

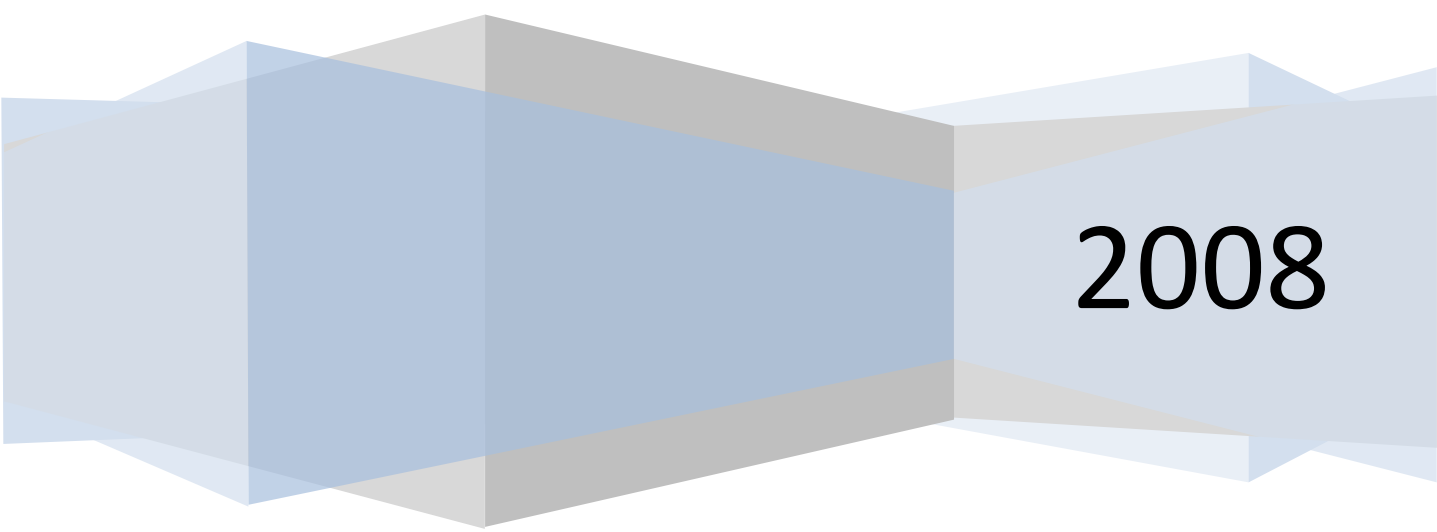
Departemen Pendidikan Nasional

Direktorat Jenderal Manajemen Pendidikan Dasar Dan Menengah

Rekayasa Perangkat Lunak

Buku Teks Sekolah Menengah Kejuruan

Aunur R. Mulyarto



2008

Aunur R. Mulyarto

REKAYASA PERANGKAT LUNAK

Untuk SMK



Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

Hak Cipta pada Departemen Pendidikan Nasional
Dilindungi Undang-undang

REKAYASA PERANGKAT LUNAK

Untuk SMK

Penulis : Aunur R. Mulyarto
Ilustrasi, Tata Letak :
Perancang Kulit :

Ukuran Buku : x cm

.....
, Mulyarto R Aunur, -----
... Rekayasa Perangkat Lunak : SMK/oleh Aunur R. Mulyarto/
Jakarta: Direktorat Pembinaan Sekolah Menengah Kejuruan, Departemen ----
Pendidikan Nasional, 2008.
vi. hlm.
ISBN-.....-.....-.....
1. Rekayasa Perangkat Lunak I. Judul

Diterbitkan oleh Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional
Tahun 2008

Diperbanyak oleh....

KATA SAMBUTAN

Puji syukur kami panjatkan kehadiran Allah SWT, berkat rahmat dan karunia Nya, Pemerintah, dalam hal ini, Direktorat Pembinaan Sekolah Menengah Kejuruan Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Departemen Pendidikan Nasional, pada tahun 2008, telah melaksanakan penulisan pembelian hak cipta buku teks pelajaran ini dari penulis untuk disebarluaskan kepada masyarakat melalui *website* bagi siswa SMK.

Buku teks pelajaran ini telah melalui proses penilaian oleh Badan Standar Nasional Pendidikan sebagai buku teks pelajaran untuk SMK yang memenuhi syarat kelayakan untuk digunakan dalam proses pembelajaran melalui Peraturan Menteri Pendidikan Nasional Nomor 12 tahun 2008.

Kami menyampaikan penghargaan yang setinggi-tingginya kepada seluruh penulis yang telah berkenan mengalihkan hak cipta karyanya kepada Departemen Pendidikan Nasional untuk digunakan secara luas oleh para pendidik dan peserta didik SMK di seluruh Indonesia.

Buku teks pelajaran yang telah dialihkan hak ciptanya kepada Departemen Pendidikan Nasional tersebut, dapat diunduh (*download*), digandakan, dicetak, dialihmediakan, atau difotokopi oleh masyarakat. Namun untuk penggandaan yang bersifat komersial harga penjualannya harus memenuhi ketentuan yang ditetapkan oleh Pemerintah. Dengan ditayangkannya *soft copy* ini akan lebih memudahkan bagi masyarakat untuk mengaksesnya sehingga peserta didik dan pendidik di seluruh Indonesia maupun sekolah Indonesia yang berada di luar negeri dapat memanfaatkan sumber belajar ini.

Kami berharap, semua pihak dapat mendukung kebijakan ini. Selanjutnya, kepada para peserta didik kami ucapkan selamat belajar dan semoga dapat memanfaatkan buku ini sebaik-baiknya. Kami menyadari bahwa buku ini masih perlu ditingkatkan mutunya. Oleh karena itu, saran dan kritik sangat kami harapkan.

Jakarta,
Direktur Pembinaan SMK

Pengantar Penulis

Dengan segala kerendahan hati, kami mengucapkan syukur kepada Allah SWT. Karena hanya dengan lindungan, rahmat dan karuniaNya-lah maka buku ini dapat diselesaikan.

Buku yang berjudul 'Rekayasa Perangkat Lunak' merupakan buku yang disusun untuk memenuhi kebutuhan buku pegangan bagi siswa Sekolah Menengah Kejuruan. Khususnya pada program keahlian Rekayasa Perangkat Lunak.

Pada kesempatan ini kami ingin menyampaikan terima kasih yang sebesar-besarnya kepada :

- Direktur Pembinaan SMK atas kesempatan yang diberikan pada penulis untuk menyusun dan menyelesaikan buku ini.
- Prof.Dr.Ir. Harijono, M.App.Sc selaku Dekan Fakultas Teknologi Pertanian Universitas Brawijaya Malang, yang telah memberikan ijin dan kemudahan bagi kami dalam melaksanakan kegiatan penulisan buku.
- Rekan-rekan sejawat di Jurusan Teknologi Industri Pertanian, Fakultas Teknologi Pertanian, Universitas Brawijaya Malang yang telah memberikan bantuan baik moril maupun materiil selama penulisan buku ini berlangsung.
- Rekan-rekan di Direktorat Pembinaan SMK yang telah banyak membantu dalam masalah-masalah administratif selama berlangsungnya penulisan buku ini.

Demi penyempurnaan buku ini, kami sangat mengharapkan saran, kritik dan masukan. Akhirnya kami berharap semoga laporan ini bermanfaat. Amien.

Malang, Desember 2008

Penulis

Daftar Isi

	Halaman
PENGANTAR PENULIS	i
PENGANTAR DIREKTUR PEMBINAAN SMK	ii
DAFTAR ISI	iv
DAFTAR GAMBAR	xi
DAFTAR TABEL	xxi
DAFTAR LAMPIRAN	xxii
DAFTAR ISTILAH / GLOSARI	xxiii
SINOPSIS	xxvi
PETA KOMPETENSI	xxvii
1 PENDAHULUAN	1
1.1. Pengertian	1
1.2. Tujuan Rekayasa Perangkat Lunak	2
1.3. Ruang Lingkup	3
1.4. Rekayasa Perangkat Lunak Dan Disiplin Ilmu Komputer	4
1.5. Rekayasa Perangkat Lunak Dan Disiplin Ilmu Lain	6
1.6. Perkembangan Rekayasa Perangkat Lunak	7
1.7. Profesi Dan Sertifikasi	8
1.8. Tentang Buku Ini	9
1.8.1. Isi	9
1.8.2. Bagaimana menggunakan buku ini	10
2 PRINSIP PEMECAHAN MASALAH DALAM REKAYASA PERANGKAT LUNAK	13
2.1. Masalah	11
2.1.1. Pengertian	11
2.1.2. Tipe-tipe masalah	13
2.2. Pemecahan Masalah	15
2.2.1. Proses	15
2.2.2. Strategi	16
2.3. Solusi Masalah	21
2.4. Pendekatan Sistem Dalam Pemecahan Masalah	22
2.4.1. Sistem	22
2.4.2. Pendekatan Sistem	24
3 METODE REKAYASA PERANGKAT LUNAK	25
3.1. Model Proses Rekayasa Perangkat Lunak	25
3.1.1. The Waterfall Model	27

3.1.2.	Prototyping Model	28
3.1.3.	Unified Process dan Unified Modeling Language	30
3.2.	Tahapan Rekayasa Perangkat Lunak	33
3.2.1.	Analisis	33
3.2.2.	Disain	38
3.2.3.	Konstruksi	39
3.2.4.	Pengujian	39
3.2.5.	Perawatan dan Konfigurasi	40
4	DASAR SISTEM KOMPUTER	41
4.1.	Data	41
4.2.	Informasi	41
4.3.	Sistem Komputer	43
4.4.	Perangkat Keras	44
4.5.	Perangkat Lunak	52
4.5.1.	Klasifikasi	52
5	SISTEM OPERASI	57
5.1.	Fungsi	57
5.1.1.	Antar muka pengguna	57
5.1.2.	Manajemen sumberdaya	60
5.1.3.	Manajemen file	60
5.1.4.	Manajemen proses	61
5.1.5.	Manajemen system masukan dan keluaran (I / O)	62
5.2.	BIOS	63
5.3.	Jenis-Jenis Sistem Operasi	64
5.3.1.	DOS	64
5.3.2.	UNIX	65
5.3.3.	Microsoft Windows	68
5.3.4.	Apple Mac OS	70
5.3.5.	Linux	71
5.4.	Menjalankan Sistem Operasi	72
5.4.1.	Instalasi	73
5.4.2.	Bootting	76
5.4.3.	Perintah berbasis teks	78
5.4.4.	Bekerja dengan GUI	89
6	JARINGAN KOMPUTER	91
6.1.	Topologi	91
6.2.	Jangkauan	94
6.3.	Protocol dan Tcp/Ip	96
6.4.	Perangkat Keras Jaringan	98
6.5.	Bekerja Dalam Komputer Jaringan	99
6.5.1.	Persiapan	100
6.5.2.	Konfigurasi koneksi jaringan	103

6.5.3.	Berbagi file, printer, dan sumber daya lain	103
7	DASAR-DASAR PEMROGRAMAN	106
7.1.	Program dan Bahasa Pemrograman	106
7.2.	Klasifikasi Bahasa Pemrograman	107
7.3.	Interpreter dan Compiler	109
7.4.	Tahapan-Tahapan Pemrograman	110
7.5.	Teknik Pemrograman	113
7.5.1.	Pemrograman Prosedural	113
7.5.2.	Pemrograman Terstruktur	114
7.5.3.	Pemrograman Berorientasi Obyek	117
7.6.	Pemilihan Bahasa Pemrograman	119
8	TIPE DATA, VARIABEL, KONSTANTA, DAN OPERATOR	121
8.1.	Tipe Data	121
8.1.1.	Numerik	122
8.1.2.	Character	123
8.1.3.	Boolean	124
8.1.4.	Array	124
8.1.5.	Record	124
8.1.6.	Image	124
8.1.7.	Date Time	125
8.1.8.	Tipe data lain	125
8.2.	Variabel	127
8.3.	Konstanta	129
8.4.	Operator	130
8.4.1.	Operator aritmatika	130
8.4.2.	Operator perbandingan	131
8.4.3.	Operator logika	132
9	ALGORITMA PEMROGRAMAN DASAR	133
9.1.	Pengertian Algoritma	133
9.2.	Cara Penulisan Algoritma	135
9.3.	Flowchart	136
9.4.	Struktur berurutan	138
9.5.	Struktur pemilihan / percabangan	141
9.5.1.	Struktur pemilihan sederhana	142
9.5.2.	Struktur pemilihan bersarang (nested)	143
9.6.	Struktur Pengulangan	147
9.6.1.	Struktur pengulangan dengan For	147
9.6.2.	Struktur pengulangan dengan While, Loop, dan Repeat	150
10	ALGORITMA PEMROGRAMAN LANJUTAN	153
10.1.	Prosedur	153

10.2.	Array	156
10.2.1.	Array satu dimensi	157
10.2.2.	Array multi-dimensi	158
11	PENERAPAN ALGORITMA DALAM BAHASA PEMROGRAMAN	161
11.1.	Sekilas Tentang Bahasa Pemrograman Python	161
11.1.1.	Menjalankan Python	162
11.2.	Variabel, Konstanta, Tipe Data dan Operator	165
11.3.	Perintah Dasar Input - Output	167
11.4.	Penerapan Struktur Algoritma	168
11.4.1.	Struktur Berurutan	168
11.4.2.	Struktur Pemilihan	170
11.4.3.	Struktur Kendali Pengulangan	176
11.5.	Fungsi	182
11.5.1.	Fungsi built in	182
11.5.2.	Membuat fungsi	183
11.6.	Modul	184
11.6.1.	Modul dalam standar library	185
11.6.2.	Membuat Modul	187
12	PEMROGRAMAN BERBASIS GUI	189
12.1.	Pengenalan Pada Visual Basic	189
12.1.1.	Memulai Visual Basic	190
12.1.2.	IDE Visual Basic	191
12.1.3.	Toolbox	193
12.1.4.	Mengatur Lingkungan Kerja Visual Basic	195
12.2.	Prinsip Pokok Pemrograman Berbasis GUI	197
12.2.1.	Visual Design	197
12.2.2.	Event-Driven Programming	199
12.3.	Struktur Kendali	199
12.4.	Mengenai Procedure dan Fungsi	206
12.4.1.	Sub Procedure	207
12.4.2.	Function Procedure	207
12.4.3.	Fungsi built-in	211
13	PENCARIAN KESALAHAN DAN PENGUJIAN	212
13.1.	Jenis-Jenis Kesalahan	212
13.1.1.	Run Time Errors	213
13.1.2.	Logical Errors	218
13.1.3.	Precision Errors	219
13.2.	Teknik Pencarian Kesalahan	220
13.3.	Pengujian	221
13.4.	Dokumentasi	222

14	DASAR-DASAR SISTEM BASIS DATA	223
14.1.	Pengertian Basis Data	223
14.1.1.	Operasi dasar basis data	225
14.1.2.	Tujuan basis data	226
14.2.	Sistem Basis Data	227
14.2.1.	Komponen sistem basis data	227
14.3.	Tipe Basis Data	230
14.4.	Tahapan Disain Sistem Basis Data	231
14.4.1.	Data Planning	232
14.4.2.	Requirements Specification	232
14.4.3.	Conceptual Design	232
14.4.4.	Logical Design	233
14.4.5.	Physical Design	233
14.5.	Perangkat Lunak DBMS	233
14.5.1.	Microsoft Access	233
14.5.2.	MyQSL	234
14.5.3.	Microsoft SQL Server	236
14.5.4.	PostgreSQL	237
14.5.5.	Oracle Database	237
15	MODEL DATA	238
15.1.	Diagram Entity-Relationship	239
15.2.	Entitas	240
15.3.	Atribut	241
15.4.	Relationship	244
15.5.	Kardinalitas	245
16	BASIS DATA RELASIONAL	247
16.1.	Model Basis Data	247
16.1.1.	File System (Flat File)	248
16.1.2.	Hierarchical Model	248
16.1.3.	Network Model	249
16.1.4.	Relational Model	249
16.1.5.	Object Model	250
16.1.6.	Object-Relational Model	250
16.2.	Basis Data Relasional	250
16.2.1.	Row/Baris/Tuple	252
16.2.2.	Field/Kolom/Attributes	252
16.3.	Relasi Antar Tabel	255
16.4.	Normalisasi	260
16.4.1.	Functional Dependency	260
16.4.2.	Super-Key, Candidate-Key dan Primary-Key	261
16.4.3.	Bentuk-Bentuk Normal dan Proses Normalisasi	263
16.5.	Query	266

17	MENGELOLA BASIS DATA DENGAN DBMS	269
17.1.	Memulai Microsoft Access	272
17.2.	Tabel	273
17.2.1.	Pendefinisian Field dan Tipe Data.	273
17.2.2.	Pengisian Data Pada Tabel	276
17.3.	Query	277
17.3.1.	Membuat Relasi Antar Tabel	278
17.3.2.	Membuat Query	280
17.4.	Form	288
17.4.1.	Membuat Form	289
17.4.2.	Memodifikasi Form	293
17.5.	Report	299
18	SQL	308
18.1.	Bahasa Basis Data	308
18.1.1.	Data Definiton Language	308
18.1.2.	Data Manipulation Language	309
18.2.	Sql	309
18.2.1.	Pembuatan Basis Data	310
18.2.2.	Pembuatan Tabel	311
18.2.3.	Perubahan Pada Atribut Tabel	312
18.2.4.	Pengisian Data Pada Tabel	313
18.2.5.	Membuat Query	314
18.2.6.	Update Data	322
18.2.7.	Menghapus record, table dan basis data	323
18.2.8.	Menggunakan opsi dan perintah tambahan	324
19	DASAR-DASAR PEMROGRAMAN BERBASIS WEB	328
19.1.	Intranet, Ekstranet Dan Internet	328
19.2.	Web Dan Pemrograman Web	329
19.3.	Client-Side Dan Server-Side	330
19.4.	Pengertian Web Statis Dan Dinamis	332
19.5.	Perangkat Lunak untuk Pemrograman Web	334
19.5.1.	Sistem operasi	334
19.5.2.	Web server	335
19.5.3.	Perangkat pengembang web	340
19.5.4.	Sistem Manajemen Basis Data	341
19.5.5.	Web Browser	341
19.6.	Bahasa Pemrograman Berbasis Web (Script)	343
19.6.6.	ASP dan ASP.net	343
19.6.7.	PHP	344
20	HTML	345
20.1.	Pengertian tentang HTML	345

