BAB II

KERANGKA TEORITIS DAN PERUMUSAN HIPOTESA

2.1 . RFID (RADIO FREQUENCY IDENTIFICATION)

RFID atau *Radio Frequency Identification* merupakan suatu sistem yang menggunakan gelombang radio dalam range frequency tinggi yang dapat menyimpan atau menerima data secara jarak jauh dengan menggunakan suatu piranti yang bernama RFID *transponder*. Dapat juga di definisikan sebagai sebuah pengembangan teknologi pengambilan data secara otomatis atau pengenalan atau identifikasi objek (Kanzeller, 1999). Sistem RFID pada umumnya menggunakan dua bagian komponen utama yaitu :

1. RFID Tag / RFID Transponder

RFID *tag* di pasaran saat ini terdiri dari berbagai bentuk yg di desain menyesuaikan pemakaian di lapangan. RFID *tag* berfungsi sebagai transponder yang berisikan data dengan menggunakan frekuensi 125 Khz (Saputra, Cahyadi 2010). Seperti pada gambar di bawah ini.

Gambar 2.1
RFID Tag / transponder

2. RFID Reader/RFID sensor

Seperti halnya RFID *tag,* RFID reader juga terdiri dari berbagai bentuk yg di desain oleh banyak pembuat. Di bawah ini dua dari sekian banyak RFID reader yg ada di pasaran saat ini.

Gambar 2.2
RFID Reader / sensor

2.1.1 . Tipe RFID

Teknologi RFID sudah sangat banyak digunakan untuk aplikasi industri saat ini baik yang berskala besar maupun kecil. terdapat tiga tipe RFID *tag* yaitu :

2.1.1.1 RFID *Tag* Pasif

RFID Tipe ini tidak menggunakan tenaga baterai, sehingga *chip* tersebut dapat dipergunakan selama-lamanya. Cara kerjanya adalah dengan memanfaatkan energi yang dipancarkan oleh *reader* ketika kita mendekatkan *tag* nya. Kelemahan dari *tag* tipe ini adalah jarak baca data yang hanya berjarak beberapa cm dari *reader* nya. Untuk meningkatkan jarak baca pada *chip* tipe ini adalah dengan menambahkan antena *external* pada RFID tag tersebut .

2.1.1.2 RFID *Tag* Aktif

RFID *tag* aktif, dimana *tag* tersebut diberi tenaga dengan menggunakan baterai. Daya yang dibutuhkan oleh RFID *tag* sangat kecil, sehingga *tag* yang menggunakan baterai tersebut dapat bertahan cukup lama (sampai baterai habis). Bentuk RFID *tag* aktif umumnya mempunyai ketebalan beberapa milimeter untuk tempat baterainya. Sedangkan ukurannya bervariasi, ada yang sebesar uang logam , ada yang berupa gantungan kunci, ada yang berupa kartu nama, dan lain-lain. Kelebihan dari *tag* aktif adalah jarak jangkauan untuk alat pembacaan data yaitu dapat membaca data yang terdapat dalam *tag* dari jarak yang cukup jauh. Jarak jangkau RFID *tag* aktif ini ada yang menjanjikan dapat sampai 100 meter, namun kelemahannya adalah ukuran akan menjadi lebih besar karena adanya baterai tambahan.

2.1.1.3. RFID *tag* semi aktif

RFID *tag* semi aktif bekerja dengan menggunakan sumber tenaga bagi sistem rangkaiannya, namun sumber tenaga tidak diperlukan untuk menyuplai pengiriman sinyal balasan. Keuntungan *tag* jenis ini adalah lama masa hidup baterai yang lebih lama daripada *tag* aktif.

2.1.2 . RFID *Tag* Berdasarkan Tipe *Memory*

Berdasarkan tipe *memory* yang digunakan pada setiap RFID *tag*, pada dasarnya *memory* yang di gunakan di bedakan menjadi dua tipe sesuai dengan kemampuan kedua *memory* tersebut.

2.1.2.1. Read / Write *(Baca/Tulis)*

Memori baca/tulis secara tidak langsung sama seperti namanya, memorinya dapat dibaca dan ditulis secara berulang-ulang. Data yang dimilikinya bersifat dinamis.

2. 1.2.2 Read only (Hanya baca)

Tipe ini memiliki memori yang hanya diprogram pada saat *tag* ini dibuat dan setelah itu datanya tidak bisa diubah sama sekali. Data bersifat statis.

2.1.3.Frekuensi Kerja RFID

RFID *tag* yang biasa digunakan bila di klasifikasikan berdasarkan frekuensi radio terdapat beberapa klasifikasi sbb:

1. *Low frequency tag* (antara 125 kHz sampai 134 kHz)
2. *High frequency tag* (13.56 MHz)
3. *UHF tag* (868 MHz sampai 956 MHz)
4. *Microwave tag* (2.45 GHz)

Sampai sat ini belum ada peraturan *global* mengenai *microwave tag* sehingga *microwave tag* tidak dapat digunakan secara *global*. Jarak antara antena pembaca RFID dengan *tag* secara langsung dipengaruhi oleh frekuensi kerja yang digunakannya. Frekuensi RFID yang berbeda akan menghasilkan jangkauan komunikasi yang berbeda pula. Frekuensi RFID yang digunakan pada tugas akhir ini adalah 13.56 MHz dengan

menggunakan RFID *reader/writer* ACR 120 dengan *tag* Mifare 1 kbyte memiliki jarak operasi 10 cm.

2.1.4. Spesifikasi RFID *Tag*

Spesifikasi RFID *tag* yang akan digunakan adalah Mifare *RF Interface* (ISO/IEC 14443 A) dengan kemampuan sebagai berikut:

1. Pertukaran data secara *contactless* dan tidak dibutuhkan baterai untuk pertukaran data dan *supply energy*.
2. Jarak operasi hingga 10 cm
3. Frekuensi operasi 13,56 MHz
4. Kecepatan transfer data 106 Kbps
5. EEPROM 1 Kbytes, 16 sektor dengan 4 blok tiap sektor dengan masing-masing 16 byte (satu blok terdiri dari 16 bytes)
6. Lama penyimpanan 10 tahun
7. Kemampuan tulis 100.000 kali
8. *Transport key* melindungi akses ke EEPROM
9. *Mutual three pass authentication* (ISO/IEC DIS 9798-2)
10. Enkripsi data pada kanal RF
11. *Serial Number* yang unik pada setiap *device*

2.2 Sistem Komunikasi Nirkabel (*Wireless Communication*)

Sistem komunikasi *wireless* (nirkabel/tanpa kabel) merupakan sistem penyampaian informasi (berupa data, suara, gambar, video) tanpa media

kabel sebagai perantaranya, tetapi menggunakan media yang lain berupa udara yang dibawa lewat gelombang. Sistem Komunikasi menggunakan frekuensi/spektrum radio, yang memungkinkan terjadinya transmisi (pengiriman/penerimaan) informasi tanpa koneksi fisik. Sistem komunikasi ini bisa dilakukan dimana saja, tidak terlalu terpaku pada tempat, sebab tidak terikat dengan koneksi fisik.

