dengan cara mengubah cara kerja dan tampilan aplikasi, atau bahkan menambahkan fitur-fitur yang benar-benar baru sesuai dengan kebutuhan. VBA merupakan teknologi pengembangan *customizable application* yang sangat tepat, agar dapat dikustomisasi dan diintegrasikan dengan data dan sistem yang ada, serta aplikasi lain yang juga ber-VBA.

VBA menyediakan suatu lingkungan pengembangan terintegrasi (*integrated development environment*), seperti halnya pemrograman dengan menggunakan visual basic, termasuk tampilan projek, properti dan alat bantu *debugging*. VBA juga mendukung form-form Microsoft untuk pengembangan *dialog boxes*, dan *ActiveX® Controls* untuk pengembangan user interface secara cepat. Dengan terintegrasinya secara langsung ke dalam aplikasi induk, VBA menawarkan keunggulan terhadap kecepatan, antara lain, kinerja proses, integrasi

yang kuat dengan aplikasi induk dan kemampuan untuk membuat solusi tanpa menggunakan alat bantu tambahan. (Hart-Davis, 2000).

VBA menawarkan keuntungan-keuntungan bagi pengembang, manajer sistem informasi, dan pengguna akhir :

Keuntungan bagi pengembang, antara lain, (1) aplikasi yang ber-VBA merupakan aplikasi *open-system*, melalui model obyek, dan komponen berbasis Active-X, akan dapat berguna bagi para pengembang untuk menggunakan obyek-obyek atau komponen-komponen tersebut dalam membangun suatu solusi yang dikustomisasi, (2) kode-kode dapat digunakan kembali, karena kesamaan dari dasar bahasa pemrograman, dalam hal ini adalah Visual Basic, (3) VBA memungkinkan kustomisasi aplikasi untuk menyediakan solusi yang dikhususkan pada kebutuhan pelanggan tertentu, (4) dengan terus meningkatnya keberadaan aplikasi-aplikasi ber-VBA, pengembang sekarang dapat mengintegrasikan aplikasi-aplikasi ini untuk berbagi data dan informasi secara lebih mudah dan cepat, dan (5) VBA memungkinkan pengembang untuk membangun solusi dengan mengintegrasikan dari aplikasi-aplikasi yang telah ada dan dari pengembang yang berbeda-beda, berdasarkan pada keterkaitan fungsionalitas dari aplikasi-aplikasi tersebut dalam memenuhi kebutuhan pelanggan, sehingga dapat menekan biaya pengembangan.

Keuntungan yang didapat oleh manajer sistem informasi, dengan adanya aplikasi ber-VBA, antara lain, (1) adanya kemungkinan untuk membeli aplikasi paket ber-VBA, daripada mengembangkan aplikasi, dalam memenuhi kebutuhan bisnis tertentu, (2) hambatan dari tuntutan pengguna akhir terhadap aplikasi akan dapat dikurangi dengan adanya komponen yang dapat digunakan kembali, dan

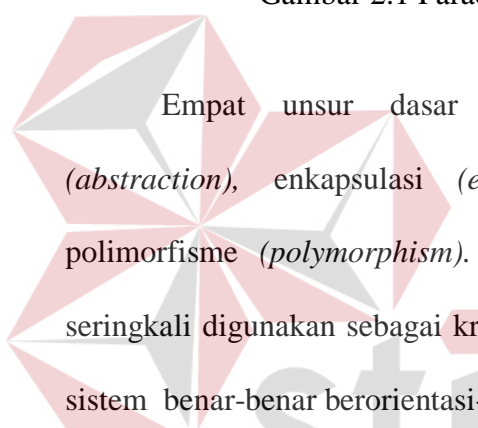
memenuhinya dalam waktu yang relatif cepat, (3) ketergantungan pelanggan terhadap keberadaan pengembang aplikasi ber-VBA dapat dikurangi, dan (4) kebutuhan biaya pelatihan bagi pengembang internal perusahaan dapat dikurangi.

Sedangkan Keuntungan yang didapat oleh pengguna akhir dengan adanya aplikasi ber-VBA, antara lain, (1) solusi akan dihasilkan lebih cepat, karena terintegrasinya VBA secara langsung dengan aplikasi induk, (2) pengguna dapat langsung mengkustomisasi aplikasi dengan solusi mereka, (3) pengguna telah mengetahui jalannya aplikasi dan solusinya, sehingga lebih sedikit pelatihan yang diperlukan, dan (4) keikutsertaan pengguna dalam mendisain solusi, misalnya, tampilan, laporan dan dokumen yang mereka inginkan. (Cummings, Steve, 2001).