20.2.	Sintaks dan Struktur Umum HTML	347
20.2.1.	Header	349
20.2.2.	Body	350
20.3.	Format Dokumen	350
20.3.3.	Heading	351
20.3.4.	Paragraph	351
20.3.5.	Line Break	351
20.3.6.	List	352
20.3.7.	Font	354
20.3.8.	Link	355
20.3.9.	Horizontal Line	356
20.3.10.	Image	356
20.3.11.	Tabel	358
20.4.	Bekerja Dengan Form	361
20.4.12.	Tipe Input	361
20.5.	Style Sheet (CSS)	365
21	PEMROGRAMAN WEB	374
21.1.	Pengenalan Pada PHP	374
21.1.1.	Aturan-aturan umum pada PHP g	375
21.1.2.	PHP dan HTML	377
21.2.	Variabel dan Tipe Data	378
21.2.1.	Variabel	378
21.2.2.	Tipe data	379
21.2.3.	Operator	381
21.3.	Translasi algoritma ke bahasa pemrograman web	382
21.3.1.	Struktur berurutan	382
21.3.2.	Struktur pemilihan	383
21.3.3.	Struktur pengulangan	385
21.4.	Require dan Include	388
21.5.	Fungsi	391
21.6.	Bekerja Dengan Sistem Manajemen Basis Data	395
21.6.1.	Koneksi	395
21.6.2.	Manipulasi data	396
21.7.	Contoh aplikasi PHP	398
	DAFTAR PUSTAKA	413
	LAMPIRAN	

Daftar Gambar

No.	Judul Gambar	Halaman
1.1.	Tujuan rekayasa perangkat lunak.	2
1.2.	Ruang lingkup rekayasa perangkat lunak (Abran et.al., 2004).	3
1.3.	Klasifikasi disiplin ilmu komputer menurut ACM (1998).	5
1.4.	Klasifikasi disiplin ilmu komputer menurut Denning (2000).	5
1.5.	Klasifikasi disiplin ilmu komputer menurut Wikipedia (2007).	6
1.6.	Keterkaitan Rekayasa Perangkat Lunak dengan bidang ilmu lain.	7
1.7.	Perkembangan Rekayasa Perangkat Lunak.	8
2.1.	Kondisi aktual, target dan masalah.	12
2.2.	Gejala dan masalah.	13
2.3.	Tipe-tipe masalah (Deek et al, 2005).	14
2.4.	Proses pemecahan masalah (diadopsi dari Deek et al, 2005)	15
2.5.	Klasifikasi strategi pemecahan masalah.	17
2.6.	Strategi dekomposisi masalah.	18
2.7.	Pemecahan masalah dengan library.	19
2.8.	Strategi problem expansion.	19
2.9.	Strategi pemecahan masalah linear.	20
2.10.	Strategi pemecahan masalah iterative.	20
2.11.	Strategi pemecahan masalah parallel.	21
2.12.	Sistem.	23
2.13.	Sistem dan komponen-komponennya (O'Brien, 1999).	23
2.14.	Tahapan-tahapan pendekatan sistem dalam penyelesaian masalah (O'Brien, 1999).	24
3.1.	System Development Life Cycle (SDLC)	26
3.2.	The Waterfall Model	27
3.3.	Klasifikasi prototyping model (Harris, 2003)	28
3.4.	Tahapan-tahapan prototyping model (Harris, 2003)	31
3.5.	RUP Life Cycle (Ambler, 2005).	31
3.6.	Tahapan dan aktifitas dalam analisis.	34
3.7.	Notasi pada DFD.	35
3.8.	Tahapan pembuatan DFD.	35

3.9.	Context diagram sistem pemesanan makanan (Hoffer et al., 2002).	36
3.10.	DFD Level 0.	37
3.11.	Tipe-tipe perawatan.	40
4.1.	Data, proses, informasi dan pengguna	42
4.2.	Sistem Komputer	44
4.3.	Komponen dasar komputer	44
4.4.	Perangkat keras komputer	46
4.5.	Display atau monitor	47
4.6.	Motherboard sebuah komputer	47
4.7.	Central Processing Unit (CPU)	48
4.8.	Berbagai jenis main memory	48
4.9.	Pemasangan expansion card	49
4.10.	Power Supply Unit	49
4.11.	CD-RW Drive, salah satu contoh Optical Disc Drive	50
4.12.	Hard Disk	51
4.13.	Skema umum sebuah keyboard	51
4.14.	Berbagai jenis mouse	52
4.15.	Tampilan desktop sistem operasi Windows XP	53
4.16.	Application software Microsoft Word (Software pengolah kata).	54
4.17.	Application software Winrar (Software kompresi dan ekstraksi file).	55
4.18.	Application software PowerSim (Software untuk simulasi sistem)	55
4.19.	Application software Hysis (Software untuk perancangan pabrik).	56
5.1.	Fungsi-fungsi sistem operasi	58
5.2.	Antar muka pengguna model CLI pada Linux / Unix	59
5.3.	Antar muka pengguna model GUI pada Linux Fedora	59
5.4.	Manajemen memori pada sistem operasi Microsoft Windows	60
5.5.	Windows Explorer sebagai sarana pengelolaan file	61
5.6.	Manajemen proses pada sistem operasi Microsoft Windows	62
5.7.	Manajemen I / O pada sistem operasi Microsoft Windows.	63
5.8.	Tampilan BIOS utility	64
5.9.	Contoh penggunaan DOS	65
5.10.	Unix dan sistem operasi turunannya	66
5.11.	Manajemen memori dan penjadwalan proses pada Unix	67
5.12.	X windows system di UNIX	68
5.13.	Windows versi 3.11	69

5.14.	Windows Vista	69
5.15.	Mac OS versi awal	70
5.16.	Mac OS X	71
5.17.	Linux dengan desktop KDE	72
5.18.	Tahapan-tahapan instalasi	74
5.19.	Testing media instalasi	74
5.20.	Proses penentuan target instalasi	75
5.21.	Proses copy file pada Fedora	76
5.22.	Proses awal booting	77
5.23.	Proses booting pada Linux Fedora	78
5.24.	Terminal sedang menjalankan mode CLI	79
5.25.	Perintah-perintah pada direktori sbin	79
5.26.	Perintah-perintah pada direktori /usr/sbin.	80
5.27.	Perintah-perintah pada direktori bin.	80
5.28.	Perintah-perintah pada direktori /usr/bin	81
5.29.	Contoh penggunaan perintah ls	81
5.30.	Contoh penggunaan perintah cd	82
5.31.	Contoh penggunaan perintah find	82
5.32.	Contoh penggunaan perintah cat dan more	83
5.33.	Contoh penggunaan perintah cp	83
5.34.	Contoh penggunaan perintah mv untuk memindahkan file.	84
5.35.	Contoh penggunaan perintah mv untuk mengganti nama file.	84
5.36.	Contoh penggunaan perintah rm untuk menghapus file atau direktori	85
5.37.	Contoh penggunaan perintah mkdir	85
5.38.	Attribute file / folder pada Microsoft WIndows	86
5.39.	Attribute file / direktori pada keluarga Unix	87
5.40.	Eksekusi perintah ps	88
5.41.	Penggunaan perintah df.	89
5.42.	Contoh hasil eksekusi perintah man untuk melihat manual suatu perintah.	89
5.43.	Membuka konteks menu dengan klik kanan.	90
5.44.	Drag and drop	90
6.1.	Topologi ring	92
6.2.	Topologi bus	93
6.3.	Topologi Star	94
6.4.	Jaringan LAN	94
6.5.	Jaringan MAN	95
6.6.	Jaringan WAN	96
6.7.	Skema Layer TCP/IP	97
6.8.	Perangkat keras jaringan	98
6.9.	Network Interface Card	98

6.10.	Cisco Router	99
6.11.	Switch	99
6.12.	Hub	99
6.13.	Membuka system properties	101
6.14.	Device manager	101
6.15.	Output perintah lspci untuk memeriksa network adapter..	102
6.16.	Memeriksa protocol TCP/IP	102
6.17.	Kondisi koneksi jaringan	103
6.18.	Mengatur file sharing	104
6.19.	Menjelajah komputer yang ada di jaringan	105
6.20.	Printer sharing	105
7.1.	Klasifikasi bahasa pemrograman berdasarkan kedekatannya dengan perangkat keras	108
7.2.	Klasifikasi bahasa pemrograman berdasarkan tujuan dan fungsinya	109
7.3.	Proses eksekusi interpreter	109
7.4.	Proses eksekusi compiler	110
7.5.	Tahapan-tahapan dalam pemrograman	111
7.6.	Pemrograman prosedural	113
7.7.	Top-Down Design	115
7.8.	Class, object, attribute dan method	117
7.9.	Pewarisan	118
7.10.	Contoh abstraction	119
8.1.	Pengelompokkan tipe data	122
9.1.	Notasi Flowchart	137
9.2.	Program flowchart	138
9.3.	Flowchart untuk penyelesain contoh 9.4.	140
9.4.	Flowchart penyelesaian untuk contoh 9.6	141
9.5.	Flowchart penyelesaian masalah nonton film	142
9.6.	Flowchart penyelesaian untuk contoh 9.8.	143
9.7.	Flowchart penyelesaian untuk contoh 9.9	145
9.8.	Flowchart penyelesaian untuk contoh 9.10	146
9.9.	Struktur algoritma pengulangan dengan For.	148
9.10.	Flowchart penyelesaian untuk contoh 9.13	149
9.11.	Struktur pengulangan dengan While..Wend	150
9.12.	Contoh penggunaan While .. wend	151
9.13.	Flowchart penyelesaian untuk contoh 9.16	152
10.1.	Skema penggunaan prosedur	154
10.2.	Penyelesaian contoh 10.2	155
10.3.	Lemari dengan banyak kotak laci di dalamnya	157

10.4.	Perbedaan array satu dimensi dan dua dimensi	159
11.1.	Jendela IDLE atau Python Shell	163
11.2.	Mode interaktif	163
11.3.	Jendela editor baru pada IDLE	164
11.4.	Proses eksekusi kode program pada mode skrip	164
12.1.	Tampilan awal Visual Basic	190
12.2.	Tampilan awal untuk pilihan Standard.EXE	191
12.3.	IDE Visual Basic	192
12.4.	Toolbox VB 6	194
12.5.	Jendela pengaturan editor	196
12.6.	Jendela pengaturan format editor	196
12.7.	Jendela pengaturan environment	197
12.8.	Obyek, Property, Method dan Event	198
13.1.	Jenis-jenis kesalahan	213
13.2.	Kesalahan penulisan reserve word	213
13.3.	Kesalahan penggunaan nama variable	214
13.4.	Cara penulisan pernyataan yang salah	215
13.5.	Kesalahan out of data	215
13.6.	Pembacaan data jika jumlah data lebih banyak	216
13.7.	Kesalahan penggunaan tipe data	216
13.8.	kesalahan karena division by zero	217
13.9.	Kesalahan penggunaan argument	218
13.10.	Tahapan-tahapan dalam pencarian kesalahan	220
14.1.	Lemari arsip dan basis data	224
14.2.	Tingkatan dalam abstraksi data (Lewis et al., 2002)	225
14.3.	Operasi-operasi dasar pada basis data	226
14.4.	Komponen-komponen sistem basis data dan keterkaitannya	227
14.5.	Struktur umum DBMS	229
14.6.	Klasifikasi pengguna sistem basis data	230
14.7.	Tipe-tipe basis data (O'Brien, 1999)	231
14.8.	Tahapan disain basis data (diadopsi dari O'Brien, 1999)	232
14.9.	Logo Microsoft Access	233
14.10.	Tampilan Microsoft Access	234
14.11.	Logo MySQL	235
14.12.	Tampilan awal phpMyAdmin	235
14.13.	Logo Microsoft SQL Server	236
14.14.	GUI pada Microsoft SQL Server	236
14.15.	Logo PostgreSQL	236
14.16.	Logo Oracle	236

15.1.	Notasi entitas pada ER-Diagram	239
15.2.	Penggunaan notasi atribut pada ER-Diagram	239
15.3.	Penggunaan notasi relationship pada ER-Diagram	240
15.4.	Entitas siswa dan atributnya	241
15.5.	Entitas guru dan atributnya	242
15.6.	Entitas mobil dan atributnya	242
15.7.	Relationship	244
15.8.	Hubungan one-to-one suami dan istri	245
15.9.	Hubungan one-to-many kelas dengan siswa	245
16.1.	Model basis data (Powell, 2006)	247
16.2.	Flat file	248
16.3.	Hierarchical model (Powell, 2006)	249
16.4.	Network model (Powell, 2006).	249
16.5.	Relational Model (Powell, 2006)	250
16.6.	Hubungan table/file/relation, row/record/tuple dan column/field/attribute	251
16.7.	Kolom, constraint dan tipe data (Powell, 2006).	253
16.8.	Contoh atribut sederhana	254
16.9.	Contoh atribut komposit	254
16.10.	Contoh atribut bernilai tunggal dan atribut bernilai banyak.	255
16.11.	Tabel Pengarang	255
16.12.	Tabel Penerbit	256
16.13.	Tabel buku	256
16.14.	ER-Diagram untuk Penerbit dan Buku	257
16.15.	Hubungan table Penerbit dan Buku	257
16.16.	ER-Diagram untuk Pengarang – Buku	258
16.17.	Hubungan table Pengarang dan Buku	259
16.18.	Relasi antar table	259
16.19.	Contoh ketergantungan fungsional	261
16.20.	Table siswa	262
16.21.	Table Guru dengan salah satu kolom bernilai banyak	263
16.22.	Table Guru dengan semua kolom sudah bernilai tunggal	264
16.23.	Hasil dekomposisi tabel	265
17.1.	ER Diagram untuk kasus Basis Data Penjualan Buku	270
17.2.	Tampilan awal Microsoft Access	272
17.3.	Penentuan nama dan lokasi basis data	272
17.4.	Bagian-bagian sebuah basis data pada Microsoft Access	273
17.5.	Tahap awal pembuatan table	274
17.6.	Pendefinisian field, tipe data, constraint dan domain	274
17.7.	Toolbar Microsoft Access	275

17.8.	Struktur table pembeli	275
17.9.	Struktur table buku	275
17.10.	Struktur table pesanan	276
17.11.	Struktur table item_pesanan	276
17.12.	Hasil pengisian data pada table pembeli	276
17.13.	Hasil pengisian data pada table buku	277
17.14.	Hasil pengisian data pada table pesanan	277
17.15.	Hasil pengisian data pada table item_pesanan	277
17.16.	Jendela Relationships	278
17.17.	Jendela Show Table	278
17.18.	Tabel-tabel yang akan direlasikan	278
17.19.	Jendela untuk edit relationships	279
17.20.	Relasi untuk keseluruhan table	279
17.21.	Jendela query pada mode design view	280
17.22.	Prosedur dan hasil query table buku	281
17.23.	Query nama pengarang dan bukunya	282
17.24.	Query judul buku dan harga dengan urutan	283
17.25.	Query dengan criteria tertentu	283
17.26.	Query dengan menggunakan operator and	284
17.27.	Query dengan menggunakan operator or	284
17.28.	Pemilihan table untuk query dua table.	285
17.29.	Query dua tabel	286
17.30.	Query tiga table.	287
17.31.	Query empat table	288
17.32.	Jenis-jenis form	289
17.33.	Membuka jendela Form Wizard	289
17.34.	Pemilihan table yang akan dibuat formnya	290
17.35.	Pemilihan field untuk form	290
17.36.	Jendela untuk memilih model tampilan form	291
17.37.	Jendela untuk memilih style form	291
17.38.	Jendela untuk memberi nama form	292
17.39.	Form Pembeli	292
17.40.	Form Buku	293
17.41.	Jendela Form Pembeli pada mode Design View	293
17.42.	Bagian-bagian suatu form	294
17.43.	Perubahan pada Label fields pada Form Pembeli	295
17.44.	Modifikasi tampilan form	296
17.45.	Mendefinisikan aksi untuk suatu Command Button	296
17.46.	Mendefinisikan teks pada Command Button	297
17.47.	Mendefinisikan nama Command Button	297
17.48.	Mendefinisikan nama Command Button	298
17.49.	Hasil modifikasi Form Pembelian	299
17.50.	Jendela query untuk sumber report	300
17.51.	Pemilihan query sebagai sumber data laporan.	300
17.52.	Pemilihan fields yang terlibat	301