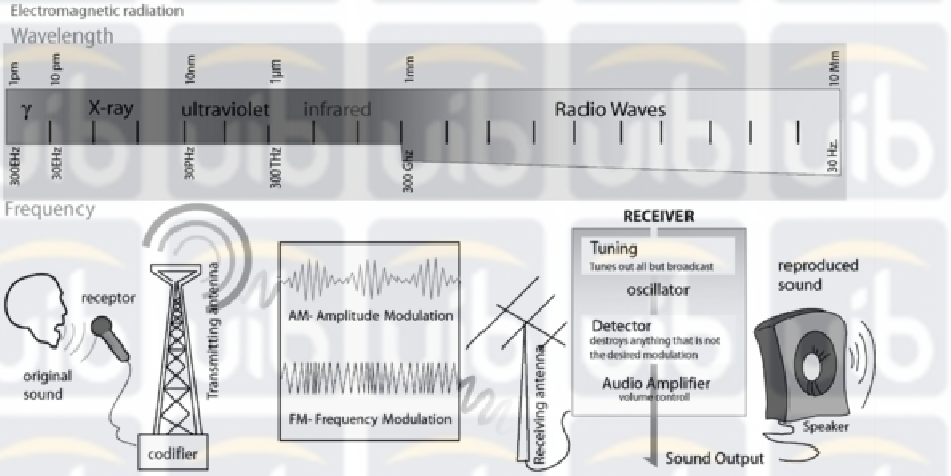
2.2.1 Gelombang Radio

Radio adalah sinyal transmisi *wireless* dengan modulasi dari gelombang elektromanetik dengan frekuensi cahaya tampak. Gelombang radio pada frekuensi 3-50 MHz dapat dipantulkan oleh lapisan *ionosfer*, suatu lapisan yang terbentuk dari ion dan elektron pada ketinggian sekitar 60 km sampai dengan 600 km di atas permukaan bumi.

Dengan pemantulan oleh lapisan *ionosfer* ini, maka komunikasi radio pada band ini bisa mencapai jarak lebih dari 2.000 km tanpa perangkat pemancar ulang (*repeater*). Ini berbeda dengan komunikasi pada band VHF-tinggi (50-300 MHz) dan UHF (300-3.000 MHz). Untuk mencapai jarak yang jauh, maka komunikasi pada *band* ini memerlukan perangkat *repeater*, dan untuk band orde *gigahertz* (lebih dari 1.000 MHz) dapat memanfaatkan satelit sebagai penguat dan pemantul sinyal yang dikirimkan.

Dikarenakan komunikasi radio pada band HF dan VHF-rendah memanfaatkan lapisan *ionosfer* sebagai pemantul, propagasi gelombangnya akan sangat bergantung pada kondisi lapisan tersebut. Pada saat kondisi

ionosfer baik dan frekuensi kerja yang digunakan sesuai dengan kondisi lapisan tersebut, maka peluang keberhasilan komunikasi juga besar sehingga komunikasi radio menjadi lebih optimal. Penjalaran radiasi elektromagnetik artinya adalah osilasi medan elektromagnetik yang melewati udara atau ruang vakum. Gelombang ini tak memerlukan medium dalam perambatannya. Informasi dibawa secara sistematis dengan cara mengubah amplitudo dan frekuensinya. Ketika gelombang radio melewati konduktor listrik, medan osilasi akan menginduksi sehingga timbul arus AC di konduktor yang lain.



Gambar 2.3
Sistem Komunikasi Nirkabel

2.2.2 Komunikasi Analog Radio Frekuensi

Sinyal analog adalah sinyal yang berupa gelombang elektromagnetik dan bergerak atas dasar frekuensi. Frekuensi adalah jumlah getaran bolak-

balik sinyal analog dalam satu siklus lengkap per detik. Satu siklus lengkap terjadi pada saat gelombang berada pada titik bertegangan nol, menuju titik bertegangan positif tertinggi pada gelombang, menurun ke titik tegangan negatif dan menuju ke titik nol kembali.semakin tinggi kecepatan, semakin banyak siklus lengkap yang terjadi dalam suatu periode tertentu. Kecepatan frekuensi tersebut dinyatakan dalam Hertz (Hz). Sebagai contoh sebuah gelombang yang berayun bolak-balik sebanyak sepuluh kali setiap detik berarti memiliki kecepatan 10 Hz.

Sinyal analog dapat digunakan dalam media tertutup seperti kabel *coaxial*, TV kabel dan kabel tembaga. Sinyal analog dapat pula digunakan melalui media terbuka seperti gelombang mikro, telepon rumah tanpa kabel dan telepon seluler. Pengiriman sinyal analog dapat dianalogikan seperti mengirim air melalui pipa. Aliran pipa kehilangan tenaganya saat disalurkan melalui sebuah pipa. Semakin jauh pipa, semakin banyak tenaga yang terkurang dan aliran menjadi semakin lemah. Demikian pula sinyal analog juga memungut interferensi elektrik atau noise dari dalam jalur. Kabel listrik, petir dan mesin listrik semua menginjeksikan noise dalam bentuk elektrik pada sinyal analog. Untuk mengatasi kelemahan tersebut maka diperlukan alat penguat sinyal yang disebut *amplifier*.

2.2.3 Komunikasi Digital Radio Frekuensi

Sebagai pengganti gelombang maka sinyal pada sistem digital ditransmisikan dalam bentuk bit bit biner. Sistem biner adalah sistem on – off

(atau sistem 1-0), jadi apabila ada tegangan atau on maka akan diangkakan 1, sedangkan bila tidak ada tegangan atau off maka akan diangkakan 0. Meski memiliki kelemahan terhadap *noise* interferensi listrik apabila jarak semakin jauh, namun sinyal digital masih dapat diperbaiki atau "direparasi" artinya dengan cara membangkitkan ulang bit bit tersebut dengan tidak meregenerasikan noise.

Kelebihan pada sinyal sistem digital dibandingkan sinyal analog adalah :

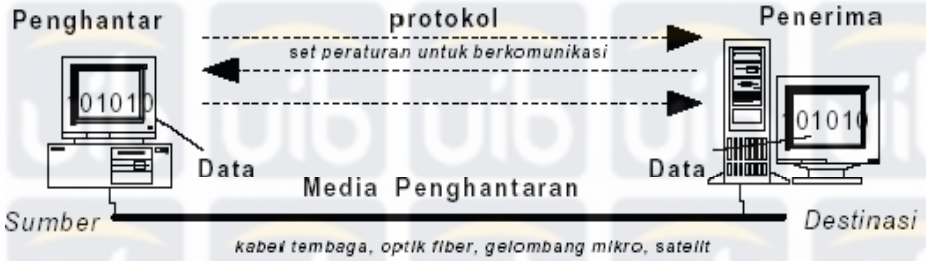
a. Kualitas suara lebih jernih, selain lebih jelas sinyal digital memiliki sedikit kesalahan.
b. Kecepatan dari sinyal digital lebih tinggi dibandingkan sinyal analog.
c. Sinyal digital memiliki lebih sedikit kesalahan daripada sinyal analog.
d. Untuk sinyal digital memerlukan tenaga pendukung yang tidak terlalu kompleks.

2.3 Sistem Komunikasi Data

Komunikasi data adalah merupakan bagian dari telekomunikasi yang secara khusus berkenaan dengan transmisi atau pemindahan data dan informasi diantara komputer dan piranti-piranti yang lain dalam bentuk digital yang dikirimkan melalui media komunikasi data. Data berarti informasi yang disajikan oleh isyarat digital. Komunikasi data merupakan bagian vital dari suatu masyarakat informasi karena sistem ini menyediakan

infrastruktur yang memungkinkan komputer-komputer dapat berkomunikasi satu sama lain.Komponen komponen tersebut adalah :

- **Pengirim**, adalah piranti yang mengirimkan data
- **Penerima**, adalah piranti yang menerima data
- **Data**, adalah informasi yang akan dipindahkan
- **Media pengiriman**, adalah media atau saluran yang digunakan untuk mengirimkan data
- **Protokol**, adalah aturan-aturan yang berfungsi untuk menyelaraskan hubungan.