2.2. Object Oriented Programming (OOP)

Dalam pemrograman berbasis *Object Oriented Programming* (OOP), sebuah program dibagi menjadi bagian-bagian kecil yang disebut dengan obyek. Setiap obyek memiliki entiti yang terpisah dengan entiti obyek-obyek lain dalam lingkungannya. Obyek-obyek yang terpisah ini dapat diolah sendiri-sendiri, dan setiap obyek memiliki sekumpulan sifat dan metode yang melakukan fungsi tertentu sesuai dengan yang telah kita programkan padanya. (Adi Kurniadi, 2000).

Gambar 2.1 Paradigma pemrograman berorientasi pada obyek



Empat unsur dasar sistem berorientasi-obyek adalah abstraksi (*abstraction*), enkapsulasi (*encapsulation*), inheritansi (*inheritance*), dan polimorfisme (*polymorphism*). Ini semua merupakan persyaratan kunci dan seringkali digunakan sebagai kriteria evaluasi sewaktu menentukan apakah suatu sistem benar-benar berorientasi-obyek.

Abstraksi adalah teknik yang digunakan oleh kita semua dalam mengelola kompleksnya informasi yang kita himpun setiap hari. Dengan demikian kita dapat mengenali hal-hal yang mirip dan mengabaikan perbedaanya, memikirkan hal-hal yang sifatnya umum dan bukan hal-hal yang khusus, dan memandang sesuatu sebagaimana adanya tanpa memikirkan sebab-musababnya. Dengan menggunakan abstraksi, maka dapat memusatkan perhatian pada obyek aplikasi dan bukan pada implementasi. Sehingga dapat memisahkan rancangan dari aspek teknologi pemrosesan komputer dan membuat pengguna dapat turut ambil bagian di dalamnya. Pengguna, seringkali pakar materi subyek, menjadi partisipan kunci dalam proses rancangan.

Enkapsulasi seringkali dianggap sebagai “ penyembunyian informasi ” ini merupakan kemampuan sebuah obyek kelas untuk membatasi akses *client* ke representasi internal obyek kelas (data dan fungsi). Salah satu metodenya adalah anggota-anggota kelas privat, yang mana enkapsulasi dilakukan dengan mendefinisikan variable-variabel dan fungsi-fungsi dari sebuah kelas sebagai privat. Dengan anggapan bahwa fungsi-fungsi anggota kelas didefinisikan sebagai publik yang secara internal memanipulasi variable-variabel dan fungsi-fungsi anggota privat. Dalam istilah berorientasi-obyek, metode-metode public dari sebuah kelas merupakan abstraksi dari representasi privat kelas. Hasilnya adalah bahwa representasi privat di enkapsulasikan.

Inheritansi adalah sebuah teknik untuk meniadakan *Redundancy* (kelebihan yang tidak perlu), membuat kode dapat digunakan kembali dengan menggeneralisasi kode umum dalam kelas induk dan kemudian mengembangkan kode sembarang yang sifatnya khusus dalam subkelas. Obyek apapun yang termasuk dalam subkelas secara otomatis menggunakan kembali seluruh perilaku dan properti kelas induk.

Sedangkan polimorfisme adalah sebuah teknik dimana dua kelas atau lebih bisa saja memiliki perilaku dengan nama sama dan maksud dasar yang sama (antarmuka) namun implementasinya berbeda. Dengan menggunakan kata kunci *Implements* yang diikuti oleh nama antarmuka yang sah, sebuah kelas dapat memberikan implementasinya pada semua properti dan metode yang telah didefinisikan dalam sebuah antarmuka. Karena sebuah obyek adalah wujud dari sebuah kelas spesifik, ia mendukung antarmuka-antarmuka yang diimplementasikan oleh kelas ke yang sesuai. (Deborah-Kurata, 1995).

2.3. ADO dan ADOX (Siebold, 2001)

Microsoft® ActiveX® Data Objects (ADO) memungkinkan aplikasi klien untuk dapat mengakses dan memanipulasi data dari database server melalui penyedia OLE DB. Keunggulan ADO adalah kemudahan dalam penggunaan, kecepatan, penggunaan memori yang rendah, dan membutuhkan kapasitas yang sedikit. ADO merupakan pendukung utama dalam pengembangan aplikasi klien/server dan berbasis Web. Karena ADO digunakan dalam pengaksesan data dari berbagai sumber, dibutuhkan pemahaman konsep dasar sistem manajemen relasional database, konsep dasar online analytical processing (OLAP), dan dasar internet dan protokol internet, dalam mengimplementasikannya. ADO adalah salah satu bagian dari strategi Microsoft Universal Data Access (UDA), dan digunakan bersama dengan teknologi OLE DB. OLE DB berbasis pada Microsoft Component Object Model (COM). Karenanya, pemahaman terhadap COM juga akan sangat membantu dalam memahami beberapa konsep ADO lebih dalam.