17.53.	Jendela untuk menentukan dasar tampilan report	302
17.54.	Jendela untuk menentukan grouping data	302
17.55.	Jendela untuk menentukan urutan data	303
17.56.	Jendela untuk mengatur tampilan ringkasan	303
17.57.	Jendela untuk mengatur lay-out dan orientation	304
17.58.	Jendela untuk mengatur style laporan	304
17.59.	Hasil pembuatan laporan menggunakan Wizard	305
17.60.	Laporan dalam mode Design View	305
17.61.	Design laporan setelah dilakukan perbaikan	306
17.62.	Print Preview laporan setelah perbaikan	307
18.1.	Contoh perintah SQL untuk pembuatan basis data baru	310
18.2.	Perintah pembuatan pembeli	311
18.3.	Perintah pembuatan table buku	311
18.4.	Perintah <code>show tables</code> untuk melihat daftar table	312
18.5.	Penggunaan perintah <code>describe</code>	312
18.6.	Penggunaan perintah <code>alter</code>	313
18.7.	Penggunaan perintah <code>insert</code>	314
18.8.	Memasukkan beberapa record data sekaligus	314
18.9.	Contoh query untuk menampilkan semua data dari sebuah tabel.	316
18.10.	Contoh query untuk menampilkan semua data pada kolom tertentu	317
18.11.	Penggunaan pernyataan <code>where</code> sederhana	317
18.12.	Contoh lain penggunaan <code>where</code>	318
18.13.	Pernyataan <code>where</code> dengan menggunakan operator logika	319
18.14.	Penggabungan dua table	320
18.15.	Contoh lain operasi join	320
18.16.	Operasi join tiga buah table	321
18.17.	Operasi join empat buah table	322
18.18.	Penggunaan perintah <code>update</code>	323
18.19.	Penggunaan perintah <code>delete</code>	323
18.20.	Penggunaan perintah <code>drop table</code>	324
18.21.	Penggunaan perintah <code>drop database</code> .	324
18.22.	Mengurutkan tampilan dengan opsi <code>order by</code>	325
18.23.	Penggunaan pernyataan <code>min</code> dan <code>max</code> .	325
18.24.	Penggunaan pernyataan <code>sum</code> dan <code>avg</code>	326
18.25.	Penggunaan pernyataan <code>count</code>	326
18.26.	Penggunaan pernyataan <code>group by</code>	327
19.1.	Client side dan server side	332
19.2.	Halaman awal situs Google	333
19.3.	Halaman web dinamis	334
19.4.	Memeriksa service Apache pada Linux	336

19.5.	Memeriksa service Apache pada Microsoft Windows	337
19.6.	Lokasi file konfigurasi Apache pada Linux (distro Mandriva).	338
19.7.	Lokasi file konfigurasi Apache pada Microsoft Windows (menggunakan AppServ).	338
19.8.	Memeriksa dan menginstal IIS	340
19.9.	Microsoft Internet Explorer	342
19.10.	Safari.	342
19.11.	Opera.	343
20.1.	Teks editor Notepad	346
20.2.	Macromedia Dreamweaver	346
20.3.	Quanta pada system operasi Linux	347
20.4.	Bluefish pada system operasi Linux	347
20.5.	Struktur umum dokumen HTML	349
20.6.	Header dokumen HTML tanpa tag title	349
20.7.	Header dokumen HTML dengan tag title	349
20.8.	Dokumen HTML dengan body content sederhana	350
20.9.	Dokumen HTML dengan body content yang lebih kompleks	350
20.10.	Penggunaan heading	351
20.11.	Penggunaan paragraph	351
20.12.	Tag <code>
</code> dan <code><P></code> .	352
20.13.	Penggunaan Ordered List	352
20.14.	Penggunaan Unordered List	353
20.15.	Penggunaan Direktori List	353
20.16.	Penggunaan Menu List	353
20.17.	Penggunaan Definition list	354
20.18.	Penggunaan tag Font	355
20.19.	Penggunaan tag anchor	355
20.20.	Penggunaan garis	356
20.21.	Penggunaan tag image	357
20.22.	Penggunaan attribute-tag IMG	357
20.23.	Table sederhana	358
20.24.	Tabel dengan format yang lebih kompleks	359
20.25.	Cellpadding, cellspacing dan border	359
20.26.	Rowspan.	360
20.27.	Colspan.	360
20.28.	Tabel dengan sel berisi gambar	361
20.29.	Penggunaan input tipe TEXT	362
20.30.	Penggunaan input tipe PASSWORD	362
20.31.	Penggunaan input tipe CHECKBOX	363
20.32.	Penggunaan input tipe RADIO	363
20.33.	Penggunaan input tipe SUBMIT	364
20.34.	Penggunaan input tipe RESET	364

20.35.	Penggunaan tipe TEXTAREA	365
20.36.	Penggunaan tipe SELECT	365
21.1.	Hasil eksekusi contoh 21.1	374
21.2.	Hasil eksekusi contoh 21.2	375
21.3.	Lokasi direktori yang bisa dibaca web server	376
21.4.	Cara menjalankan file dengan ekstensi .php	377
21.5.	Menyisipkan skrip PHP pada dokumen HTML	378
21.6.	Menyisipkan menyisipkan tag HTML ke dokumen PHP	378
21.7.	Contoh penggunaan variable	380

Daftar Tabel

No.	Judul Tabel	Halaman
3.1.	Aturan-aturan dalam DFD	38
4.1.	Karakteristik kualitas informasi (O'Brien, 1999).	42
5.1.	Perintah yang berhubungan dengan pengelolaan file/direktori	87
8.1.	Operator Aritmatika	130
8.2.	Operator perbandingan	131
8.3.	Operator logika	132
16.1.	Super-key, candidate key, dan primary key untuk Table Siswa	262
16.2.	Tabel-tabel hasil dekomposisi	266
17.1.	Tabel dan atribut pada Basis Data Penjualan Buku	271
17.2.	Tabel, atribut, tipe data dan constraint/domain pada Basis Data Penjualan Buku	271
20.1.	Daftar attribute TYPE untuk Ordered list dan Unordered list	354
20.2.	Bagian-bagian pada tag Table	358
20.3.	Attribute-attribute tag <INPUT>	362
21.1.	Jenis-jenis operator	382

Daftar Lampiran

No.	Judul Lampiran	Halaman
1	Daftar Alamat Situs	401
2	Fungsi Built-in pada Visual Basic.	403

Daftar Istilah / Glosari

Basis data (database)

Kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya

Constraint

Batasan-batasan dari masalah

Control

Aktivitas monitoring dan evaluasi terhadap feedback untuk menentukan apakah system telah bekerja dengan baik atau tidak

Database Management System (DBMS)

Perangkat Lunak yang khusus / spesifik ditujukan untuk pengelolaan basis data

Feedback

Data tentang kinerja system

Gejala

Signal atau tanda terjadinya suatu masalah

Givens

Representasi dari fakta-fakta spesifik yang harus diidentifikasi sebelum proses pemecah masalah dilakukan

Goals

Sesuatu yang ingin kita tuju atau selesaikan

Ilmu komputer

Suatu studi sistematis pada proses-proses algoritma yang menjelaskan dan mentransfor-masikan informasi

Input

Elemen-elemen yang masuk ke dalam system

Masalah (problem)

Perbedaan antara situasi aktual dan situasi yang diharapkan atau perbedaan antara kondisi sekarang dengan target atau tujuan yang diinginkan

Model

Penyederhanaan dari suatu system atau Tiruan dari suatu system dengan sedikit atau banyak penyederhanaan

Output

Perpindahan elemen-elemen yang dihasilkan dari proses perubahan ke tujuan yang diinginkan

Pemecahan masalah

Sebuah proses dimana suatu situasi dianalisa kemudian solusi-solusi dibuat bila ditemukan ada masalah dengan cara pendefinisian, pengurangan atau penghilangan, atau pencegahan masalah

Pendekatan sistem

Pendekatan yang berorientasi pada system untuk mendefinisikan masalah dan kesempatan dan mengembangkan solusinya

Perangkat lunak

Seluruh instruksi yang digunakan untuk memproses informasi

Prosedur

Instruksi yang dibutuhkan oleh pengguna dalam memproses informasi

Proses

Perubahan atau transformasi input menjadi output

Prototyping

Salah satu pendekatan dalam pengembangan perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan

Rekayasa Perangkat Lunak

suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan

Sistem

Kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai tujuan tertentu

Sistem basis data

Kumpulan elemen-elemen seperti basis data, perangkat lunak, perangkat keras, dan manusia yang saling berinteraksi untuk mencapai tujuan yaitu pengorganisasian data.

Software

Lihat Perangkat Lunak

Software Engineering

Lihat Rekayasa Perangkat Lunak

Solusi

Bagian akhir atau output dari proses pemecahan masalah.

Strategi pemecahan masalah

Metode atau pendekatan yang digunakan seseorang ketika menghadapi masalah

Unknowns

Sesuatu yang harus kita dapatkan setelah proses pemecahan masalah dilakukan

Sinopsis

Buku ini diberi judul “Rekayasa Perangkat Lunak”, sama dengan salah satu program keahlian pada Sekolah Menengah Kejuruan (SMK). Meskipun demikian, sebenarnya isi dari buku ini tidak secara khusus membahas tentang Rekayasa Perangkat Lunak. Dari sisi pandang bidang Ilmu Komputer ada lima sub-bidang yang tercakup dalam dalam buku ini, yaitu sub-bidang Rekayasa Perangkat Lunak, Sistem Operasi dan Jaringan, Algoritma dan Struktur Data, Bahasa Pemrograman dan Basis Data. Hal ini disesuaikan dengan kurikulum tingkat SMK untuk Program Keahlian Rekayasa Perangkat Lunak.

Pokok bahasan tentang Rekayasa Perangkat Lunak secara umum membahas dasar-dasar pengertian Rekayasa Perangkat Lunak, masalah dan pemecahan masalah, dan metode-metode pengembangan perangkat lunak. Pembahasan tentang sub-bidang Sistem Operasi dan Jaringan berisi system computer, system operasi dan jaringan computer. Sub-bidang Algoritma dan Pemrograman menempati porsi terbesar dalam buku ini. Cakupan materi meliputi algoritma, Bahasa Pemrograman, Pemrograman Terstruktur, Pemrograman berbasis GUI, dan Pemrograman berbasis web. Satu bab difokuskan pada pembahasan tentang pencarian kesalahan (debugging) dan pengujian (testing). Sub-bidang terakhir yang menjadi bagian dari buku ini adalah Basis Data dengan cakupan tentang system basis data, pemodelan konseptual, basis data relasional, normalisasi, dan SQL.

Peta Kompetensi

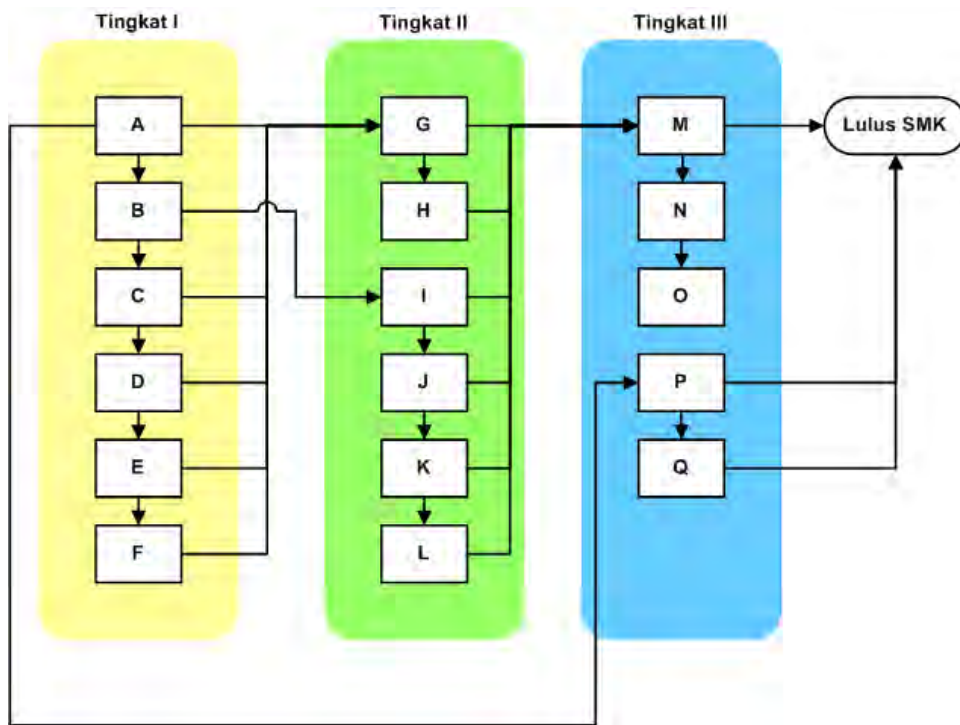
Secara umum, buku ini mengacu pada 9 kompetensi inti bagi SMK seperti yang disarankan dalam dokumen SKKNI untuk Bidang Programmer Komputer. Ke-9 unit kompetensi tersebut dapat dilihat pada Table A. Sesuai dengan perkembangan dalam dunia komputer dan untuk lebih memudahkan urutan, maka pada buku ini beberapa unit kompetensi ditambahkan. Secara lengkap, unit kompetensi yang ada pada buku ini dan bab yang relevan dapat dilihat pada Table A. Sedangkan diagram pencapaian kompetensi dapat dilihat pada Gambar B.

Tabel A. Unit-unit kompetensi inti Bidang Programmer Komputer yang disarankan untuk SMK.

No.	Nomor Unit	Judul Kompetensi
1	TIK.PR.02.001.01	Membuat algoritma pemrograman dasar
2	TIK.PR.02.002.01	Membuat algoritma pemrograman lanjutan
3	TIK.PR.02.005.01	Menulis program dasar
4	TIK.PR.02.008.01	Mengoperasikan bahasa pemrograman terstruktur / procedural
5	TIK.PR.02.012.01	Mengkompilasi dan menjalankan sebuah aplikasi dasar
6	TIK.PR.02.020.01	Mengoperasikan aplikasi basis data
7	TIK.PR.02.027.01	Menerapkan dasar-dasar pembuatan web statik dasar
8	TIK.PR.02.024.01	Membuat dokumen dengan HTML, sesuai spesifikasi
9	TIK.PR.02.028.01	Menerapkan dasar-dasar pembuatan web statik lanjut

Tabel B. Unit-unit kompetensi dan bab-bab yang terkait di dalam buku ini.

No.	Kode	Kompetensi	Bab Yang Terkait
1	A	Memahami pengertian dan prinsip-prinsip rekayasa perangkat lunak	Bab 1, 2 dan 3
2	B	Mengoperasikan system operasi dan jaringan komputer	Bab 4, 5 dan 6
3	C	Memahami dasar pemrograman dan bahasa pemrograman	Bab 7
4	D	Memahami tipe data, variable, konstanta dan operator	Bab 8
5	E	Membuat algoritma pemrograman dasar	Bab 9
6	F	Membuat algoritma pemrograman lanjutan	Bab 10
6	G	Menerapkan algoritma dalam bahasa pemrograman	Bab 11
7	H	Membangun aplikasi pemrograman berbasis GUI	Bab 12
9	I	Melakukan pencarian kesalahan dan pengujian program	Bab 13
10	J	Memahami prinsip dasar sistem basis data	Bab 14
11	K	Menerapkan prinsip pemodelan konseptual dan basis data relasional	Bab 15 dan 16
12	L	Mengelola basis data dengan perangkat Database Management System	Bab 17
13	M	Membuat query dan perintah-perintah SQL	Bab 18
14	N	Memahami prinsip kerja internet dan pemrograman web	Bab 19
15	O	Membuat web statis dengan HTML	Bab 20
16	P	Membangun aplikasi pemrograman berbasis web	Bab 21



Keterangan

A	Memahami pengertian dan prinsip-prinsip rekayasa perangkat lunak
B	Mengoperasikan system operasi dan jaringan komputer
C	Memahami dasar pemrograman dan bahasa pemrograman
D	Memahami tipe data, variable, konstanta dan operator
E	Membuat algoritma pemrograman dasar
F	Membuat algoritma pemrograman lanjutan
G	Menerapkan algoritma dalam bahasa pemrograman
H	Membangun aplikasi pemrograman berbasis GUI
I	Melakukan pencarian kesalahan dan pengujian program
J	Memahami prinsip dasar system basis data
K	Menerapkan prinsip pemodelan konseptual dan basis data relasional
L	Mengelola basis data dengan perangkat Database Management System
M	Membuat query dan perintah-perintah SQL
N	Memahami prinsip kerja internet dan pemrograman web
O	Membuat web statis dengan HTML
P	Membangun aplikasi pemrograman berbasis web

Gambar A. Diagram pencapaian kompetensi.

Pendahuluan

TUJUAN

Setelah anda mempelajari bab ini diharapkan anda akan mampu :

- Menjelaskan pengertian perangkat lunak, program, prosedur dan rekayasa perangkat lunak.
 - Memahami tujuan rekayasa perangkat lunak
 - Memahami ruang lingkup rekayasa perangkat lunak
 - Memahami posisi bidang rekayasa perangkat lunak pada disiplin ilmu komputer dan keterkaitannya dengan bidang ilmu lain
 - Mengetahui perkembangan ilmu rekayasa perangkat lunak
 - Mengetahui profesi dan sertifikasi dalam bidang rekayasa perangkat lunak.
-

1.1. PENGERTIAN

Istilah Rekayasa Perangkat Lunak secara umum disepakati sebagai terjemahan dari istilah *Software Engineering*. Istilah Software Engineering mulai dipopulerkan tahun 1968 pada Software Engineering Conference yang diselenggarakan oleh NATO. Sebagian orang mengartikan Rekayasa Perangkat Lunak hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak (*software*) dan program komputer.

Perangkat lunak:

Seluruh instruksi yang digunakan untuk memproses informasi.

Perangkat lunak dapat berupa program atau prosedur.