Gambar 2.4
Sistem Komunikasi Data

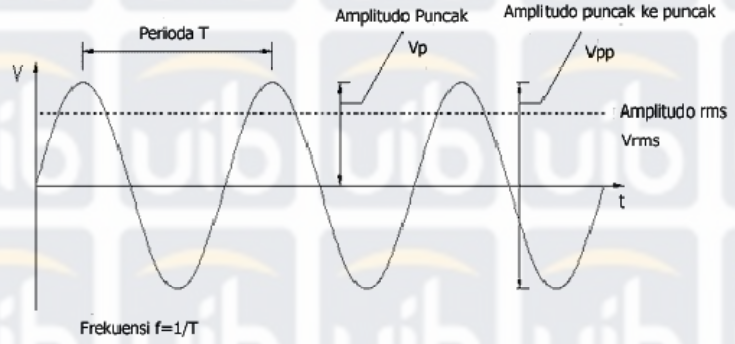
2.3.1 Perbedaan Sinyal/Isyarat Analog Dengan Digital

2.3.1.1 Sinyal Analog

Sinyal analog adalah sinyal data dalam bentuk gelombang yang yang kontinyu, yang membawa informasi dengan mengubah karakteristik gelombang. Dua parameter/karakteristik terpenting yang dimiliki oleh isyarat analog adalah amplitude dan frekuensi. Isyarat analog biasanya dinyatakan dengan gelombang sinus, mengingat gelombang sinus merupakan dasar

untuk semua bentuk isyarat analog. Hal ini didasarkan kenyataan bahwa berdasarkan analisis *fourier*, suatu sinyal analog dapat diperoleh dari perpaduan sejumlah gelombang sinus. Dengan menggunakan sinyal analog, maka jangkauan transmisi data dapat mencapai jarak yang jauh, tetapi sinyal ini mudah terpengaruh oleh *noise*. Gelombang pada sinyal analog yang umumnya berbentuk gelombang sinus memiliki tiga variable dasar, yaitu amplitudo, frekuensi dan *phase*.

- ***Amplitudo*** merupakan ukuran tinggi rendahnya tegangan dari sinyal analog.
- *Frekuensi* adalah jumlah gelombang sinyal analog yang terjadi dalam satu detik.
- *Phase* adalah besar sudut dari sinyal analog pada saat tertentu.

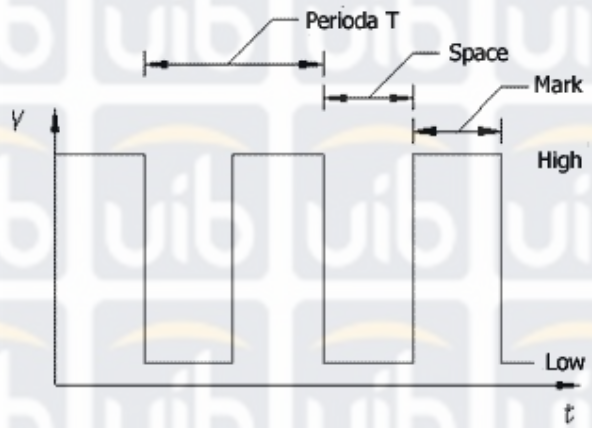


Gambar 2.5
Bentuk Sinyal Analog

2.3.1.2 Sinyal Digital

Sinyal digital merupakan sinyal data dalam bentuk deposit yang dapat mengalami perubahan yang tiba-tiba dan mempunyai besaran 0 dan 1. Sinyal

digital hanya memiliki dua keadaan, yaitu 0 dan 1, sehingga tidak mudah terpengaruh oleh derau, tetapi transmisi dengan sinyal digital hanya mencapai jarak jangkau pengiriman data yang relatif dekat. Biasanya sinyal ini juga dikenal dengan sinyal diskret. Sinyal yang mempunyai dua keadaan ini biasa disebut dengan bit. Bit merupakan istilah khas pada sinyal digital. Sebuah bit dapat berupa nol (0) atau satu (1). Kemungkinan nilai untuk sebuah bit adalah 2 buah . Kemungkinan nilai untuk 2 bit adalah sebanyak 4 , berupa 00,01,10, dan 11. Secara umum, jumlah kemungkinan nilai yang terbentuk olehkombinasi n bit adalah sebesar 2n buah.



Gambar 2.6
Bentuk Sinyal Digital

2.3.2 Protokol

Protokol adalah sebuah aturan yang mendefinisikan beberapa fungsi yang ada dalam sebuah jaringan komputer, misalnya mengirim pesan, data, informasi dan fungsi lain yang harus dipenuhi oleh sisi pengirim dan sisi penerima agar komunikasi dapat berlangsung dengan benar, walaupun sistem

yang ada dalam jaringan tersebut berbeda sama sekali. Protokol ini mengurusi perbedaan format data pada kedua sistem hingga pada masalah koneksi listrik.

Standar protokol yang terkenal yaitu OSI (*Open System Interconnecting*) yang ditentukan oleh ISO (*International Standart Organization*).

2.3.2.1. Komponen Protokol

Beberapa komponen yang terdapat dalam protokol adalah sebagai berikut :

1. Aturan atau prosedur

   Aturan atau prosedur ini berfungsi untuk mengatur penyambungan atau pemutusan hubungan serta mengatur proses transfer data.

2. Format atau bentuk

   Merupakan representasi dari pesan yang dikirim.

3. Kosakata (*vocabulary*)

   Merupakan jenis pesan dan makna masing-masing pesan

2.3.2.2. Fungsi Protokol

Secara umum fungsi dari protokol adalah untuk menghubungkan sisi pengirim dan sisi penerima dalam berkomunikasi serta dalam bertukar informasi agar dapat berjalan dengan baik dan benar. Sedangkan fungsi protokol secara detail dapat dijelaskan berikut:

1. Fragmentasi dan *reassembly*

Fungsi dari fragmentasi dan *reasembly* adalah membagi informasi yang dikirim menjadi beberapa paket data pada saat sisi pengirim mengirimkan informasi dan setelah diterima maka sisi penerima akan menggabungkan lagi menjadi paket informasi yang lengkap.

2. *Encaptulation*

Fungsi dari *encaptulation* adalah melengkapi informasi yang dikirimkan dengan *address*, kode-kode koreksi dan lain-lain.

3. *Connection control*

Fungsi dari *Connection control* adalah membangun hubungan (*connection*) komunikasi dari sisi pengirim dan sisi penerima, dimana dalam membangun hubungan ini juga termasuk dalam hal pengiriman data dan mengakhiri hubungan.

4. *Flow control*

Berfungsi sebagai pengatur perjalanan data dari sisi pengirim ke sisi penerima.

5. *Error control*

Dalam pengiriman data tak lepas dari kesalahan, baik itu dalam proses pengiriman maupun pada waktu data itu diterima. Fungsi dari *error control* adalah mengontrol terjadinya kesalahan yang terjadi pada waktu data dikirimkan.

6 Transmission service

Fungsi dari transmission service adalah memberi pelayanan komunikasi data khususnya yang berkaitan dengan prioritas dan keamanan serta perlindungan data.

2.3.3 Susunan Protokol

Protokol jaringan disusun dalam bentuk lapisan-lapisan (*layer)*. Hal ini mengandung arti supaya jaringan yang dibuat nantinya tidak menjadi rumit. Di dalam *layer* ini, jumlah, nama, isi dan fungsi setiap *layer* berbeda-beda. Akan tetapi tujuan dari setiap *layer* ini adalah memberi layanan ke *layer* yang ada diatasnya. Susunan dari *layer* ini menunjukkan tahapan dalam melakukan komunikasi. Antara setiap *layer* yang berdekatan terdapat sebuah *interface*. *Interface* ini menentukan layanan *layer* yang di bawah kepada *layer* yang di atasnya. Pada saat merencanakan sebah jaringan, hendaknya memperhatikan bagaimana menentukan *interface* yang tepat yang akan ditempatkan di antara dua *layer* yang bersangkutan.

2.3.4. Standarisasi Protokol (ISO 7498)

ISO (*International Standard Organization*) mengajukan struktur dan fungsi protokol komunikasi data. Model tersebut dikenal sebagai OSI (*Open System Interconnection*) *Reference Model*. Terdiri atas 7 *layer* (lapisan) yang mendefinisikan fungsi. Untuk tiap *layer*nya dapat terdiri atas sejumlah *protokol* yang berbeda, masing-masing menyediakan pelayanan yang sesuai dengan fungsi *layer* tersebut.