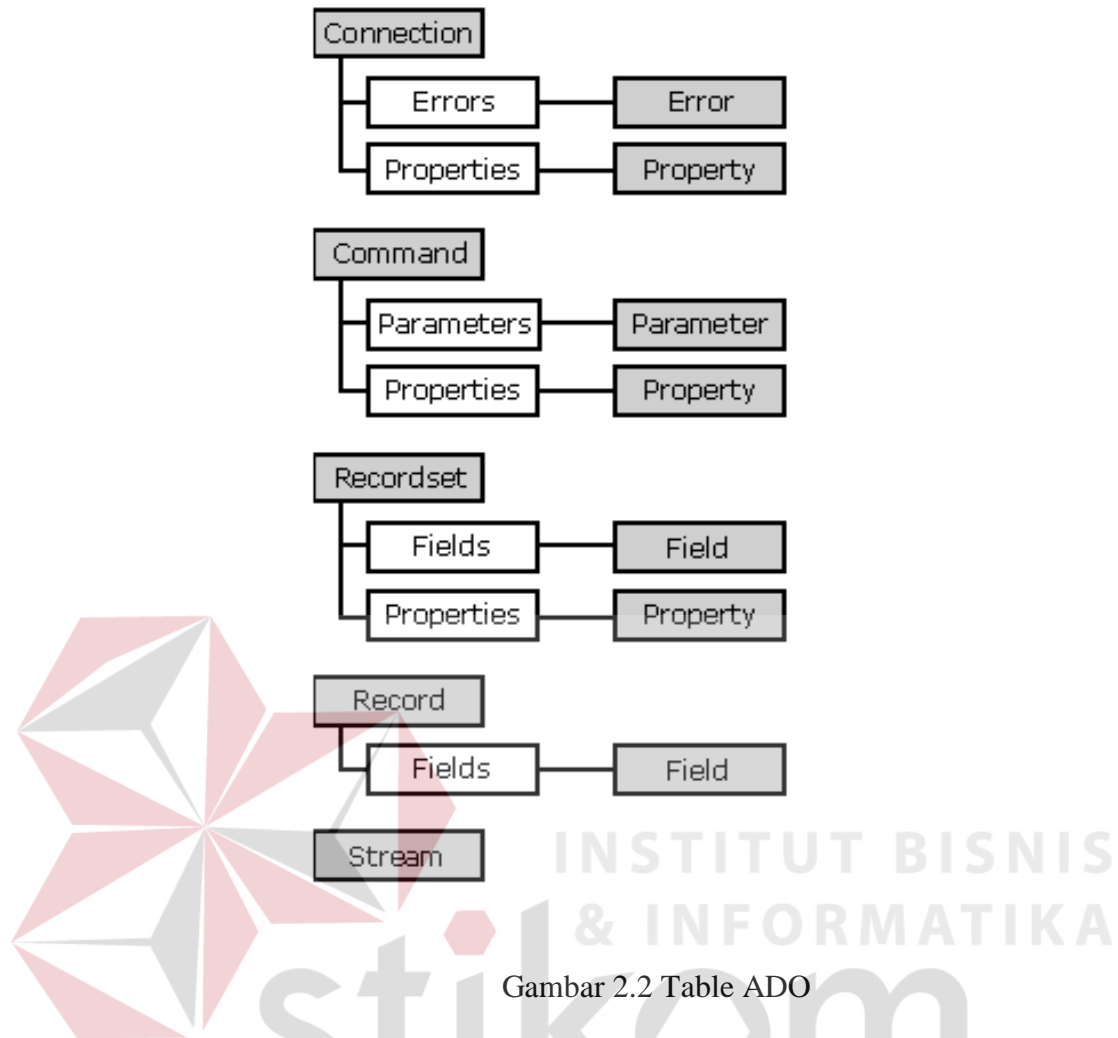
Universal Data Access (UDA) adalah strategi Microsoft untuk menyediakan akses terhadap informasi dari suatu perusahaan. Ide dasar dari UDA adalah untuk dapat mengakses tiap data dari tempat data tersebut berada secara efisien, lebih daripada proses untuk memindahkan data untuk disentralisasikan ke data store. UDA adalah suatu data store, alat bantu, dan bahasa yang independen. UDA menawarkan suatu komponen tingkat tinggi, antarmuka yang mudah digunakan, dan komponen tingkat rendah, antarmuka dengan performansi tinggi secara praktis untuk tiap data store yang ada. UDA dapat digunakan secara fleksibel untuk mengintegrasikan data store yang berbeda dan digunakan pada alat bantu, aplikasi, dan platform untuk menciptakan solusi yang dibutuhkan.