- **Program** adalah kumpulan instruksi komputer
- **Prosedur** adalah instruksi yang dibutuhkan oleh pengguna dalam memproses informasi

(O'Brien, 1999)

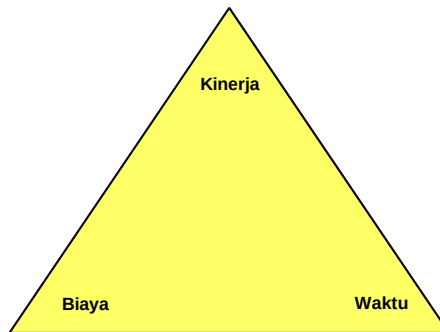
Rekayasa Perangkat Lunak:

Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.

Jelaslah bahwa Rekayasa Perangkat Lunak tidak hanya berhubungan dengan teknis pembuatan program komputer. Pernyataan “*semua aspek produksi*” pada definisi di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari Rekayasa Perangkat Lunak.

1.2. TUJUAN REKAYASA PERANGKAT LUNAK

Secara umum tujuan Rekayasa Perangkat Lunak tidak berbeda dengan bidang rekayasa yang lain. Ada tiga hal utama yang ingin dituju dari suatu bentuk rekayasa seperti terlihat pada gambar 1.1.



Gambar 1.1. Tujuan rekayasa perangkat lunak.

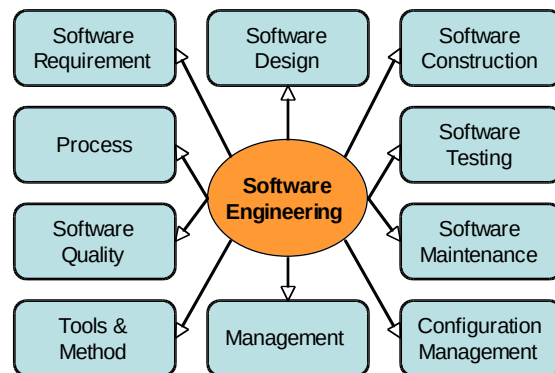
Dari gambar 1.1 dapat diartikan bahwa bidang rekayasa akan selalu berusaha menghasilkan output yang kinerjanya tinggi, biaya rendah dan waktu penyelesaian yang tepat.

Tujuan Rekayasa Perangkat Lunak

- a. *Memperoleh biaya produksi perangkat lunak yang rendah.*
- b. *Menghasilkan perangkat lunak yang kinerjanya tinggi, andal dan tepat waktu.*
- c. *Menghasilkan perangkat lunak yang portabilitasnya tinggi.*
- d. *Menghasilkan perangkat lunak yang biaya perawatannya rendah.*

1.3. RUANG LINGKUP

Sesuai definisi yang telah disampaikan sebelumnya, maka ruang lingkup Rekayasa Perangkat Lunak dapat digambarkan sebagai berikut.



Gambar 1.2. Ruang lingkup rekayasa perangkat lunak (Abran et.al., 2004).

- *Software requirements* berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
- *Software design* mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.
- *Software construction* berhubungan dengan detail pengembangan perangkat lunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.
- *Software testing* meliputi pengujian pada keseluruhan perilaku perangkat lunak.
- *Software maintenance* mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.

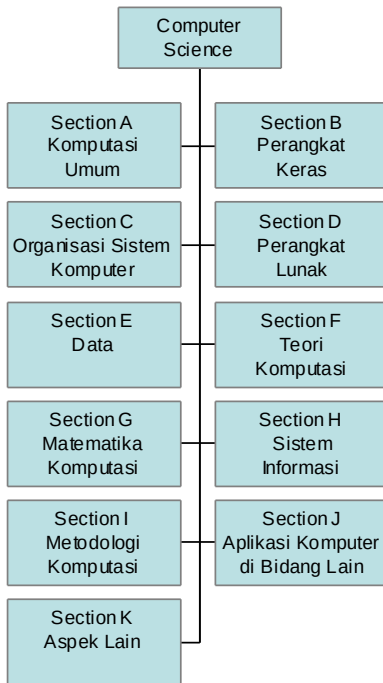
- *Software configuration management* berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
- *Software engineering management* berkaitan dengan pengelolaan dan pengukuran rekayasa perangkat lunak, termasuk perencanaan proyek perangkat lunak.
- *Software engineering tools and methods* mencakup kajian teoritis tentang alat bantu dan metode rekayasa perangkat lunak.
- *Software engineering process* berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses rekayasa perangkat lunak.
- *Software quality* menitikberatkan pada kualitas dan daur hidup perangkat lunak.

1.4. REKAYASA PERANGKAT LUNAK DAN DISIPLIN ILMU KOMPUTER

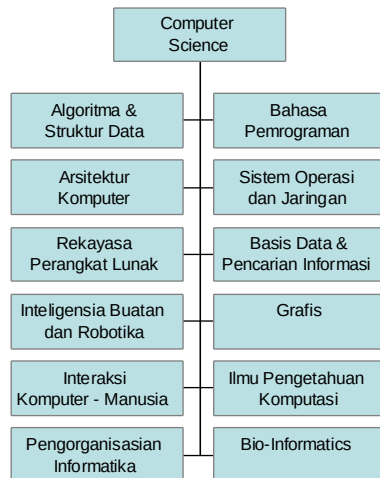
Disiplin ilmu komputer (*Computer Science*) lahir pada awal tahun 1940 an yang merupakan integrasi dari teori algoritma, logika matematika dan ditemukannya cara penyimpanan program secara elektronik pada komputer. Sejak itu ilmu komputer mengalami perkembangan yang terus menerus sehingga cakupannya menjadi semakin meluas.

Cakupan pengetahuan dalam ilmu komputer seringkali didiskripsikan sebagai suatu studi sistematis pada proses-proses algoritma yang menjelaskan dan mentransformasikan informasi (Denning, 2000). Termasuk di sini adalah teori, analisis, disain, efisiensi, penerapan dan aplikasinya.

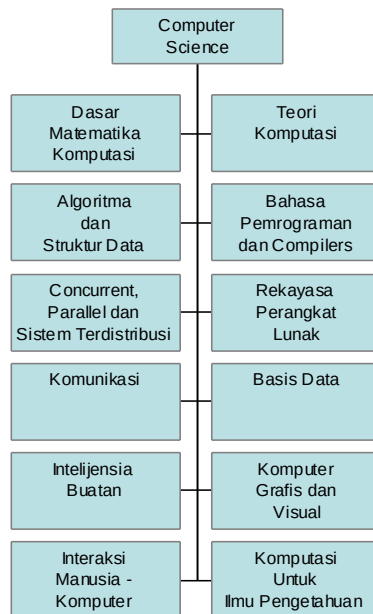
Ada beberapa model klasifikasi sub-bidang ilmu dalam disiplin ilmu komputer seperti terlihat pada Gambar 1.3, 1.4 dan 1.5.



Gambar 1.3. Klasifikasi disiplin ilmu komputer menurut ACM (1998).



Gambar 1.4. Klasifikasi disiplin ilmu komputer menurut Denning (2000).

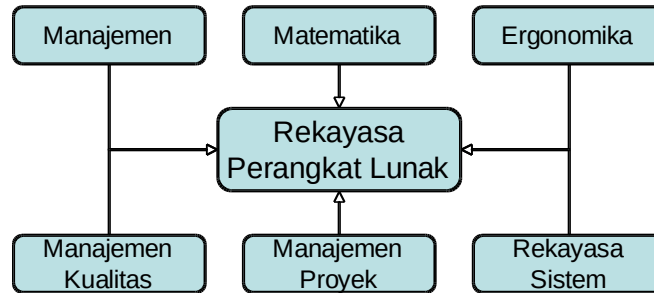


Gambar 1.5. Klasifikasi disiplin ilmu komputer menurut Wikipedia (2007).

Berdasarkan klasifikasi Denning (2000) dan Wikipedia (2007), Rekayasa Perangkat Lunak merupakan sub-bidang ilmu komputer yang setara dengan sub-bidang lainnya. Sedangkan menurut ACM (*Association for Computing Machinery*), Rekayasa Perangkat Lunak merupakan bagian dari Section D (Perangkat Lunak). Meskipun terlihat terpisah-pisah, namun dalam penerapannya, sub-bidang Rekayasa Perangkat Lunak selalu membutuhkan dukungan dari sub-bidang lain, terutama sub-bidang Algoritma dan Struktur Data, Bahasa Pemrograman, Basis Data, Sistem Operasi dan Jaringan, dan Sistem Informasi.

1.5. REKAYASA PERANGKAT LUNAK DAN DISIPLIN ILMU LAIN

Cakupan ruang lingkup yang cukup luas, membuat Rekayasa Perangkat Lunak sangat terkait dengan disiplin bidang ilmu lain. Tidak saja dengan sub-bidang dalam disiplin ilmu komputer namun dengan beberapa disiplin ilmu lain di luar ilmu komputer. Secara skematis keterkaitan Rekayasa Perangkat Lunak dengan ilmu lain dapat dilihat pada Gambar 1.6.



Gambar 1.6. Keterkaitan Rekayasa Perangkat Lunak dengan bidang ilmu lain.

- Bidang ilmu manajemen meliputi akuntansi, finansial, pemasaran, manajemen operasi, ekonomi, analisis kuantitatif, manajemen sumber daya manusia, kebijakan dan strategi bisnis.
- Bidang ilmu matematika meliputi aljabar linier, kalkulus, peluang, statistik, analisis numeric dan matematika diskrit.
- Bidang ilmu manajemen proyek meliputi semua hal yang berkaitan dengan proyek, seperti ruang lingkup proyek, anggaran, tenaga kerja, kualitas, manajemen resiko, dan penjadwalan proyek.
- Bidang ilmu manajemen kualitas meliputi pengembangan sistem kualitas, manajemen resiko dan keandalan, perbaikan kualitas, dan metode-metode kuantitatif.
- Bidang ilmu ergonomika menyangkut hubungan (interaksi) antara manusia dengan elemen-elemen lain dalam sistem komputer.
- Bidang ilmu rekayasa sistem meliputi teori sistem, analisis biaya-keuntungan, pemodelan, simulasi, proses dan operasi bisnis.

1.6. PERKEMBANGAN REKAYASA PERANGKAT LUNAK

Meskipun baru dicetuskan pada tahun 1968, namun Rekayasa Perangkat Lunak memiliki akar sejarah yang cukup panjang. Gambar 1.7 menyajikan intisari perkembangan Rekayasa Perangkat Lunak.

Meskipun telah mempunyai sejarah yang cukup panjang, namun dari sisi disiplin ilmu, Rekayasa Perangkat Lunak masih relatif muda dan akan terus berkembang. Arah perkembangan yang saat ini sedang dikembangkan antara lain meliputi : *Agile Software Development*, *Experimental Software Development*, *Model-Driven Software Development* dan *Software Product Lines*.

Tahun	Kejadian
1940an	Komputer pertama yang membolehkan pengguna menulis kode program langsung
1950an	Generasi awal interpreter dan bahasa macro Generasi pertama compiler
1960an	Generasi kedua compiler Komputer mainframe mulai dikomersialkan Pengembangan perangkat lunak pesanan Konsep Software Engineering mulai digunakan
1970an	Perangkat pengembang perangkat lunak Perangkat minicomputer komersial
1980an	Perangkat Komputer Personal (PC) komersial Peningkatan permintaan perangkat lunak
1990an	Pemrograman berorientasi obyek (OOP) Agile Process dan Extreme Programming Peningkatan drastis kapasitas memori Peningkatan penggunaan internet
2000an	Platform interpreter modern (Java, .Net, PHP, dll) Outsourcing

Gambar 1.7. Perkembangan Rekayasa Perangkat Lunak.

1.7. PROFESI DAN SERTIFIKASI

Profesi sebagai seorang *Software Engineer* mungkin masih terasa asing di telinga orang Indonesia. Sebagian besar orang Indonesia mungkin lebih familiar dengan sebutan Ahli Teknologi Informasi, Analis Sistem Informasi, Programmer, Operator atau sebutan profesi lainnya. Hal ini karena adanya kerancuan tentang istilah Rekayasa Perangkat Lunak seperti telah disebutkan di awal bab. Namun di Negara-negara yang maju dalam bidang teknologi informasi, sebutan *Software Engineer* telah mulai banyak digunakan.

Sertifikasi kompetensi dalam bidang Rekayasa Perangkat Lunak, saat ini masih menjadi perdebatan di kalangan ahli dan penyedia perangkat lunak. Sebagian besar sertifikasi dalam industri perangkat lunak biasanya sangat spesifik untuk perangkat lunak tertentu. Sebagai contoh, Perusahaan perangkat lunak seperti Redhat Linux Inc., Adobe Inc., Oracle, atau Microsoft, memberikan sertifikasi kemampuan pada seseorang yang menguasai perangkat lunak yang diproduksinya.

ACM (*Association for Computing Machinery*) pernah menyelenggarakan sertifikasi untuk program *Software Engineer* pada tahun

1980an, namun dihentikan karena kurangnya peminat. IEEE (*Institute of Electrical and Electronics Engineers*) telah mengeluarkan lebih dari 500 sertifikat profesi perangkat lunak. Di Canada, telah dikeluarkan sebuah sertifikat legal untuk Rekayasa Perangkat Lunak yang disebut sebagai ISP (*Information Systems Profesional*).

Saat ini, sertifikasi untuk Rekayasa Perangkat Lunak di Indonesia juga belum tersedia, namun telah disusun **Standar Kompetensi Kerja Nasional Indonesia untuk Bidang Programmer Komputer**. Meskipun belum memenuhi cakupan bidang Rekayasa Perangkat Lunak secara keseluruhan, namun paling tidak dapat digunakan sebagai pendekatan sertifikasi bidang Rekayasa Perangkat Lunak.

1.8. TENTANG BUKU INI

1.8.1. Isi

Buku ini diberi judul “Rekayasa Perangkat Lunak”, sama dengan salah satu program keahlian pada Sekolah Menengah Kejuruan (SMK). Meskipun demikian, sebenarnya isi dari buku ini tidak secara khusus membahas tentang Rekayasa Perangkat Lunak. Dari sisi pandang bidang Ilmu Komputer yang telah disebutkan sebelumnya, ada lima sub-bidang Ilmu Komputer yang tercakup dalam buku ini, yaitu sub-bidang Rekayasa Perangkat Lunak, Sistem Operasi dan Jaringan, Algoritma dan Struktur Data, Bahasa Pemrograman dan Basis Data. Hal ini disesuaikan dengan kurikulum tingkat SMK untuk Program Keahlian Rekayasa Perangkat Lunak.

Pokok bahasan tentang Rekayasa Perangkat Lunak secara umum disampaikan pada Bab 1, 2, dan 3. Bab 1, 2, dan 3 membahas dasar-dasar pengertian Rekayasa Perangkat Lunak, masalah dan pemecahan masalah, dan metode-metode pengembangan perangkat lunak.

Pembahasan tentang sub-bidang Sistem Operasi dan Jaringan disampaikan pada Bab 4, 5 dan 6. Bab 4 merupakan dasar dari bab 5 dan 6, karena pada bab ini berisi pembahasan tentang dasar-dasar pengetahuan tentang sistem komputer yang mencakup data dan informasi, perangkat keras dan perangkat lunak. Bab 5 secara khusus membahas sistem operasi yang meliputi fungsi, BIOS, jenis-jenis sistem operasi dan bagaimana bekerja dengan sistem operasi. Bab 6 berisi pembahasan tentang jaringan komputer dengan materi utama meliputi topologi dan jenis jaringan, protocol, konsep TCP/IP, dan perangkat keras.

Sub-bidang Algoritma dan Pemrograman menempati porsi terbesar dalam buku ini. Bab 9 dan 10 membahas detail tentang algoritma dan struktur data. Dimulai dengan pengenalan pada algoritma sederhana (Bab 9), kemudian dilanjutkan dengan algoritma yang lebih kompleks (Bab 10).

Sedangkan Bahasa Pemrograman dibahas pada Bab 7, 8, 11, 12, 19, 20, dan 21. Bab 7, dan 8, berisi dasar-dasar pengetahuan tentang Bahasa Pemrograman yang meliputi dasar-dasar pemrograman, pengenalan pada bahasa pemrograman dan pembahasan tentang tipe data, variable, konstanta, dan operator. Bab 11, 12, 19, 20 dan 21 secara khusus membahas beberapa teknik pemrograman yang populer digunakan dibidang teknologi informasi, yaitu penerapan algoritma dalam bahasa pemrograman (Bab 11), pemrograman berbasis GUI (Bab 12), dan pemrograman berbasis web (Bab 19, 20, dan 21). Satu bab yaitu Bab 13 difokuskan pada pembahasan tentang pencarian kesalahan (debugging) dan pengujian (testing).

Sub-bidang terakhir yang menjadi bagian dari buku ini adalah Basis Data. Pada buku ini, Basis Data disampaikan pada Bab 14, 15, 16, 17, dan 18. Bab 14 memberikan dasar pengertian tentang sistem basis data, kemudian dilanjutkan dengan bab 15 yang membahas tentang pemodelan konseptual. Bab 16, 17, dan 18 secara berturut-turut membahas tentang basis data relasional, penggunaan Database Management System (DBMS), dan SQL.

1.8.2. Bagaimana Menggunakan Buku Ini.

Buku ini secara khusus ditujukan kepada siswa dan guru SMK untuk program keahlian Rekayasa Perangkat Lunak. Namun demikian, buku ini juga terbuka bagi pembaca umum yang berminat dalam dunia Rekayasa Perangkat Lunak, Algoritma dan Pemrograman, Basis Data dan Internet. Bagi siswa, buku ini dapat dijadikan buku pegangan, karena ini buku ini menyediakan bahan-bahan pelajaran yang cukup lengkap untuk mata pelajaran selama tiga tahun di bangku sekolah. Beberapa bagian dari buku ini mungkin memerlukan buku-buku bantu lainnya untuk lebih memperkaya wawasan dan peningkatan kemampuan. Sedangkan bagi guru, buku ini dapat digunakan sebagai buku referensi untuk menyusun modul-modul ajar bagi anak didiknya.