- *Application Layer*: *interface* antara aplikasi yang dihadapi *user* dan *resource* jaringan yang diakses.
- *Presentation Layer*: rutin *standard* mem-presentasi-kan data.
- *Session Layer*: membagi presentasi data ke dalam babak-babak (sesi)
- *Transport Layer:* menerima data,memecahkan dan meneruskan data ke *network layer*.
- *Network Layer:* pengalamatan dan pengiriman paket data.
- *Data-link Layer:* pengiriman data melintasi jaringan fisik.
- *Physical Layer:* karakteristik perangkat keras yang mentransmisikan sinyal data.

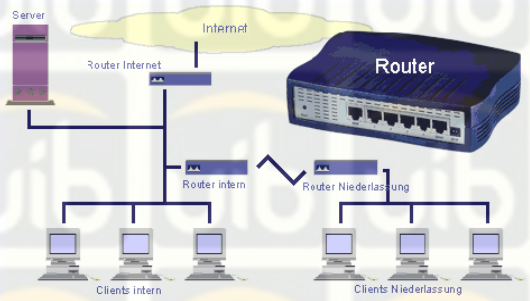
2.3.5. Router, Bridge dan Repeater

2.3.5.1 Router

Router adalah merupakan piranti yang menghubungkan dua buah jaringan yang berbeda tipe maupun protokol. Dengan *router* dapat dimungkinkan untuk :

- Menghubungkan sejumlah jaringan yang memiliki topologi dan protokol yang berbeda. Menghubungkan jaringan pada suatu lokasi dengan jaringan pada lokasi yang lain.
- Membagi suatu jaringan berukuran besar menjadi jaringan-jaringan yang lebih kecil dan mudah untuk dikelola.
- Memungkinkan jaringan dihubungkan ke internet dan informasi yang tersedia dapat diakses oleh siapa saja.

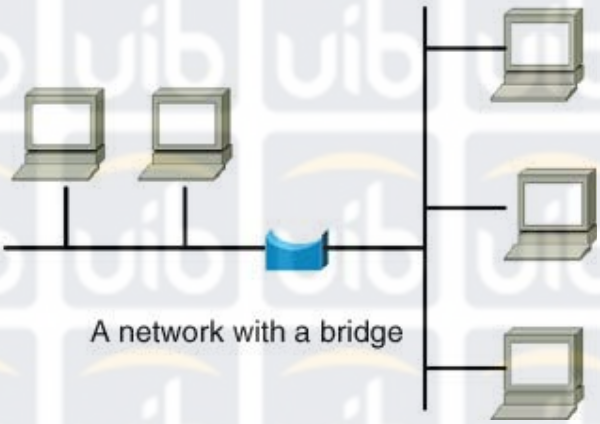
- Mencari jalan terefisien untuk mengirimkan data ke tujuan.
- Melindungi jaringan dari pemakai-pemakai yang tidak berhak dengan cara membatasi akses terhadap data-data yang tidak berhak untuk diakses.



Gambar 2.7
Jaringan Komputer

2.3.5.2. Bridge

Bridge adalah jenis perangkat yang diperlukan jika dua buah jaringan bertipe sama (ataupun bertopologi berbeda) tetapi dikehendaki agar lalu lintas lokal masing-masing jaringan tidak saling mempengaruhi jaringan yang lainnya. Bridge memiliki sifat yang tidak mengubah isi maupun bentuk *frame* yang diterimanya, disamping itu *bridge* memiliki *buffer* yang cukup untuk menghadapi ketidaksesuaian kecepatan pengiriman dan penerimaan data.



Gambar 2.8
Jaringan Komputer Dengan Menggunakan *Bridge*

Adapun alasan menggunakan *bridge* adalah sebagai berikut :

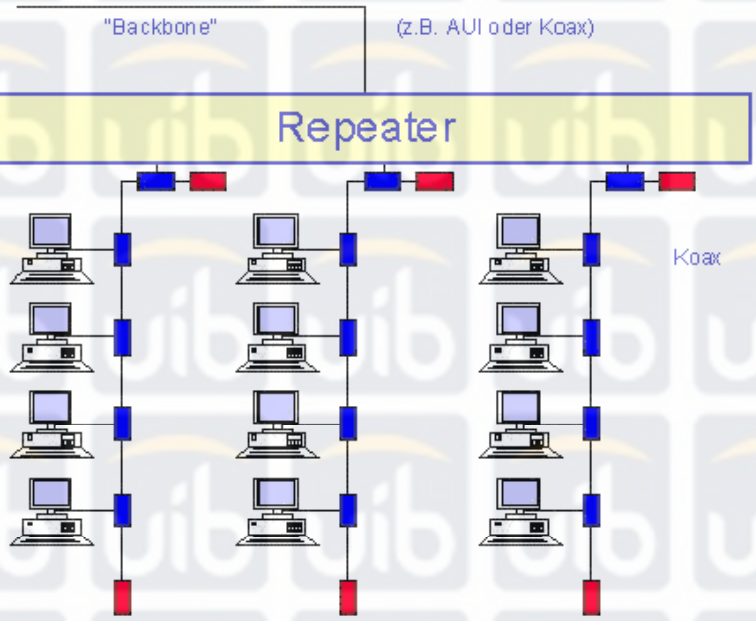
- Keterbatasan jaringan, hal ini terkait erat dengan jumlah maksimum stasiun, panjang maksimum segmen, dan bentang jaringan.
- Kehandalan dan keamanan lalu lintas data, *bridge* dapat menyaring lalu lintas data antar dua segmen jaringan.
- Semakin besar jaringan, performa atau unjuk kerja semakin menurun
- Bila dua sistem pada tempat yang berjauhan disambungkan, penggunaan *bridge* dengan saluran komunikasi jarak jauh, akan lebih menguntungkan dibandingkan dengan menghubungkan langsung dua sistem tersebut

2.3.5.3 Repeater

Repeater adalah piranti yang berfungsi untuk memperbaiki dan memperkuat sinyal atau isyarat yang melewatinya, Dua sub jaringan yang dilewatkan pada *repeater* memiliki protokol yang sama untuk semua lapisan.

Repeater juga berfungsi untuk memperbesar batasan panjang satu segmen. Sehingga dapat digunakan untuk memperpanjang jangkauan jaringan. Repeater bekerja sebagai sebuah stasiun untuk menerima sinyal yang masuk dan mengirimnya kembali pada frekuensi yang berbeda.

Tujuan utama *repeater* adalah memperluas jangkauan operasi dari stasiun bergerak, atau stasiun di daerah rendah atau di daerah terpencil di mana komunikasi simplex biasanya tidak mungkin. Juga dapat digunakan sebagai *channel* kontak panggilan sebelum beralih ke frekuensi simpleks.



Gambar 2.9
Jaringan Komputer Dengan Menggunakan Repeater

2.4 Pemograman RFID Dengan Menggunakan Visual Basic 6.0

Visual Basic adalah salah suatu bahasa pemograman untuk membangun aplikasi dalam lingkungan *Windows*. Dalam pengembangan aplikasi, *Visual Basic* menggunakan pendekatan *visual* untuk merancang *user interface* dalam bentuk *form*, sedangkan untuk kodingnya menggunakan bahasa *Basic* yang cenderung mudah dipelajari. *Visual Basic* telah menjadi *tools* yang terkenal bagi para pemula maupun para *developer*. Dalam lingkungan *Windows*, *user interface* sangat memegang peranan penting, karena dalam pemakaian aplikasi yang di buat, pemakai senantiasa berinteraksi dengan *user-interface* tanpa menyadari bahwa dibelakangnya berjalan instruksi-instruksi program yang mendukung tampilan dan proses yang dilakukan.