Microsoft® Data Access Components (MDAC) SDK berisi komponen-komponen yang dibutuhkan oleh Universal Data Access untuk dapat digunakan. Teknologi ini menyediakan kerangka dasar untuk tujuan umum dalam mengakses data pada sistem operasi Microsoft® Windows. Ada tiga teknologi utama dalam MDAC. ActiveX Data Objects (ADO) yang merupakan komponen tingkat tinggi, antarmuka yang mudah digunakan dalam mengakses OLE DB. OLE DB merupakan komponen tingkat rendah antarmuka terhadap berbagai data store dengan performansi tinggi. ADO dan OLE DB dapat digunakan untuk data relasional (tabular) dan nonrelasional (hirarki). Dan yang terakhir adalah Open Database Connectivity (ODBC) yang merupakan komponen tingkat rendah, antarmuka dengan performansi tinggi yang didisain secara khusus untuk data store relasional.

ADO menyediakan suatu layer abstraksi antara klien atau aplikasi middle-tier dan antarmuka tingkat rendah OLE DB. ADO menggunakan sekumpulan kecil obyek otomasi yang memungkinkan untuk berhubungan dengan OLE DB secara sederhana dan efisien. Antarmuka ini menjadikan ADO pilihan yang tepat bagi pengembang, yang ingin mengakses data tanpa harus mempelajari COM dan OLE DB secara detail, dengan menggunakan bahasa tingkat yang lebih tinggi, seperti Visual Basic dan bahkan VB Script.

2.3.1 ADO: ActiveX Data Objects (Siebold, 2001)

ADO memungkinkan klien suatu aplikasi untuk mengakses dan memanipulasi data melalui provider OLE DB. ADO berisi obyek-obyek yang menghubungkan suatu sumber data dan dapat membaca, menambah, membaharui, ataupun menghapus data.



Gambar 2.2 Table ADO

ADO menjelaskan hubungan (*connection*) pemakai sumber data khusus terhadap suatu obyek. Yang berbeda dari DAO terdapat pada obyek *workspace* bagi pemakai dan obyek database menjelaskan tentang sumber data. Perintah (*command*) obyek ADO sama dengan *querydef* pada obyek DAO dimana kedua obyek dapat digunakan pada saat menjalankan pernyataan SQL pada suatu sumber data. Demikian juga, *Recordset* dalam kedua obyek ADO dan DAO dapat digunakan untuk melihat isi tabel ataupun hasil dari pernyataan SQL.

2.4. Microsoft Visual Basic 6.0

Pada tahun 1980-an sistem operasi DOS cukup populer dikalangan pemakai PC karena di dalamnya disertakan bahasa BASIC yang dikenal dengan QBASIC (*QuickBasic*). Diera *windows*, Microsoft menciptakan Visual Basic yang merupakan pengembangan dari bahasa pemrograman BASIC (*Beginners All-purpose Symbolic Instruction Code*). Kemudian berkembang versi – versinya sampai pada versi Microsoft Visual Basic 6.0. Visual Basic adalah bahasa pemrograman berbasis windows, yang sangat interaktif dan lebih *compatible* dengan sistem operasi windows yang umum digunakan untuk menghasilkan aplikasi yang bekerja pada sistem operasi Windows 95 atau Windows NT 4. Banyak keunggulan dimilikinya, yang paling menonjol adalah kemudahan pemakaian. Microsoft Visual Basic 6.0 menyediakan fasilitas yang memungkinkan untuk menyusun sebuah program dengan memasang objek-objek grafis dalam sebuah *form* dengan penggunaan aplikasi GUI (*Graphical User Interface*).

Pada bidang *database* Microsoft Visual Basic 6.0 memberikan kemudahan seperti Oracle, Microsoft SQL Server, AS400, MySQL, DB2 dan lain sebagainya. Dengan adanya ODBC (*Open Database Connectivity*), DAO (*Data Access Object*), ADO (*ActiveX Data Object*) maupun RDO (*Remote Data Object*) Microsoft Visual Basic 6.0 memberikan lebih banyak kemudahan. Microsoft Visual Basic 6.0 menyediakan tiga macam *interface* yang bisa digunakan untuk merancang aplikasi sesuai dengan kebutuhan. Selain MDI (*Multi Document Interface*) dan SDI (*Single Document Interface*), kini tersedia *Explorer Document Interface* dengan gaya seperti Windows Explorer.