Buku ini disusun sedemikian rupa agar siswa dapat belajar secara mandiri dan terdorong untuk mencoba secara langsung. Oleh karena itu dalam buku ini, akan banyak dijumpai ilustrasi baik yang berupa gambar, skema maupun *listing program*. Hal ini dimaksudkan agar siswa dapat dengan mudah memahami penjelasan ataupun penerapan suatu konsep tertentu. Bahkan pada bagian akhir bab-bab tertentu diakhiri dengan beberapa contoh aplikasi dari pokok bahasan pada bab tersebut.

Prinsip-Prinsip Pemecahan Masalah

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian masalah dan gejala.
 - Menjelaskan tipe-tipe masalah
 - Menerapkan prinsip-prinsip pemecahan masalah meliputi proses dan strategi pada masalah-masalah sederhana
 - Memahami apa yang disebut sebagai solusi
 - Menjelaskan pengertian sistem dan kerja sistem
 - Memahami prinsip pemecahan masalah dengan pendekatan sistem
-

Bab ini secara khusus akan membahas hubungan antara prinsip-prinsip pemecahan masalah dan rekayasa perangkat lunak. Hal ini karena secara konsep, rekayasa perangkat lunak memiliki kedekatan dengan prinsip-prinsip pemecahan masalah. Pemahaman tentang masalah, strategi dan proses pemecahan masalah, serta pendekatan sistem pada pemecahan masalah akan sangat membantu proses rekayasa perangkat lunak.

2.1. MASALAH

2.1.1. Pengertian

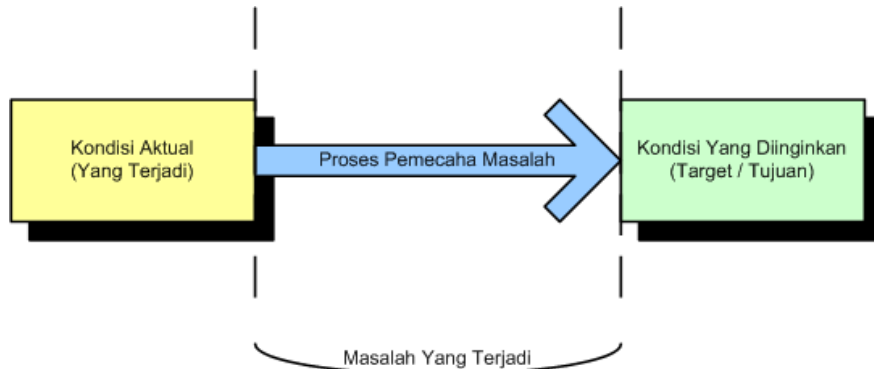
Masalah (problem)

Perbedaan antara situasi yang terjadi dan situasi yang diharapkan.

Atau

Perbedaan antara kondisi sekarang dengan target atau tujuan yang diinginkan.

Secara skematis, masalah dapat dilihat pada gambar 2.1. Dari gambar tersebut dapat dilihat proses pemecahan masalah merupakan proses untuk memperpendek jarak antara situasi aktual dengan situasi yang diharapkan.



Gambar 2.1. Kondisi aktual, target dan masalah.

Seringkali kita kesulitan membedakan antara gejala dan masalah.

Gejala :

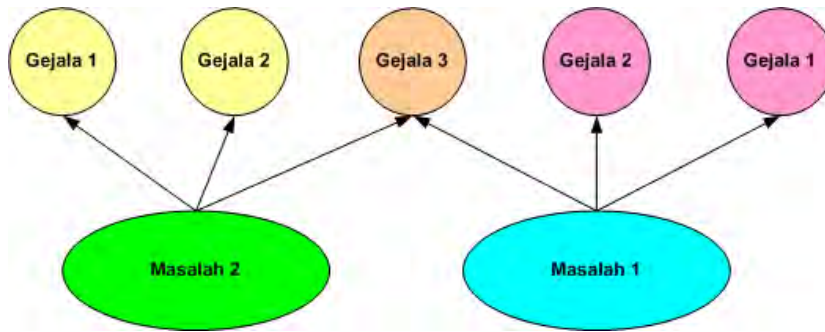
Signal atau tanda terjadinya suatu masalah,

Masalah :

Kondisi yang dapat menyebabkan timbulnya hasil yang tidak diinginkan.

Seperti halnya orang sakit flu, maka pusing, demam, batuk, dan pilek merupakan gejalanya, sedangkan masalahnya adalah sakit flu. Apabila kita mengobati hanya dengan obat sakit kepala, maka penyakit flu tidak akan sembuh.

Lebih mudah menentukan gejala dari pada masalah, karena gejala selalu tampak, bahkan seringkali dalam bentuk yang terukur (kuantitatif). Sebagai contoh, penurunan nilai rata-rata semester terakhir pada rapor anda merupakan gejala yang tampak dan mudah diukur karena kita tinggal membandingkan nilai rata-rata semester sebelumnya dengan nilai rata-rata semester terakhir. Namun untuk mengetahui masalah apa yang menyebabkan terjadinya penurunan penjualan, perlu kajian yang lebih mendalam. Satu buah masalah mungkin mempunyai satu gejala, tapi juga mungkin lebih dari satu gejala. Kesulitan utama dalam menentukan masalah yang sebenarnya adalah karena satu atau lebih gejala mungkin dimiliki oleh beberapa masalah secara bersamaan (Gambar 2.2).



Gambar 2.2. Gejala dan masalah.

Problem yang didefinisikan dengan baik, menurut Deek et al (2005), minimal akan memuat *goals*, *givens*, dan *unknowns*.

Goals:

Sesuatu yang ingin kita tuju atau selesaikan.

Givens :

Representasi dari fakta-fakta spesifik yang harus diidentifikasi sebelum proses pemecah masalah dilakukan.

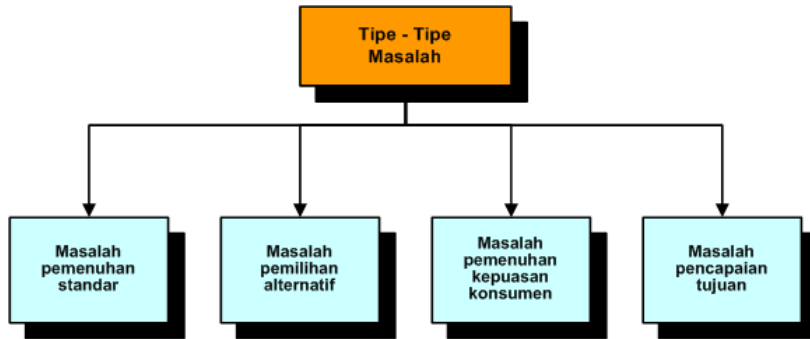
Unknowns :

Sesuatu yang harus kita dapatkan setelah proses pemecahan masalah dilakukan.

Sebagai contoh, pada masalah pengurutan bilangan, *Goals* – nya adalah mengatur letak bilangan sesuai urutan yang diinginkan, *Givens* – nya adalah bilangan-bilangan yang letaknya acak tak berurutan, sedangkan *unknowns* – nya adalah urutan bilangan. Selain tiga hal tersebut, biasanya juga ditambahkan *constraint*, yaitu batasan-batasan dari masalah. Pada kasus pengurutan bilangan, *constraint* – nya adalah masalah ini hanya terbatas pada bilangan-bilangan yang telah disediakan tidak pada bilangan-bilangan lain.

2.1.2. Tipe-Tipe Masalah

Ada beberapa tipe masalah dapat dikenali. Klasifikasi masalah dapat dilihat pada gambar 2.3. Klasifikasi ini didasarkan pada pendekatan dalam pendefinisian masalah.



Gambar 2.3. Tipe-tipe masalah (Deek et al, 2005).

- Masalah pemenuhan standar

Tipe masalah dalam kelompok ini adalah masalah-masalah yang berhubungan dengan pencapaian standar yang telah ditentukan dalam sebuah organisasi. Biasanya tujuan seperti ini berlaku dalam jangka yang relative panjang.

- Masalah pemilihan alternative

Masalah dalam kelompok ini berhubungan dengan bagaimana memilih solusi terbaik dari berbagai alternative berdasarkan kriteria-kriteria tertentu. Permasalahan ini seringkali kita jumpai dalam kehidupan sehari-hari, seperti bagaimana memilih sekolah yang tepat, memilih lokasi tempat tinggal, memilih bidang pekerjaan. Masing-masing alternatif dan kriteria memiliki bobot yang telah disepakati.

- Masalah pemenuhan kepuasan konsumen

Pada organisasi-organisasi yang bersifat profit (mencari keuntungan), masalah-masalah pada kelompok ini merupakan tipe yang seringkali muncul. Konsumen memiliki berbagai macam keinginan yang satu sama lain berbeda. Memenuhi seluruh keinginan konsumen sangat tidak mungkin dan sangat memberatkan sebuah organisasi. Oleh karena itu perlu dicari pemecahan yang sama-sama menguntungkan, baik bagi konsumen maupun organisasi tersebut.

- Masalah pencapaian tujuan

Tipe ini mirip dengan tipe pertama (masalah pemenuhan standar). Yang berbeda adalah, pada tipe ini tujuan yang ingin dicapai dapat berubah-ubah dan bersifat jangka pendek.

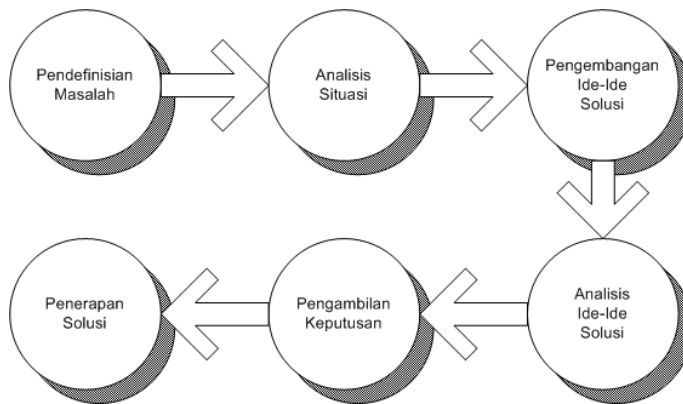
2.2. PEMECAHAN MASALAH

Pemecahan masalah :

sebuah proses dimana suatu situasi dianalisa kemudian solusi-solusi dibuat bila ditemukan ada masalah, dengan cara pendefinisian, pengurangan atau penghilangan, atau pencegahan masalah.

2.2.1. Proses

Ada banyak urutan proses pemecahan masalah yang diajukan oleh para ahli, salah satunya seperti terlihat pada Gambar 2.4.



Gambar 2.4. Proses pemecahan masalah (diadopsi dari Deek et al, 2005)

Pada gambar 2.4 terlihat serangkaian tahapan proses yang berbeda yang dapat digunakan dalam berbagai tingkatan, tergantung dari karakteristik problemnya. Masalah yang berbeda membutuhkan penggunaan elemen yang berbeda, bahkan mungkin urutan yang berbeda. Tahapan kritis dari proses pemecahan masalah adalah **Pendefinisian Masalah**. Apabila masalah tidak cukup jelas didefinisikan maka tahapan-tahapan berikut sulit untuk dijalankan. Bahkan apabila dipaksakan, kemungkinan besar solusi yang tepat tidak akan diperoleh.

Secara umum proses pemecahan masalah dapat dilakukan dengan empat tahapan utama yaitu :

- Memahami dan mendefinisikan masalah

Bagian ini merupakan bagian yang sangat penting karena menjadi awal dari seluruh proses pemecahan masalah. Pada bagian ini harus sudah teridentifikasi *goals*, *givens*, *unknowns*, dan *constraint* dari suatu masalah.

Tujuan pada bagian ini adalah mengorganisasi masalah dengan baik dan menghilangkan bagian-bagian yang dirasa kurang penting.

- Membuat rencana untuk pemecahan masalah

Pada bagian ini ada dua aktifitas penting yaitu :

- a) identifikasi berbagai alternatif solusi yang mungkin
- b) membuat rencana pemecahan masalah

Solusi pada suatu masalah biasanya tidak hanya satu tapi mungkin bisa beberapa macam solusi. Sebagai ilustrasi, apabila kita berada di kota Surabaya dan ingin pergi ke Jakarta, maka banyak alternatif solusi yang mungkin bisa dilakukan, misalnya kita bisa menempuh dengan angkutan darat, laut atau udara. Dengan angkutan darat kita bisa menggunakan kereta api, bus atau angkutan yang lain. Jalurnya pun kita bisa lewat jalur utara, tengah atau selatan. Jadi banyak sekali alternatif solusi yang bisa kita kembangkan. Masing-masing mempunyai karakteristik sendiri-sendiri. Dari sekian banyak alternative solusi ini kita harus memilih satu alternatif solusi yang berdasarkan kriteria tertentu merupakan alternatif solusi yang paling potensial dapat menyelesaikan permasalahan. Setelah terpilih, maka kita dapat membuat rencana kasar (*outline*) penyelesaian masalah dan membagi masalah dalam bagian-bagian yang lebih kecil. Rencana kasar (*outline*) penyelesaian masalah hanya berisi tahapan-tahapan utama penyelesaian masalah.

- Merancang dan menerapkan rencana untuk memperoleh solusi

Pada bagian ini rencana kasar penyelesaian masalah diperbaiki dan diperjelas dengan pembagian dan urutan rinci yang harus ditempuh dalam penyelesaian masalah.

- Memeriksa dan menyampaikan hasil dari pemecahan masalah

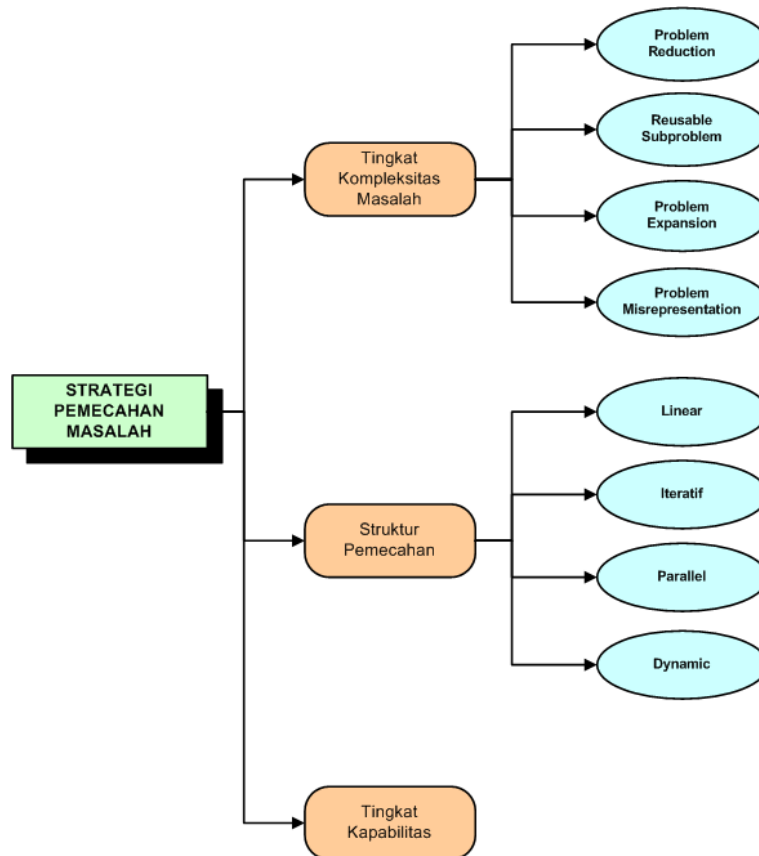
Bagian ini bertujuan untuk memeriksa apakah akurasi (ketepatan) hasil dari alternatif solusi yang dipilih telah memenuhi tujuan yang diinginkan. Selain itu juga untuk melihat bagaimana efektifitas dan kinerja dari alternatif solusi yang dipilih.

2.2.2.Strategi

Strategi pemecahan / Skema Pemecahan Masalah :

Metode atau pendekatan yang digunakan seseorang ketika menghadapi masalah.

Ada banyak strategi pemecahan masalah yang coba diterapkan oleh para ahli. Masing-masing memiliki keunggulan dan kekurangan, sehingga kadang-kadang untuk memecahkan suatu masalah tidak cukup dengan satu metode. Hal ini ditujukan untuk mendapat solusi yang optimal. Deek et al (2005) telah membuat klasifikasi strategi pemecahan masalah seperti terlihat pada Gambar 2.5.



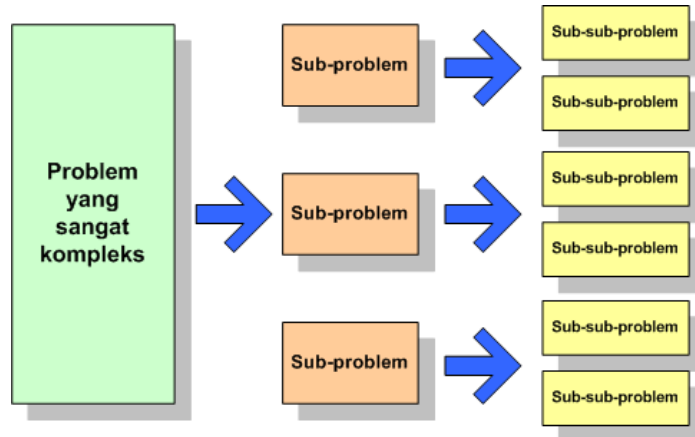
Gambar 2.5. Klasifikasi strategi pemecahan masalah.