Pada pemrograman *visual*, pengembangan aplikasi dimulai dengan pembentukan *user interface*, kemudian mengatur properti dari obyek-obyek yang digunakan dalam *user interface*, dan baru dilakukan penulisan kode program untuk menangani *event* (kejadian-kejadian). Tahap pengembangan aplikasi demikian dikenal dengan istilah pengembangan aplikasi dengan pendekatan *Bottom Up*. Perkembangan PC diiringi juga dengan perkembangan sistem operasi yang salah satunya adalah DOS (*Disk Operating System*). DOS merupakan sistem operasi non GUI (*Graphicals User Interface*). DOS pada saat itu banyak digunakan pada komputer IBM. Sejak keberadaaan SO (Sistem Operasi) non GUI, muncul bahasa pemrograman linier / bahasa pemrograman struktural yang peranannya cukup

dominan dalam pengembangan aplikasi berbasis *visual*. Bahasa-bahasa pemrograman tersebut sering dikenal sebagai bahasa pemrograman konvensional (bahasa pemrograman *under* DOS) yang diantaranya adalah C/C++, *Pascal, Basic, xBase, Prolog, Java* dan lainya. Setelah munculnya SO (*Sistem Operasi*) berbasis GUI, maka pemrograman yang ditujukan untuk SO (*Sistem Operasi*) berbasis GUI banyak ditekankan pada tampilan/desainnya. Oleh karena itulah bahasa pemrograman dengan menggunakan konsep RAD (*Rapid Application Development*) muncul.

Orientasi pemrograman yang timbul saat ini adalah bagaimana menggunakan *tools* untuk membuat antar muka dalam program aplikasi dengan sedikit kode program, bukan bagaimana membuat komponen antar muka dengan kode program yang kompleks dan waktu yang lama dalam pembuatannya.

Dengan pemrograman menggunakan konsep RAD, *user* hanya meng-*drag-drop* objek untuk membuat *interface* (antar muka) dan kemudian dilanjutkan dengan penulisan kode program untuk mengendalikan objek dan memberi perintah – perintah tertentu, dan akibatnya lebih cepat pembuatannya dibanding dengan bahasa pemrograman konvensional.

Beberapa kompilator bahasa pemrograman berbasis GUI dan berkonsep RAD adalah *Visual Basic, Visual VoxPro, Visual dBase,* C++ *Builder, Borlan Delphi, Jbuilder, Visual Prolog* dan masih banyak yang lain. Turbo Paskal, versi RAD-nya yaitu *Borland DELPHI, Quick BASIC* oleh *Microsoft* dikembangkan menjadi *VISUAL BASIC*. Bahasa C++ versi RAD-nya yaitu

C++ *BUILDER* dan Microsoft *Visual* C++, Microsoft *FoxPro* versi RAD-nya yaitu *Microsoft Visual FoxPro,* dBase versi RAD-nya yaitu *Microsoft Visual dBase, Turbo Prolog* versi RAD-nya yaitu *Visual Prolog, Java* versi RAD-nya yaitu *JBuilder*.

Contoh aplikasi pemograman *Visual Basic* 6.0 yang diterapkan penulis berupa pemograman RFID dimana program mampu membaca dan menulis kartu *Mifare* 1K melalui modem RFID. Beberapa tahapan atau prosedur dimana RFID dapat dikenali, dibaca dan menulis ke kartu *Mifar*e 1K dimana kartu Mifare 1K ini terdiri atas memori yang dapat menyimpan data dalam bentuk *bytes*. Berikut contoh pemograman dengan menggunakan *Visual Basic* 6.0 :

1. Mendeteksi dan *Connect* ke RFID *Reader*

Program *Visual Basic* di bawah, adalah langkah pertama dari *programmer* untuk mengenali RFID *Reader* berupa inisialisasi mendeteksi *modem* RFID Reader dengan sintak *"SCardEstablishContext(dwScope, 0, 0, hContext)"*.

Contoh program inisialisasi RFID *Reader* :

```
Dim rc As Long
Dim dwScope As Long
Dim mszReaders(2048) As Byte
Dim mszGroup(1024) As Byte
Dim pcchReaders As Long

hCard = 0
hContext = 0
fLookForCard = True
' Set Scope (see Scard.bas "Scopes")
dwScope = SCARD_SCOPE_USER

' Establish Context
rc = SCardEstablishContext(dwScope, 0, 0, hContext)

' Set maximum Length of mszReaders
```

```
pcchReaders = 2048

' Create a Multistring (terminated with two '\0's)
mszGroup(0) = &H0
mszGroup(1) = &H0
rc = SCardListReaders(hContext, mszGroup(LBound(mszGroup)),
mszReaders(LBound(mszReaders)), pcchReaders)
List1.Clear
' Split multistring and add single Readers to list
For i = 0 To pcchReaders - 2 Step 1
      curReader = curReader + GetString(mszReaders(i))

     If mszReaders(i) = &H0 And i <> 0 Then
       List1.AddItem (curReader)
       curReader = ""
   End If
Next i
For i = 0 To List1.ListCount - 1 Step 1
List1.ListIndex = i
If List1.Text = "OMNIKEY CardMan 5121-CL 0" Then
SelectedReader = List1.Text
End If
Next i
End Sub
```

2. RFID *Reader* membaca kartu *Mifare* 1K

Setelah komputer menginisialisasi dan mendeteksi *modem* RFID *Reader*, untuk proses membaca kartu *Mifare* 1K melalui RFID *Reader* dengan menggunakan sintak *"SCardCLMifareStdRead(hCard, BlockNr, ucData(LBound(ucData)), 16, NumofDataLen)"*.

Contoh program proses *read* kartu *Mifare* 1K melalui RFID *Reader* :

```
Dim BlockNr As Long
Dim ucData(16) As Byte
Dim NumofDataLen As Long
Dim StringDump As String
Dim DisplayText As String

BlockNr = 1
NumofDataLen = 16
rc = SCardCLMifareStdRead(hCard, BlockNr, ucData(LBound(ucData)),
16, NumofDataLen)
   If rc = 0 Then
      DisplayText = StrDump(ucData, NumofDataLen)
       List2.AddItem DisplayText
   End If
```

3. RFID *Reader* menulis ke kartu *Mifare* 1K

Setelah komputer menginisialisasi dan mendeteksi *modem* RFID *Reader*, proses menulis kartu *Mifare* 1K melalui RFID *Reader* dengan menggunakan sintak *"SCardCLMifareStdWrite(hCard, BlockNr, ucData(LBound(ucData)), 16)"*.

Contoh program proses *write* kartu *Mifare* 1K melalui RFID *Reader* :

```
Dim BlockNr As Long
Dim ucData() As Byte
Dim DataLen As Long

Call Connect
BlockNr = 1
If fCardConnected = False Then
List2.AddItem " At first Card has to be connected"
GoTo EXITSUB
End If

If BlockNr < 128 Then
If ((BlockNr + 1) Mod 4) = 0 Then
List2.AddItem  "  Sector  block  cannot  be  written  with  this
application"
GoTo EXITSUB
End If
End If
If BlockNr > 128 Then
If ((BlockNr + 1) Mod 16) = 0 Then
List2.AddItem  "  Sector  block  cannot  be  written  with  this
applpication "
GoTo EXITSUB
End If
End If

ucData = StrConv(WriteText, vbFromUnicode)
rc         =         SCardCLMifareStdWrite(hCard,         BlockNr,
ucData(LBound(ucData)), 16)
If rc <> 0 Then
List2.AddItem " Error in writting Data "
List2.AddItem HandleError(rc)
End If
EXITSUB:
List2.ListIndex = List2.ListCount - 1
```

4. Diskoneksi RFID *Reader*

Untuk mendiskoneksi RFID *Reader* ke komputer, *programmer* menggunakan sintak *"(hCard, dwDisposition)"* kemudian dilanjutkan

dengan *release contex* dengan menggunakan sintak "SCardReleaseContext(hContext)".