- *Strategi pemecahan masalah berdasarkan tingkat kompleksitas masalah*

a) Problem reduction

Strategi ini sering juga disebut sebagai dekomposisi (*decomposition*). Pada strategi ini masalah yang kompleks dipecah-pecah menjadi bagian-bagian yang lebih kecil dan lebih sederhana (Gambar 2.6). Satu per satu bagian-bagian kecil ini kemudian dievaluasi dan dicarikan pemecahan masalahnya. Strategi ini banyak sekali digunakan terutama karena sangat mempermudah proses pemecahan masalah.

Beberapa teknik pemrograman perangkat lunak menggunakan pendekatan ini.



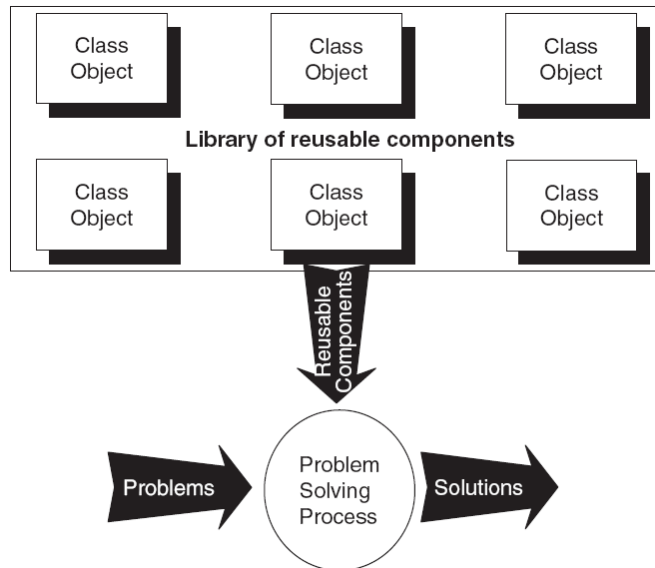
Gambar 2.6. Strategi dekomposisi masalah.

b) Reusable problem

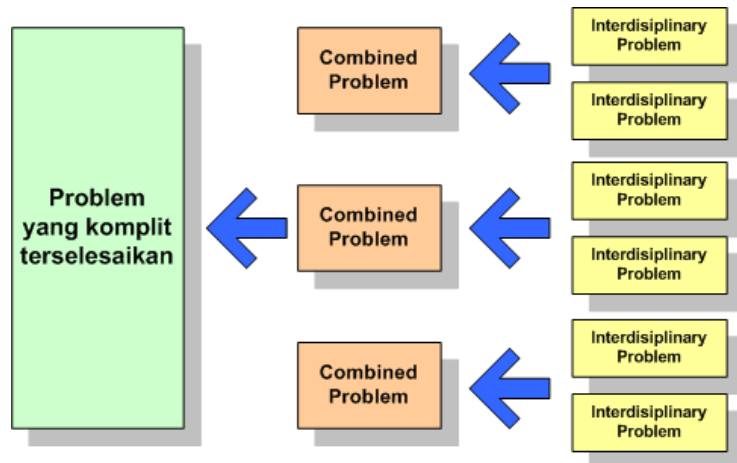
Kadang kala sebuah masalah memiliki kesamaan pada bagian-bagian tertentu dengan masalah lain. Pada kasus seperti ini apabila strategi *problem reduction* diterapkan, maka akan sangat memakan biaya dan waktu. Strategi yang dapat digunakan adalah dengan menggunakan solusi-solusi yang sudah pernah digunakan untuk masalah lain dan menerapkan dengan atau tanpa perubahan pada masalah yang akan kita selesaikan. Solusi-solusi yang sudah pernah dibuat ini dalam dunia pemrograman biasa disebut sebagai *library* (Gambar 2.7).

c) Problem expansion

Strategi *problem reduction* menganut prinsip bahwa masalah yang akan diselesaikan sudah dikenali dengan baik dan hanya dapat diselesaikan dengan dekomposisi, tapi pada masalah yang sumbernya dari berbagai macam hal (*interdisciplinary*) maka strategi ini tidak dapat digunakan. Strategi *problem expansion* dapat digunakan untuk mengatasi masalah ini. Strategi ini dilakukan dengan cara mengkombinasikan masalah-masalah kecil menjadi masalah yang agak besar agar kejelasan masalah semakin tampak. Setelah dirasa cukup jelas maka pemecahan masalah baru dilakukan (Gambar 2.8).



Gambar 2.7. Pemecahan masalah dengan *library*.



Gambar 2.8. Strategi *problem expansion*.

d) Problem misrepresentation

Strategi ini dilakukan untuk mengurangi ketidaktepatan (*misrepresentation*) dalam mendefinisikan masalah. Beberapa ketidaktepatan yang sering muncul adalah:

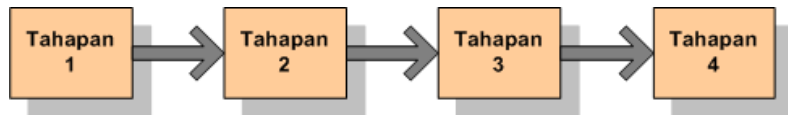
- 1) Penggunaan istilah/kata/kalimat yang tidak tepat.
- 2) Ruang lingkup masalah yang tidak jelas.

- 3) Penempatan hubungan antara masalah satu dengan masalah lain yang kurang tepat.
- 4) Penggunaan standar dalam notasi atau diagram pemecahan masalah yang tidak sesuai.

- *Strategi pemecahan masalah berdasarkan struktur pemecahan masalah*

- a) Linear

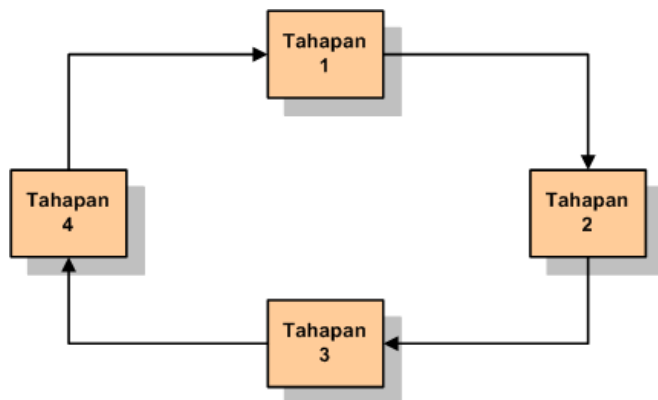
Strategi *linear* dilakukan dengan mengerjakan tahapan-tahapan proses pemecahan masalah secara berurutan (Gambar 2.9). Tahapan ke 3 misalnya sangat bergantung pada tahap ke 2, sedangkan tahapan ke 3 sendiri menjadi dasar tahapan ke 4.



Gambar 2.9. Strategi pemecahan masalah linear.

- b) Iterative

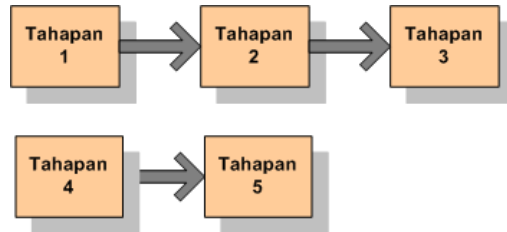
Strategi *iterative* digambarkan dengan bentuk lingkaran (Gambar 2.10). Pada gambar tersebut sebenarnya pola yang dipakai tetap berurutan, namun setelah tahapan terakhir, proses berulang kembali ke tahapan 1 untuk memperbaiki kekurangan yang mungkin ada. Proses ini berulang terus sampai suatu masalah dirasa sudah terselesaikan.



Gambar 2.10. Strategi pemecahan masalah *iterative*.

c) Parallel

Berbeda dengan strategi *linear*, strategi *parallel* mengerjakan tahapan-tahapan proses secara bersama-sama dalam bentuk *parallel* (Gambar 2.11). penggabungan strategi *linear* dan *parallel* sering kali digunakan. Pada bagian-bagian tertentu, tahapan harus dikerjakan secara *linear* dan pada bagian lain yang memungkinkan, dapat diterapkan strategi *parallel*.



Gambar 2.11. Strategi pemecahan masalah *parallel*.

d) Dynamic

Strategi ini tidak mengikuti pola-pola tertentu seperti yang telah disebutkan. Pola dapat bersifat acak tergantung pada interaksi dan hubungan antar tahapan.

- Strategi pemecahan masalah berdasarkan tingkat kapabilitas

Strategi ini didasarkan pada ketersediaan sumberdaya pada sebuah organisasi. Salah satu yang menjadi pertimbangan penting dalam ketersediaan dan kemampuan teknologi yang dimiliki. Semakin baik dan maju teknologi maka akan berpengaruh pada bagaimana proses pemecahan masalah dilakukan.

2.3. SOLUSI MASALAH

Solusi :

Bagian akhir atau output dari proses pemecahan masalah.

Idealnya ketika solusi telah diperoleh, tujuan pemecahan masalah harusnya telah didapatkan dan problem dapat dihilangkan atau dikurangi. Namun dalam prakteknya, ketika solusi diperoleh berarti tahapan baru dalam pemecahan masalah akan muncul, yaitu tahapan evaluasi, pemeriksaan, perbaikan dan seterusnya. Sehingga sebenarnya tidak ada masalah yang

benar-benar bisa dihilangkan. Sebagai contoh ketika kita merancang perangkat lunak untuk menyelesaikan suatu permasalahan dan perangkat lunak kita telah jadi, maka kita akan dihadapkan pada proses-proses sehubungan dengan perangkat lunak tersebut, seperti evaluasi kinerja perangkat lunak, perbaikan pada bagian-bagian tertentu, perubahan tampilan dan sebagainya.

2.4. PENDEKATAN SISTEM DALAM PEMECAHAN MASALAH

Rekayasa perangkat lunak secara alamiah akan selalu menggunakan pendekatan sistem. Hal ini karena perangkat lunak sendiri adalah sebuah sistem.

2.4.1. Sistem

Sistem :

Kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai tujuan tertentu yang tidak diinginkan.

Jadi pada setiap sistem harus terdapat **elemen**, **interaksi** dan **tujuan** (Gambar 2.12). Suatu sistem biasanya mempunyai tiga komponen dasar yang saling berinteraksi, yaitu input, proses dan output.

Input :

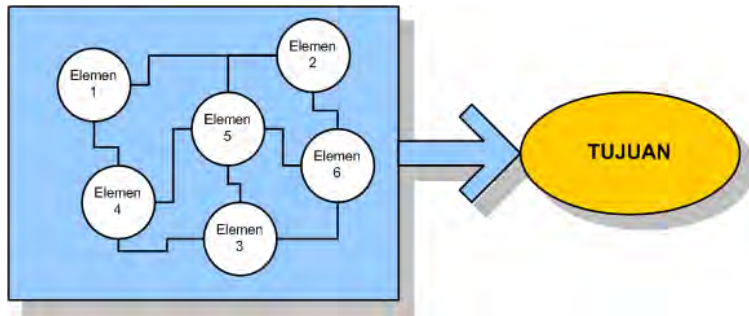
Elemen-elemen yang masuk ke dalam sistem.

Proses :

Perubahan input menjadi output.

Output :

Perpindahan elemen-elemen yang dihasilkan dari proses perubahan ke tujuan yang diinginkan.



Gambar 2.12. Sistem.

Selain tiga komponen ini, biasanya juga ditambahkan komponen *feedback* dan *control*.

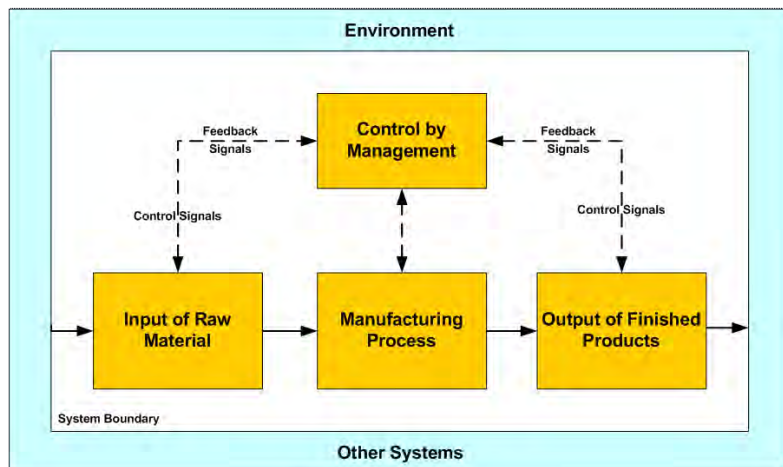
Feedback :

data tentang kinerja sistem sedangkan

Control :

aktivitas monitoring dan evaluasi terhadap feedback untuk menentukan apakah sistem telah bekerja dengan baik atau tidak

Gambar 2.13 menunjukkan bagaimana interaksi dari kelima komponen ini.



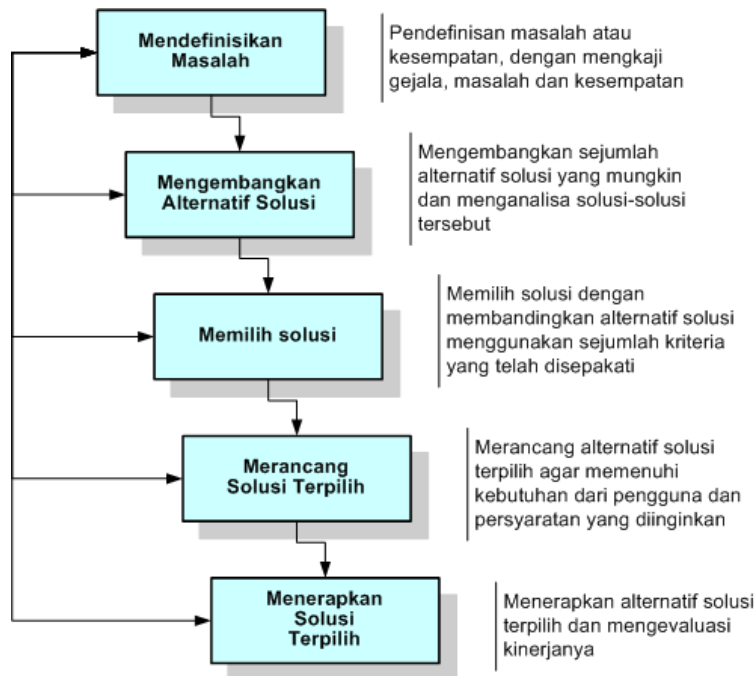
Gambar 2.13. Interaksi komponen-komponen dalam sistem (O'Brien, 1999).

2.4.2. Pendekatan sistem

Pendekatan sistem :

Pendekatan yang berorientasi pada sistem untuk mendefinisikan masalah dan kesempatan dan mengembangkan solusinya.

Tahapan-tahapan pendekatan sistem dalam penyelesaian masalah dapat dilihat pada Gambar 2.14. Kalau dicermati, gambar ini hampir serupa dengan Gambar 2.4 karena memang secara prinsip, pemecahan masalah akan selalu menempuh tahapan-tahapan seperti pada kedua gambar tersebut. Yang berbeda adalah orientasinya. Pada pendekatan sistem suatu masalah dapat dirunut dari interaksi antara elemen-elemen di dalam sistem atau interaksi antara elemen di dalam sistem dengan lingkungannya. Masalah yang terjadi akan menyebabkan kerja sistem menjadi terhambat atau gagal dalam mencapai tujuan. Pada pendekatan sistem, untuk setiap masalah apapun harus dicari mana sistemnya, mana sub-sistemnya, apa saja elemen sistemnya, dan bagaimana interaksinya.



Gambar 2.14. Tahapan-tahapan pendekatan sistem dalam penyelesaian masalah (O'Brien, 1999).

Metode Rekayasa Perangkat Lunak

TUJUAN

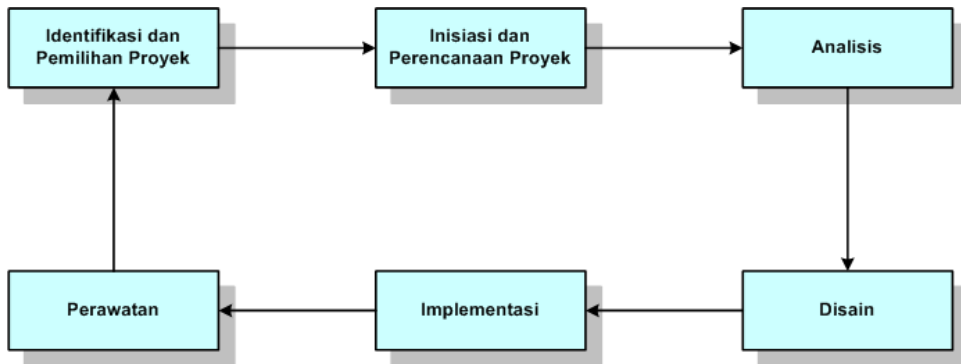
Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Memahami karakteristik umum model proses dalam rekayasa perangkat lunak.
 - Menyebutkan beberapa model rekayasa perangkat lunak .
 - Mengetahui prinsip-prinsip dari metode *waterfall*, *prototyping*, dan *unified process*.
 - Memahami tahapan-tahapan dalam rekayasa perangkat lunak.
-

Rekayasa perangkat lunak (software engineering) merupakan reaksi yang muncul dari kompleksnya permasalahan dalam pengembangan perangkat lunak. Rekayasa perangkat lunak yang sukses tidak hanya membutuhkan kemampuan komputasi seperti algoritma, pemrograman, dan basis data yang kuat, namun juga perlu penentuan tujuan yang baik, identifikasi alternative-alternatif yang berpotensi, metode pengembangan, urutan aktifitas, identifikasi kebutuhan sumberdaya, dan faktor-faktor lain. Hal-hal seperti ini terkait dengan apa yang disebut dengan metode rekayasa perangkat lunak. Bab ini secara ringkas akan membahas model-model rekayasa perangkat lunak dan tahapan dalam rekayasa perangkat lunak.

3.1 MODEL PROSES REKAYASA PERANGKAT LUNAK

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut **System Development Life Cycle (SDLC)** seperti terlihat pada Gambar 3.1.



Gambar 3.1. System Development Life Cycle (SDLC).

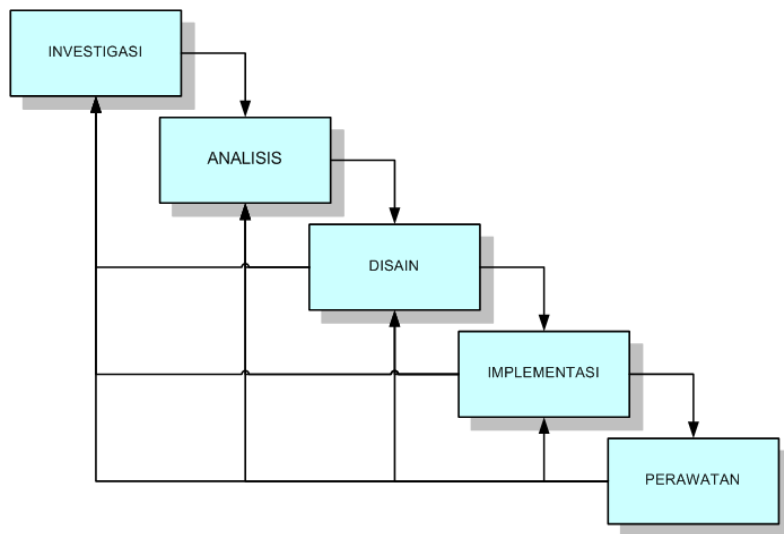
Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri. Namun secara umum ada persamaan dari model-model ini, yaitu:

- *Kebutuhan terhadap definisi masalah yang jelas.* Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. Oleh karena itu pemahaman masalah seperti dijelaskan pada Bab 2, merupakan bagian penting dari model pengembangan perangkat lunak.
- *Tahapan-tahapan pengembangan yang terorganisir.* Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum *analysis – design – coding – testing – maintenance*.
- *Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan.* Stakeholder dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.
- *Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak.* Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.
- *Outcome dari proses pengembangan perangkat lunak harus bernilai ekonomis.* Nilai dari sebuah perangkat lunak sebenarnya agak susah di-rupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat berupa penurunan biaya operasi, efisiensi penggunaan sumberdaya, peningkatan keuntungan organisasi, peningkatan “image” organisasi dan lain-lain.

Ada banyak model pengembangan perangkat lunak, antara lain *The Waterfall Model*, *Joint Application Development (JAD)*, *Information Engineering (IE)*, *Rapid Application Development (RAD)* termasuk di dalamnya *Prototyping*, *Unified Process (UP)*, *Structural Analysis and Design (SAD)* dan *Framework for the Application of System thinking (FAST)*. Pada buku ini akan dibahas tiga model pengembangan yaitu *The Waterfall Model*, *Prototyping*, dan *Unified Process (UP)*.

3.1.1 The waterfall model

Model siklus hidup (*life cycle model*) adalah model utama dan dasar dari banyak model. Dua model yang cukup dikenal dalam dunia rekayasa perangkat lunak adalah *The Waterfall Model* dan *Incremental and Iteratif Model*. Ada 5 tahapan utama dalam *The Waterfall Model* seperti terlihat pada Gambar 3.2. Disebut waterfall (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat.



Gambar 3.2. The Waterfall Model

- Tahap investigasi dilakukan untuk menentukan apakah ada masalah pada organisasi atau adakah peluang suatu sistem informasi dikembangkan. Pada tahapan ini studi kelayakan perlu dilakukan untuk menentukan apakah sistem informasi yang akan dikembangkan merupakan solusi yang layak
- Tahap analisis bertujuan untuk mengidentifikasi kebutuhan pengguna sistem dan organisasi serta menganalisa sistem yang ada (sebelum diterapkan sistem informasi yang baru).

- Tahap disain bertujuan menentukan spesifikasi detil dari komponen-komponen sistem (manusia, *hardware*, *software*, *network* dan data) dan produk-produk informasi yang sesuai dengan hasil tahap analisis.
- Tahap implementasi merupakan tahapan untuk mendapatkan atau mengembangkan *hardware* dan *software* (pengkodean program), melakukan pengujian, pelatihan dan konversi ke sistem baru.
- Tahapan perawatan (maintenance) dilakukan ketika sistem informasi sudah dioperasikan. Pada tahapan ini dilakukan monitoring proses, evaluasi dan perubahan (perbaikan) bila diperlukan.

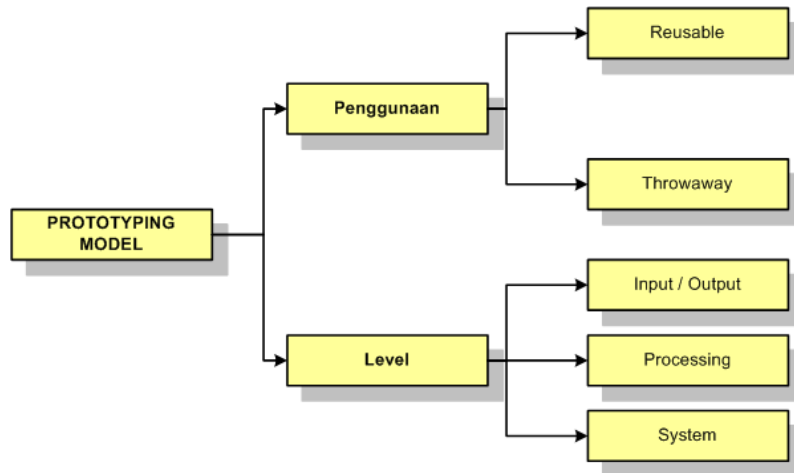
3.1.2 Prototyping model

Prototyping :

Salah satu pendekatan dalam pengembangan perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan.

(Howard, 1997)

Prototyping model dapat diklasifikasikan menjadi beberapa tipe seperti terlihat pada gambar 3.3.



Gambar 3.3. Klasifikasi prototyping model (Harris, 2003)

Reusable prototype :

Prototype yang akan ditransformasikan menjadi produk final.

Throwaway prototype :

Prototype yang akan dibuang begitu selesai menjalankan maksudnya.

Input/output prototype :

Prototype yang terbatas pada antar muka pengguna (user interface).

Processing prototype :

Prototype yang meliputi perawatan file dasar dan proses-proses transaksi.

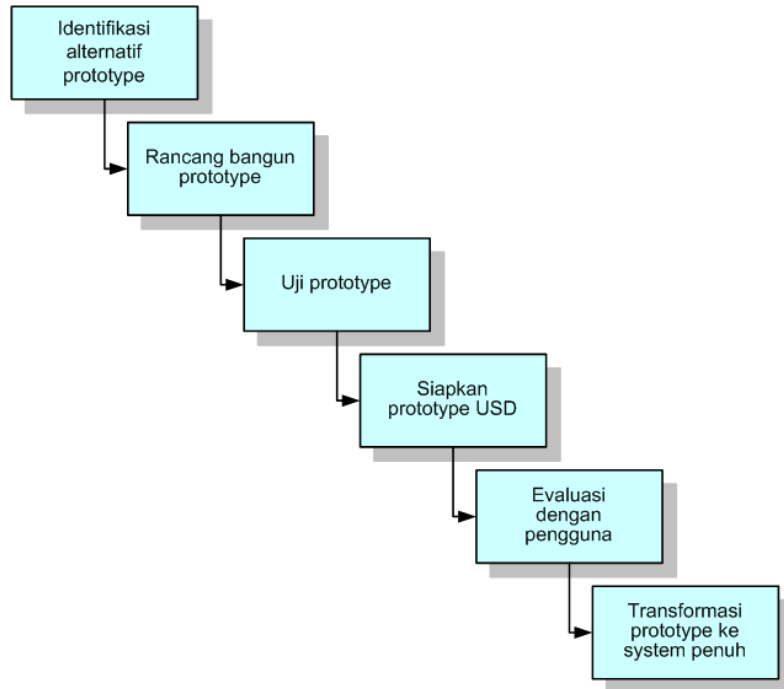
System prototype :

Prototype yang berupa model lengkap dari perangkat lunak.

Tahap-tahap dalam prototyping boleh dikata merupakan tahap-tahap yang dipercepat. Strategi utama dalam prototyping adalah kerjakan yang mudah terlebih dahulu dan sampaikan hasil kepada pengguna sesegera mungkin. Harris (2003) membagi prototyping dalam enam tahapan seperti terlihat pada gambar 3.4. Tahapan-tahapan secara ringkas dapat dijelaskan sebagai berikut:

- *Identifikasi kandidat prototyping.* Kandidat dalam kasus ini meliputi *user interface* (menu, dialog, input dan output), file-file transaksi utama, dan fungsi-fungsi pemrosesan sederhana.
- *Rancang bangun prototype dengan bantuan software* seperti *word processor, spreadsheet, database*, pengolah grafik, dan software CASE (*Computer-Aided System Engineering*).
- *Uji prototype* untuk memastikan prototype dapat dengan mudah dijalankan untuk tujuan demonstrasi.
- *Siapkan prototype USD (User's System Diagram)* untuk mengidentifikasi bagian-bagian dari perangkat lunak yang di-prototype-kan.
- *Evaluasi dengan pengguna* untuk mengevaluasi prototype dan melakukan perubahan jika diperlukan.
- *Transformasikan prototype menjadi perangkat lunak yang beroperasi penuh* dengan melakukan penghilangan kode-kode yang tidak

dibutuhkan, penambahan program-program yang memang dibutuhkan dan perbaikan dan pengujian perangkat lunak secara berulang.



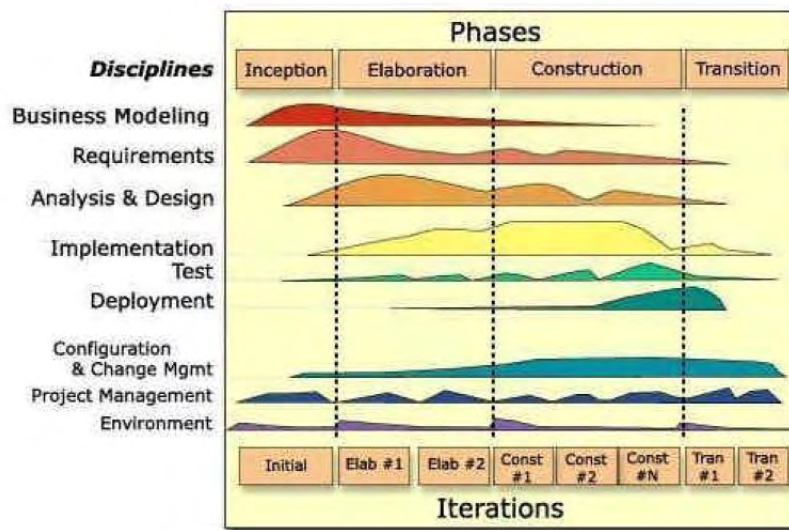
Gambar 3.4. Tahapan-tahapan prototyping model (Harris, 2003)

3.1.3 Unified Process dan Unified Modeling Language

Unified Process (UP) atau kadang disebut sebagai *Unified Software Development Process (USDP)* adalah kerangka proses pengembangan yang bersifat *use-case-driven*, berpusat pada arsitektur perangkat lunak, interatif dan tumbuh-kembang (Alhir, 2005). Kerangka pengembangan ini termasuk baru dalam metodologi pengembangan perangkat lunak. UP dapat diaplikasikan pada berbagai skala proyek, mulai dari skala kecil sampai dengan skala besar yang memiliki tingkat kompleksitas manajerial dan teknikal bervariasi.

Daur hidup UP secara umum akan tampak seperti pada bagan di Gambar 3.5. Bagan ini biasa disebut sebagai “*hump chart*”. Pada bagan ini terlihat ada empat tahap pengembangan yaitu *inception*, *elaboration*, *construction* dan *transition*. Selain itu tampak pula sejumlah aktivitas (*disciplines*) yang harus dilakukan sepanjang pengembangan perangkat lunak, yaitu, *business modeling*, *requirements*, *analysis and design*, *implementation*, *test*. Tahap dan aktivitas tersebut akan dilakukan secara iteratif (Ambler, 2005). Penjelasan singkat untuk empat tahapan dalam UP adalah sebagai berikut:

- *Inception*. Tahapan ini merupakan tahapan paling awal dimana aktivitas evaluasi terhadap sebuah proyek perangkat lunak dilakukan. Tujuannya adalah untuk mendapatkan kesepakatan dari stakeholder sehubungan dengan tujuan dan dana proyek.
- *Elaboration*. Tujuan dari tahap ini adalah untuk mendapatkan gambaran umum kebutuhan, persyaratan dan fungsi-fungsi utama perangkat lunak. Hal ini penting untuk mengetahui secara lebih baik resiko-resiko proyek, baik meliputi resiko arsitektur perangkat lunak, perencanaan, maupun implementasi. Pada tahap ini telah dimulai rancang bangun perangkat lunak secara iterative melalui aktivitas-aktivitas seperti *business modeling*, *requirements*, *analysis* dan *design* meskipun baru pada tahap awal.



Gambar 3.5. RUP Life Cycle (Ambler, 2005).

- *Construction*. Tujuan dari tahapan ini adalah membangun perangkat lunak sampai dengan saat perangkat lunak tersebut siap digunakan. Titik berat tahapan ini adalah pada penentuan tingkat prioritas kebutuhan / persyaratan, melengkapi spesifikasinya, analisis lebih dalam, disain solusi yang memenuhi kebutuhan dan persyaratan, pengkodean dan pengujian perangkat lunak. Jika dimungkinkan versi awal dari perangkat lunak diuji cobakan untuk mendapatkan masukan dari pengguna.
- *Transition*. Tahap ini difokuskan pada bagaimana menyampaikan perangkat lunak yang sudah jadi pada pengguna. Perangkat lunak akan secara resmi diuji oleh baik oleh penguji (tester) yang kompeten

maupun oleh pengguna. Beberapa aktivitas seperti pemindahan pusat data dan pelatihan pengguna dan staf pendukung harus dilakukan pada tahap ini.

Dalam pengembangan perangkat lunak dengan menggunakan UP, maka tidak lepas dari penggunaan notasi-notasi yang biasa disebut sebagai **UML (*Unified Modeling Language*)**. Meskipun UP mensyaratkan penggunaan UML, namun UML sendiri dapat digunakan pada berbagai metodologi yang lain bahkan dapat digunakan pada bidang selain sistem informasi. UML adalah bahasa pemodelan standar atau kumpulan teknik-teknik pemodelan untuk men-spesifikasi, mem-visualisasi, meng-konstruksi dan mendokumentasi hasil kerja dalam pengembangan perangkat lunak (Fowler, 2004). UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi obyek yang berkembang pesat pada akhir tahun 1980an dan awal 1990an.

Secara sederhana UML digunakan untuk menggambar sketsa sistem. Pengembang menggunakan UML untuk menyampaikan beberapa aspek dari sebuah perangkat lunak melalui notasi grafis. UML mendefinisikan notasi dan semantik. Notasi merupakan sekumpulan bentuk khusus yang memiliki makna tertentu untuk menggambarkan berbagai diagram piranti lunak dan semantik mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Ada beberapa jenis diagram yang disediakan dalam UML, antara lain adalah:

- *Use-case diagram*. Diagram ini berguna untuk menggambarkan interaksi antara pengguna dengan sebuah perangkat lunak
- *Activity diagram*. Diagram ini berguna untuk menggambarkan prosedur-prosedur perilaku perangkat lunak.
- *Class diagram*. Diagram ini berguna untuk menggambarkan class, fitur, dan hubungan-hubungan yang terjadi. Pada diagram ini pendekatan berorientasi obyek memegang peranan yang sangat penting.
- *Sequence diagram*. Diagram ini berguna untuk menggambarkan interaksi antar obyek dengan penekanan pada urutan proses atau kejadian.
- *State machine diagram*. Diagram ini digunakan untuk menggambarkan bagaimana suatu kejadian mengubah obyek selama masa hidup obyek tersebut.
- *Component diagram*. Diagram ini berguna untuk menggambarkan struktur dan koneksi komponen.

3.2 TAHAPAN REKAYASA PERANGKAT LUNAK

Seperti telah disebutkan, meskipun dalam pendekatan berbeda-beda, namun model-model di atas memiliki kesamaan, yaitu menggunakan pola tahapan *analysis – design – coding(construction) – testing – maintenance*.

3.2.1 Analisis

Analisis sistem :

Sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

Analisis sistem informasi:

salah satu dari tahapan pengembangan sistem yang memfokuskan pada masalah dan persyaratan-persyaratan bisnis, terpisah dari teknologi apapun yang akan digunakan untuk mengimplementasikan solusi pada masalah tersebut.

(Whitten, 2004).