Contoh program proses *disconnect* kartu Mifare 1K melalui RFID Reader :

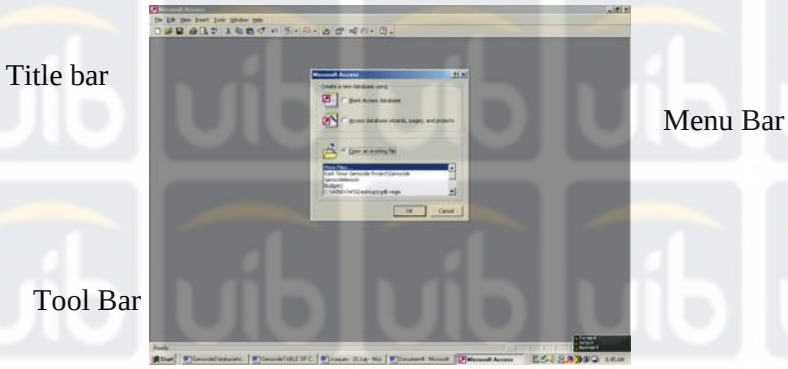
```
Dim rc As Long
Dim dwDisposition As Long

' check if a card is connected
If fCardConnected = True Then
    ' Set Disposition (see Scard.bas "Dispositions")
    dwDisposition = SCARD_LEAVE_CARD

    'Disconnect from Card
    rc = SCardDisconnect(hCard, dwDisposition)
    If rc <> OKERR_OK Then
        List2.AddItem HandleError(rc)
        Exit Sub
    End If

    List2.AddItem "Disconnect successfull"
    fCardConnected = False
    ' release
    rc = SCardReleaseContext(hContext)
    If rc <> OKERR_OK Then
        List2.AddItem HandleError(rc)
        Exit Sub
    End If
    ' remove readers from List1
    List2.AddItem "Release successfull"
    isContext = False
End If
List2.AddItem "Disconnect successfull"
```

2.5. Pemograman Microsoft Access Database Dan ADO (ActiveX Data Object) Dengan Menggunakan Visual Basic 6.0

Sebuah *database* adalah sekumpulan informasi yang berguna yang disusun secara khusus. Misalnya, buku telepon adalah suatu kumpulan nama-nama, alamat-alamat, serta nomor-nomor telepon para pelanggan. Kita menciptakan *database* agar supaya informasi itu dapat disimpan secara efisien dan digunakan bilamana perlu. *Database* dibagi atas kategori (*fields*) dan rekaman (*records*). Sebuah kategori pada dasarnya adalah sebuah kolom

informasi, sedangkan rekaman adalah informasi yang diisikan pada kolom tersebut.

Microsoft Access adalah sebuah sistem pengelolaan *database* secara elektronis yang memungkinkan disusunnya informasi yang banyak secara sistematis dan direkam ke dalam sebuah komputer. Ketika membuka Program *Access* dengan cara mengklik dua kali di *icon* Access atau tekan Start atau Programs Menu. Tool Bars sangat berguna untuk melaksanakan *shortcuts* untuk perintah-perintah yang diberikan. Toolbars bisa juga diakses dengan menggunakan sebuah *mouse*. Apabila *user* meletakkan penunjuk *mouse* di toolbar, maka Tooltips akan memberikan sebuah deskripsi tentang pilihan Toolbar.



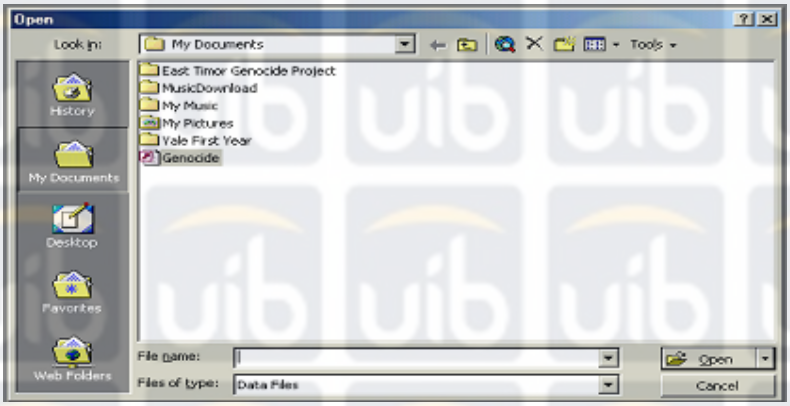
Gambar 2.10
Tampilan Access

2.5.1. Membuat Database

Untuk membuat sebuah *database*, harus terlebih dahulu memberikan sebuah nama, misalnya TOOLSdatabase. *Database* tersebut memiliki semua unsur yang terdapat dalam sistem. Langkah untuk membuat *database* melalui aplikasi Access

Klik **Blank Access *Database***

Klik **OK**



Gambar 2.11
Kotak Dialog Baru Dari Database

Ketiklah **GENOCIDE**

Tekan **<Enter>**

Database GENOCIDE.mdb sudah terciptakan dan dapat melihatnya pada layar:

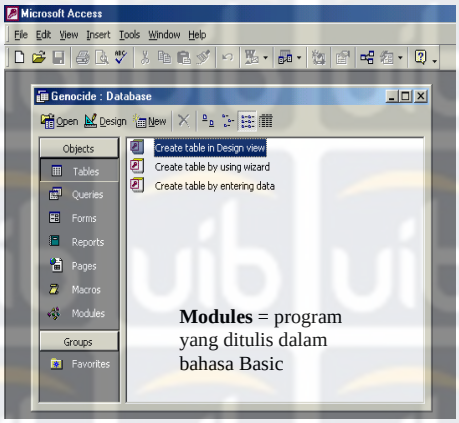
Tabel = kerangka dasar dari sebuah *database*. Tabel = jaringan dimana kita masukkan data

Queries mencari dan memanggil data

Forms memungkinkan data entry dalam table yang mudah dibaca pada layar.

Reports menciptakan output data dari tabel.

Macros = program bahasa sederhana yang me



Gambar 2.12
Daftar Objek Database

Metoda yang digunakan untuk mengakses *database under Windows*, beberapa metoda dapat dilakukan seperti ADO (*ActiveX Data Object*), DAO (*Data Access Object*) dan RDO (*Remote Data Object*). *ActiveX Data Object* adalah model akses data tingkat tinggi dan metode yang paling disenangi di *Visual Basic*. Alasannya karena ADO lebih sederhana dan lebih fleksibel daripada DAO dan RDO. Bahkan ada yang menyebutnya dengan *Flatter Object Model* atau *Simpler Access*. Sintak ADO sama dengan DAO atau RDO. ADO seringkali digunakan bersama dengan OLE DB, yakni *low level interfaces* (antar muka tingkat rendah) yang dapat digunakan untuk mengakses data apapun format datanya, baik *database* SQL (*Structured Query Language*), sistem *e-mail*, *flat-file* dan lain-lain.

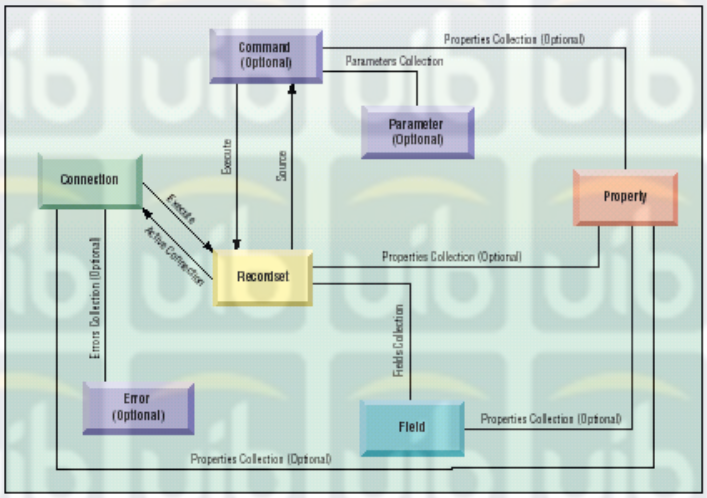
Model objek pada ADO versi 1.5, hanya terdiri dari 7 objek dan 4 koleksi, sedangkan pada RDO 2.0 terdiri dari 10 objek dan 9 koleksi objek, apalagi pada DAO 3.5 terdiri dari 10 objek dan 16 koleksi objek. Inilah yang menjadi alasan mengapa ADO lebih sederhana. Perbandingan model objek ADO, DAO dan RDO terdapat dalam tabel berikut ini.

ADO	DAO with ODBCDirect	RDO
N/A	DBEngine	rdoEngine
N/A	Workspace	rdoEnvironment
Connection	Connection	rdoConnection
Open	Workspace.OpenConnection	EstablishConnection
Close	Close	Close
OpenSchema	N/A	N/A
Command	QueryDef	rdoQuery
Created with New	Connection.CreateQueryDef	rdoConnection.CreateQuery
Execute	Execute	Execute
CommandText	SQL	SQL
Recordset	Recordset	rdoResultset
Open	Connection.OpenRecordset	rdoConnection.OpenResultset
ActiveConnection	Connection	ActiveConnection
UpdateBatch	Update(adUpdateBatch)	BatchUpdate
NextRecordset	NextRecordset	MoreResults
GetRows	GetRows	GetRows
N/A	StillExecuting	StillExecuting
Parameter (Created with New and appended to the Command)	Parameter (Automatically added to the collection for the QueryDef)	rdoParameter (Automatically added to the collection for the rdoQuery)
Direction	Direction	Direction
Type	Type	Type
Value	Value	Value
Field	Field	rdoColumn
Value	Value	Value
UnderlyingValue	VisibleValue	BatchConflictValue
OrginalValue	OriginalValue	OrginalValue
Error	Error	rdoError
Description	Description	Description
Number	Number	Number
Source	Source	Source
Property	N/A	N/A

Tabel 2.1
Perbandingan ADO,DAO dan RDO

Tujuh Objek yang terdapat pada ADO adalah

1. *Connection*
2. *Command*
3. *Recordset*
4. *Parameter*
5. *Field*
6. *Error*
7. *Property*



Gambar 2.13
Model Objek ADO

Model objek ADO tidak berhirarki seperti halnya model objek DAO atau RDO. Artinya dalam ADO, *user* dapat membuat objek *Connection*, *Command*, Parameter dan *Recordset* secara terpisah (*independent*) menggunakan pernyataan ***New*** atau ***Create***. Setelah itu objek-objek yang terpisah tersebut secara bebas dapat saling dihubungkan satu sama lain pada saat *run time*.

Contoh kode program Visual Basic :

```
If Dir$(App.Path + "\" + "CustomerData.mdb") <> "" Then
   GoTo GoHere
   Else
       Set conCatalog = New ADOX.Catalog
       conCatalog.Create "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                         "Data     Source="     +     App.Path     +
   "\CustomerData.mdb"
       Set conCollection = New ADODB.Connection
       conCollection.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                          "Data     Source="     +     App.Path     +
   "\CustomerData.mdb"
strSQL1 = "CREATE TABLE CustTable (" & _
          "CustID String," & _
          "EntryDate String," & _
          "EntryTime String," & _
          "OutDate String," & _
          "OutTime String," & _
```

```
            "StartDeposit Integer," & _
            "Duration Integer," & _
            "BalanceDeposit Integer);"
conCollection.Execute strSQL1
End If
```

	DAO	ADO
Category 1: DAO features for which a similar approach exists in ADO	Database connections Fields Objects Error Objects Transactions Properties Recordsets Connections	Database connections Fields Objects Error Objects Transactions Properties Recordsets Connections
Category 2: DAO features not in ADO	Workspace Object Database Object TableDefs Object QueryDefs Object Index Object Jet-Specific Features: Groups/Users Object Relations Object Containers Object Document Object Properties Object	
Category 3: ADO features not in DAO		Events Command Object Disconnected Recordsets

Tabel 2.2
Perbandingan ADO dan DAO

Hal lain yang menyebabkan ADO lebih baik daripada DAO adalah ADO memberikan susunan komponen objek dan cara mengakses yang lebih sederhana (*Simpler Access*). Sebagai contoh ADO menggabungkan objek *DBEngine*, *Workspaces* dan *Database* menjadi satu buah objek yang disebut dengan objek *Connection*. Demikian juga dengan metode *Find*. Pada DAO terdapat lima metode *find* (*FindFirst*, *FindNext*, *FindPrevious*, *FindLast* dan *Seek*). Sedangkan pada ADO, kelima metode tersebut menjadi satu buah metode yaitu metode *Find*, dan untuk objek *TableDefs* dan *QueryDefs* digabung menjadi objek *Command*.

2.5.2. Koneksi Database

Membuat koneksi dalam ADO berbeda dengan DAO. Pada DAO untuk membuat koneksi pertama kali harus membuat objek *workspace* untuk ODBC *direct* dan menggunakan metode *OpenConnection* untuk membuat dan membuka koneksi. Sedangkan pada RDO maupun ADO, pertama kali adalah membuat objek koneksi, setelah itu melakukan *setting* pada beberapa *property* untuk melakukan koneksi *string* dan menggunakan metode untuk membuka koneksi. Contoh kode program Visual Basic :

```
If Dir$(App.Path + "\" + "CustomerData.mdb") <> "" Then
    conDev.Open "Provider='Microsoft.JET.OLEDB.4.0';" & _
                "Data Source=" + App.Path + "\CustomerData.mdb"
    rcsDev.Open      "CustTable",      conDev,      adOpenStatic,
adLockOptimistic
    conDevFN.Open "Provider='Microsoft.JET.OLEDB.4.0';" & _
                "Data Source=" + App.Path + "\CustomerData.mdb"
    rcsDevFN.Open     "CustTable",      conDev,      adOpenStatic,
adLockOptimistic
End If
```

Setelah terhubung dengan *database*, maka *resulset* dapat digunakan. Cara yang paling sederhana untuk membuka *resultset* adalah dari pernyataan SQL yang dibangkitkan.

2.5.3 Membuat Tabel

Gambar 2.14
Tabel Database

Contoh kode program Visual Basic untuk membuat tabel :

```
If Dir$(App.Path + "\" + "CustomerData.mdb") <> "" Then
GoTo GoHere
Else
    Set conCatalog = New ADOX.Catalog
    conCatalog.Create "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                      "Data      Source="      +      App.Path      +
 "\CustomerData.mdb"
    Set conCollection = New ADODB.Connection
    conCollection.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                      "Data      Source="      +      App.Path      +
 "\CustomerData.mdb"
    strSQL1 = "CREATE TABLE CustTable (" & _
              "CustID String," & _
              "EntryDate String," & _
              "EntryTime String," & _
              "OutDate String," & _
              "OutTime String," & _
              "StartDeposit Integer," & _
              "Duration Integer," & _
              "BalanceDeposit Integer);"
    conCollection.Execute strSQL1
End If
```

2.5.4. Adding Data Pada Tabel

Secara prinsip, mengisi data pada tabel *database* sama dengan menambah data pada *database* dan penambahan data ini tidak meng-*override* data yang lama, tetapi mengisi sel tabel kosong yang terakhir. Kode program Visual Basic untuk menambah data pada tabel *database* :

```
If rcsDev.Supports(adAddNew) Then
      With rcsDevFN
      .AddNew
      .Fields("CustID") = Left$(DisplayText, 10)
      .Fields("EntryDate") = Format(Date, "mm/dd/yy")
      .Fields("EntryTime") = Format(Time, "hh:mm:ss")
      .Fields("StartDeposit") = StartDepos
      .Update
      End With
End If
```

2.5.5. Melihat Data

Untuk me-*recall* atau melihat data pada tabel *database*, kondisi *database* harus memiliki koneksi ke program aplikasi dimana walaupun *database* kondisinya masih dalam keadaan terbuka, tetap data sebelumnya

dapat di-*recall*. Kode program Visual Basic untuk me-*recall* data pada tabel *database* :

```
conCollection.Open "Provider='Microsoft.JET.OLEDB.4.0';" & _
                   "Data Source=" + App.Path + "\CustomerData.mdb"
rcsCollection.Open  "CustTableHystory",  conCollection,  adOpenStatic,
adLockOptimistic

Adodc3.RecordSource = "SELECT * FROM [CustTableHystory] WHERE CustID
= '" & SeachText.Text & "'"
Adodc3.Refresh

If Adodc3.Recordset.RecordCount <> 0 Then
   Adodc3.Recordset.MoveFirst
End If
UsersHystory.Show
```

2.5.6. Membuka Recordset/Resulset

Yang menarik dari teknik ADO adalah dalam membuat recordset. Pada ADO untuk membuat recordset tidak perlu bersama dengan objek Connection, seperti yang dilakukan pada RDO atau DAO. Ini yang dimaksud dengan membuat objek yang *independent*.

```
Set rsResults = New ADODB.Recordset
```

Namun objek tersebut tetap dapat dihubungkan dengan objek Connection melalui metode *Open*.

```
rsResults.Open "SELECT * FROM authors",cnADO, adOpenForwardOnly,
adLockReadOnly, adCmdText
```

Demikian juga untuk melihat jumlah *record* yang terambil, pada ADO menggunakan *property* EOF dan BOF saja, dimana *property* itu akan mengembalikan nilai *False* bila tidak ada *record* yang terambil dan *True* jika ada *record* yang terambil. Kelebihan lainnya, pada ADO dapat menggunakan fungsi **adGetRowsRest** pada **GetRows** untuk mengambil *record*. Berbeda pada RDO atau ADO yang harus melakukan pengulangan untuk mengambil *record* dengan **GetRows**.

```
myRows = rsResults.GetRows(adGetRowsRest)
```

2.5.7. Koleksi Field/Fields Collection

Pada ADO pun objek field dapat digunakan untuk mendapatkan nama kolom beserta dengan jumlahnya.

Contoh kode program Visual Basic :

```
For i = 0 To rsResults.Fields.Count - 1
    ReDim Preserve myHeadings(2, i)
    myHeadings(1, i) = rsResults.Fields(i).Name
    myHeadings(2, i) = rsResults.Fields(i).DefinedSize
Next
```

2.5.8. Metode Pencarian

Contoh kode program Visual Basic untuk metode pencarian pada ADO

```
Dim sCriteria As String
On Error Resume Next
sCriteria = "Author Like " & "'" & Text2.Text & "'"

'Set the mouse pointer to hour glass
Screen.MousePointer = 11
'Clear the List box
List2.Clear
If Check1.Value = 0 Then
    While adors.EOF = False
        adors.Find sCriteria, adSearchForward
        List2.AddItem adors.Fields("Author")
        adors.MoveNext
    Wend
Else
    adors.Find sCriteria, adSearchForward
    List2.AddItem adors.Fields("Author")
    adors.MoveNext
End If
'Get the record count
Text5.Text = List2.ListCount

'Set the mouse pointer back to default
Screen.MousePointer = 0
```

2.5.9. Prosedur Tersimpan/Stored Procedure

Teknik prosedur tersimpan *(stored procedure)* dalam *query* sangat baik karena akan mempermudah dalam hal pengubahan (*maintainability*) dan memiliki kinerja yang baik karena prosesnya yang cepat. Pada ADO,

pemanggilan prosedur yang tersimpan menjadi mudah dan sederhana dibandingkan dengan RDO atau DAO.

Contoh kode program Visual Basic :

```
Set cmADO = New ADODB.Command
strName = Trim(InputBox("Enter a department name"))
With cmADO
    .CommandText = "AddDepartment"
    .CommandTipe = adCmdStoredProc
End With
Set pmADO = cmADO.CreateParameter("name", adVarChar, adParamInput,
_
Len(strName), strName)
cmADO.Parameters.Append pmADO
Set    pmADO    =    cmADO.CreateParameter("dept_id",    adInteger,
adParamOutput, 4)
cmADO.Parameters.Append pmADO
cmADO.ActiveConnection = cnADO
cmADO.Execute
```

Pada ADO, objek parameter dapat menentukan parameter data tipe dan panjang argumen secara independen. Contoh kode program Visual Basic :

```
Set pmADO = cmADO.CreateParameter("name", _
adVarChar, adParamInput, Len(strName), strName)
cmADO.Parameters.Append pmADO
Set pmADO = cmADO.CreateParameter("dept_id", _
adInteger, adParamOutput, 4)
cmADO.Parameters.Append pmADO
```

Jika objek *Command* pada ADO siap untuk dijalankan, maka objek tersebut segera dihubungkan dengan objek *Connection* dan mengeksekusinya dengan menggunakan metode *Execute*.

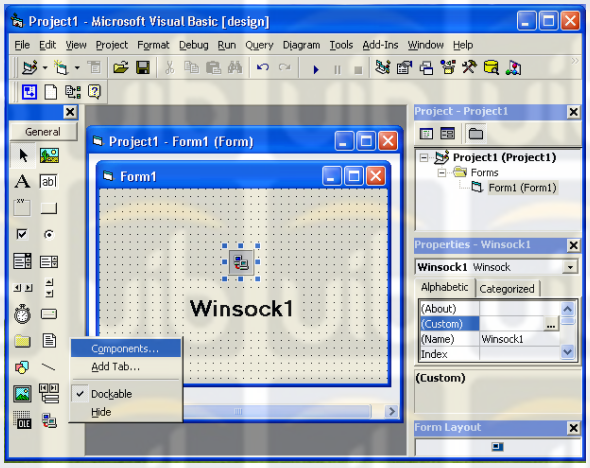
```
cmADO.ActiveConnection = cnADO
cmADO.Execute
```

2.6 Jaringan Komputer Dengan Menggunakan WinSocket

Windows socket API disingkat *Winsock* merupakan sebuah mekanisme *interprocess communication* (IPC) yang menyediakan sarana komunikasi dua arah berorientasi koneksi (*connection-oriented*) atau komunikasi tanpa

koneksi (*connectionless*) antara proses-proses di dalam dua komputer di dalam sebuah jaringan. *Windows socket* adalah salah satu implementasi yang dilakukan oleh Microsoft terhadap *system call Berkeley Sockets* yang digunakan untuk mengakses layanan sesi dan *datagram* melalui protokol TCP/IP.

Selain oleh TCP/IP, Winsock juga dapat digunakan oleh NWLink, dan AppleTalk. *Winsock* digunakan untuk merancang aplikasi *client/server* yang melibatkan transfer data lewat internet. Kontrol *Winsock* menyediakan fasilitas komunikasi antara program aplikasi berbasiskan *.htm* dengan program aplikasi *.exe* untuk mengirim atau menerima data melalui jaringan *peer to peer*. Komponen *Winsock* dapat dilihat pada Gambar 2.19.



Gambar 2.15
Komponen *Winsock* Pada *ActiveX Control*

Properti-properti yang sering digunakan pada komponen *Winsock* adalah sebagai berikut.

1. *Listen*, perintah ini digunakan untuk membaca koneksi antar program aplikasi yang menggunakan komponen *Winsock.* Kode programnya adalah "`Winsock1.Listen`";
2. *SendData*, perintah ini digunakan untuk mengirim data. Kode programnya adalah "`Winsock1.SendData`";
3. *DataArrival*, perintah ini digunakan untuk menunggu apakah ada pengiriman data. Kode programnya adalah "`Private Sub Winsock1_DataArrival`";
4. *GetData*, digunakan untuk mengambil data dari program *Winsock*. Kode programnya adalah "`Winsock1.GetData`";
5. *Close*, perintah ini digunakan untuk menutup komunikasi antar *Winsock.* Kode programnya adalah "`Winsock1.Close`" [2].

Berikut contoh *source code* sistem komunikasi *server client* dengan menggunakan *Winsock* dan *UDP protocol*.

```
Dim strData As String
Dim a As Long

With udpPeerA
  .RemoteHost = "192.168.10.11"
  .RemotePort = 1001
  .Bind 1002
End With

udpPeerA.SendData "IN" & txtUsername.Text
Enter.Enabled = False
PauseSec (5)
    If strData = "Confirm:IN" & txtUsername Then
        Unload AccessWallGate
    Else
        MsgBox ("Type the right username")
    End If
Enter.Enabled = True

Private Sub udpPeerA_DataArrival(ByVal bytesTotal As Long)
    udpPeerA.GetData strData
End Sub
```