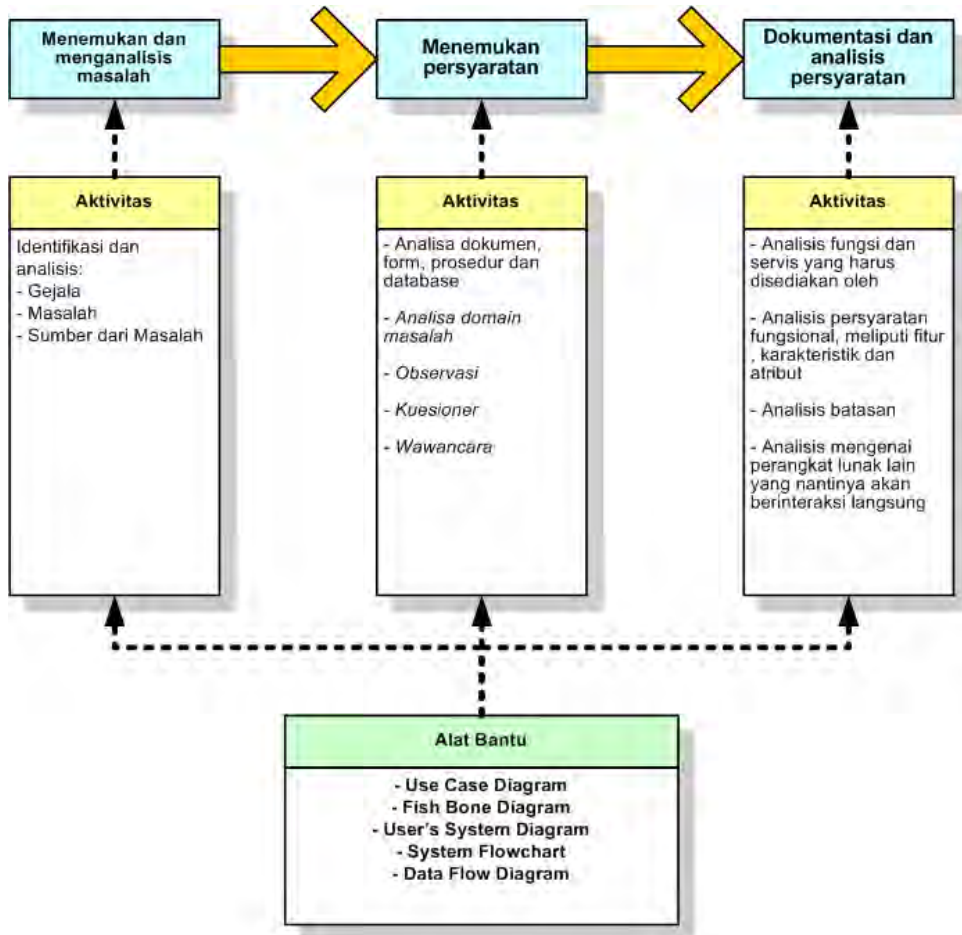
Analisis mungkin adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis. keberhasilannya Tahapan-tahapan dalam analisis rekayasa perangkat lunak secara ringkas dapat dilihat pada Gambar 3.6.

Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis.

Model proses :

Model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi

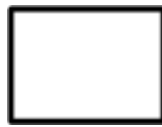
(Harris, 2003).



Gambar 3.6. Tahapan dan aktifitas dalam analisis.

Model proses juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentuk Diagram Arus Data (Data Flow Diagram / DFD). DFD menyajikan gambaran apa yang manusia, proses dan prosedur lakukan untuk mentransformasi data menjadi informasi.

Umumnya ada empat notasi yang sering digunakan dalam DFD seperti tampak Gambar 3.7.



External Entity

External Entity melambangkan sumber data (dari mana data berasal) atau penerima informasi (tujuan akhir dari data). Contoh external entity antara lain konsumen yang memesan suatu produk, manajer yang mengevaluasi laporan penjualan mingguan, dan lain-lain.



Process

Proses adalah serangkaian langkah yang dilakukan untuk memanipulasi data, misalnya pengumpulan, pengurutan, pemilihan, pelaporan, peringkasan, analisis dan lain-lain.



Data Store

Data store adalah tempat untuk menyimpan data untuk digunakan kemudian. Nama yang pada data store ini merupakan abstraksi dari data yang disimpan. Namun detail / item data apa saja yang ada, bagaimana cara akses, atau bagaimana mengorganisasinya tidak dijelaskan dalam notasi ini.

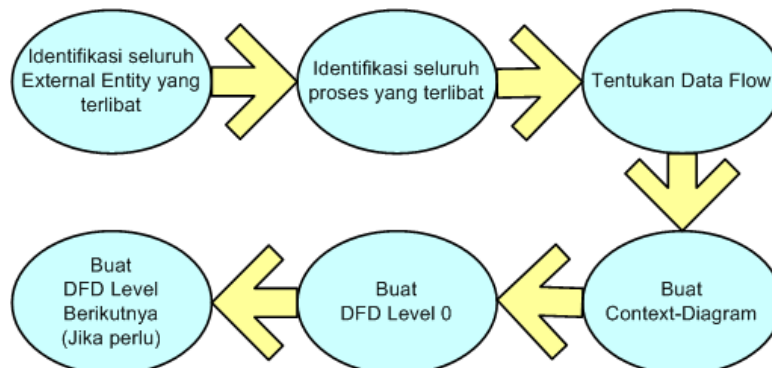


Data Flow

Data flow menunjukkan aliran data dari satu tempat ke tempat lain. Perpindahan data ini dapat dari external entity ke proses, antar proses satu dengan yang lain, dari proses ke data store. Dalam penggambarannya setiap data flow harus diberi label yang menunjukkan data apa yang mengalir.

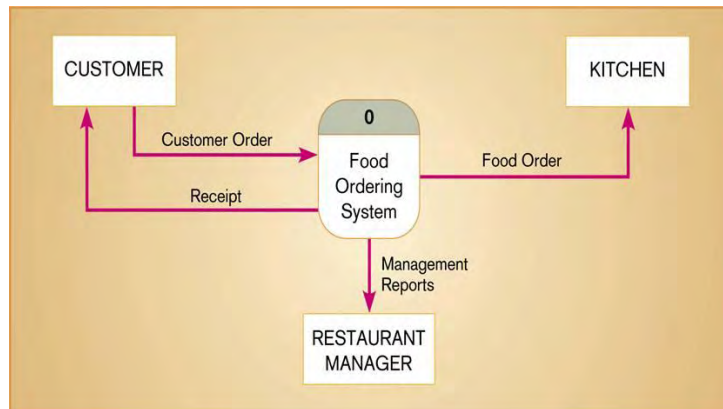
Gambar 3.7. Notasi pada DFD.

Dalam pembuatan DFD ada beberapa tahapan yang dilakukan secara berurutan. Gambar 3.8. menunjukkan urutan tahapan tersebut.



Gambar 3.8. Tahapan pembuatan DFD.

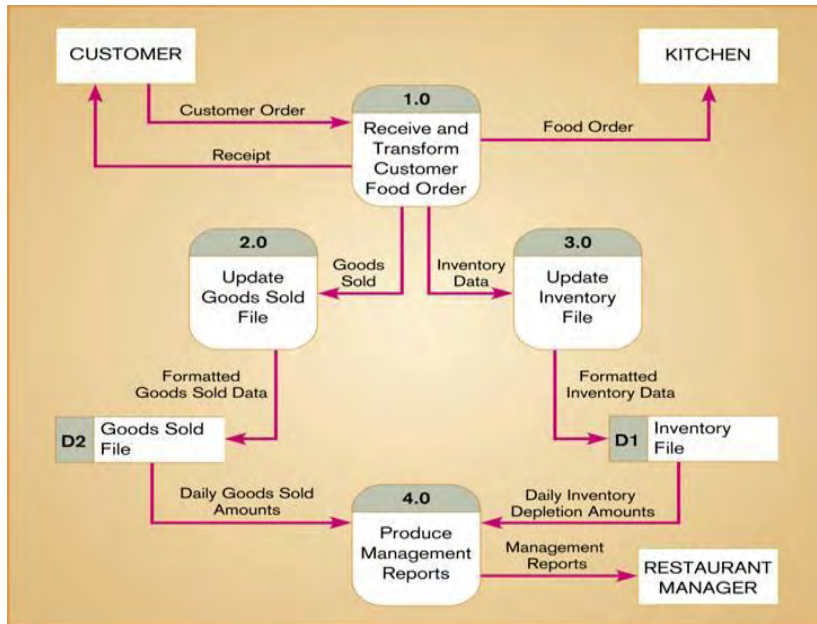
Context diagram adalah DFD ruang lingkup dari sistem yang menunjukkan batas-batas sistem, *external entity* yang berinteraksi dengan sistem dan aliran data utama antara *external entity* dengan sistem. *Context diagram* menggambarkan keseluruhan sistem dalam suatu proses tunggal. Gambar 3.9 menunjukkan sebuah contoh *context diagram*.



Gambar 3.9. *Context diagram* sistem pemesanan makanan (Hoffer et al., 2002).

Context diagram pada gambar 14.10 tampak hanya ada satu proses tunggal yang merepresentasikan sistem yang dimodelkan. Pada proses ini diberi notasi angka 0 untuk menunjukkan ini adalah level paling abstrak dari sistem. Selain itu ada tiga *external entity* yaitu *customer*, *kitchen* dan *restaurant manager*. Ketiganya dapat berperan sebagai sumber data (dalam contoh di atas adalah *customer*) atau sebagai penerima informasi (dalam contoh di atas *customer*, *kitchen*, dan *restaurant manager*). *Data flow* yang tampak pada gambar menunjukkan ada satu *data flow* yang masuk ke sistem dan ada tiga *data flow* yang keluar dari sistem. Masing-masing *data flow* diberi label yang menunjukkan data apa yang sedang mengalir.

Setelah *context diagram* terbentuk dengan benar maka langkah selanjutnya adalah merinci *context diagram* tersebut dalam DFD Level 0. DFD Level 0 adalah DFD yang merepresentasikan proses-proses, data flow dan data storage utama di dalam sistem. DFD Level 0 ini akan digunakan sebagai dasar untuk membangun DFD yang level dibawahnya (Level 1, 2, 3, .. dst) atau biasa disebut sebagai dekomposisi DFD. Gambar 3.10 merupakan DFD level 0 dari *context-diagram* pada gambar 3.9.



Gambar 3.10. DFD Level 0.

Pada gambar 17 tampak adanya pemecahan pada proses dari yang semula hanya satu menjadi empat. Masing-masing proses diberi nomor kode 1.0, 2.0, 3.0 dan 4.0. Jumlah external entity harus tetap yaitu 3 demikian pula data flow yang keluar dan masuk (input dan output) ke dalam sistem harus sama dengan pada context diagram. Sedangkan data flow yang berada di dalam sistem (yang mengalir antar proses dan atau data storage) tergantung pada proses dan data storage yang terlibat. Ada dua data storage yaitu Goods Sold File dan Inventory File. Kedua data storage ini digunakan untuk menyimpan data dari suatu proses. Data ini juga akan dibaca / diakses oleh proses yang lain. Sebagai contoh data storage Inventory File berisi data hasil proses 3.0 (Update Inventory File). Data ini akan digunakan proses 4.0 (Produce Management Reports) untuk membuat laporan yang akan disampaikan pada Restaurant Manager.

DFD level berikutnya yaitu level 1, 2 dan seterusnya diperlukan apabila level sebelumnya dirasa kurang detail. Sebagai contoh apabila DFD level 0 (Gambar 14.12) dirasa belum cukup detail menunjukkan arus data yang mengalir, maka dapat dibuat detailnya pada DFD level 1. Bagian yang harus didetilkan biasanya adalah proses. Detil pada level berikutnya, mungkin pada semua proses atau hanya pada proses-proses tertentu saja. DFD pada level 0 maupun level di bawahnya memiliki kesamaan aturan yang tersaji berikut pada tabel berikut ini.

Tabel 3.1. Aturan-aturan dalam DFD

Kelompok	Aturan
Umum	- input-input ke suatu process akan selalu berbeda dengan output-outputnya - obyek obyek (External Entity, Process, Data Storage, dan Data Flow) yang ada pada suatu DFD selalu memiliki nama yang unik
External Entity	- nama yang dipakai pada External Entity selalu menggunakan kata benda - data tidak boleh mengalir secara langsung dari External Entity yang satu ke External Entity yang lain
Process	- nama yang dipakai pada Process selalu menggunakan kata kerja - tidak ada Process yang hanya menghasilkan output - tidak ada Process yang hanya menerima input
Data Storage	- nama yang dipakai pada Data Storage selalu menggunakan kata benda - data tidak boleh mengalir secara langsung dari Data Storage yang satu ke Data Storage yang lain - data tidak boleh mengalir secara langsung dari External Entity ke Data Storage demikian juga sebaliknya.
Data Flow	- nama yang dipakai pada Data Flow selalu menggunakan kata benda - Data Flow di antara dua notasi hanya memiliki satu arah aliran - Percabangan (fork) menunjukkan adanya data yang persis sama yang mengalir dari suatu tempat ke dua atau lebih tempat yang lain - Penggabungan (join) menunjukkan adanya data yang persis sama yang mengalir dua atau lebih tempat menuju satu tempat yang lain - Data Flow menuju Data Storage berarti terjadi update data - Data Flow dari Data Storage berarti terjadi pembacaan / pengambilan data

3.2.2 Disain

Disain perangkat lunak :

Tugas, tahapan atau aktivitas yang difokuskan pada spesifikasi detil dari solusi berbasis computer.

(Whitten et al, 2004).

Disain perangkat lunak sering juga disebut sebagai *physical design*. Jika tahapan analisis system menekankan pada masalah bisnis (business rule), maka sebaliknya disain perangkat lunak fokus pada sisi teknis dan implementasi sebuah perangkat lunak (Whitten et al, 2004).

Output utama dari tahapan disain perangkat lunak adalah spesifikasi disain. Spesifikasi ini meliputi spesifikasi disain umum yang akan disampaikan kepada stakeholder system dan spesifikasi disain rinci yang akan digunakan pada tahap implementasi. Spesifikasi disain umum hanya berisi gambaran umum agar stakeholder system mengerti akan seperti apa perangkat lunak yang akan dibangun. Biasanya diagram USD tentang perangkat lunak yang baru merupakan point penting dibagian ini. Spesifikasi disain rinci atau kadang disebut disain arsitektur rinci perangkat lunak diperlukan untuk merancang sistem sehingga memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, bernilai, memiliki aspek *user friendly* dan memiliki dasar-dasar untuk pengembangan selanjutnya.

Desain arsitektur ini terdiri dari desain database, desain proses, desain *user interface* yang mencakup desain *input*, *output form* dan *report*, desain hardware, software dan jaringan. Desain proses merupakan kelanjutan dari pemodelan proses yang dilakukan pada tahapan analisis. Desain basisdata akan dibahas pada Bab 14 sampai dengan 18.

3.2.3 Konstruksi

Konstruksi :

Menterjemahkan hasil disain logis dan fisik ke dalam kode-kode program computer

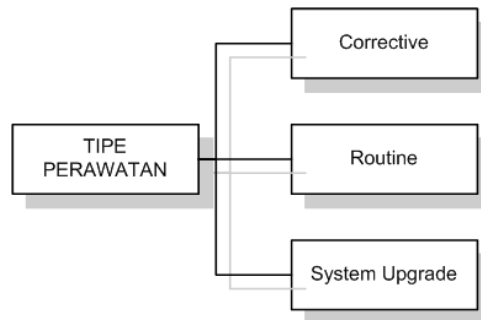
Secara detail bagian tentang pemrograman komputer akan dibahas pada Bab 7 sampai dengan Bab 12.

3.2.4 Pengujian

Pengujian sistem melibatkan semua kelompok pengguna yang telah direncanakan pada tahap sebelumnya. Pengujian tingkat penerimaan terhadap perangkat lunak akan berakhir ketika dirasa semua kelompok pengguna menyatakan bisa menerima perangkat lunak tersebut berdasarkan criteria-kriteria yang telah ditetapkan.

3.2.5 Perawatan dan Konfigurasi

Ketika sebuah perangkat lunak telah dianggap layak untuk dijalankan, maka tahapan baru menjadi muncul yaitu perawatan perangkat lunak. Ada beberapa tipe perawatan yang biasa dikenal dalam dunia perangkat lunak seperti terlihat pada diagram di Gambar 3.11.



Gambar 3.11. Tipe-tipe perawatan.

- Tipe perawatan *corrective* dilakukan jika terjadi kesalahan atau biasa dikenal sebagai bugs. Perawatan bisa dilakukan dengan memperbaiki kode program, menambah bagian yang dirasa perlu atau malah menghilangkan bagian-bagian tertentu.
- Tipe perawatan *routine* biasa juga disebut preventive maintenance dilakukan secara rutin untuk melihat kinerja perangkat lunak ada atau tidak ada kesalahan.
- Tipe perawatan *system upgrade* dilakukan jika ada perubahan dari komponen-komponen yang terlibat dalam perangkat lunak tersebut. Sebagai contoh perubahan platform system operasi dari versi lama ke versi baru menyebabkan perangkat lunak harus diupgrade.

Dasar Sistem Komputer

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian data dan informasi.
 - Memahami pengertian sistem komputer.
 - Mengidentifikasi bagian-bagian dari perangkat keras komputer.
 - Memahami pengertian perangkat lunak.
 - Menyebutkan jenis-jenis perangkat lunak.
-

4.1 DATA DAN INFORMASI

4.1.1 Data

Data:

fakta atau observasi tentang fenomena fisik atau transaksi bisnis atau lebih spesifik, data adalah pengukuran-pengukuran obyektif terhadap atribut-atribut (karakteristik) dari entitas tertentu seperti manusia, tempat, sesuatu atau kejadian

O'Brien (1999)

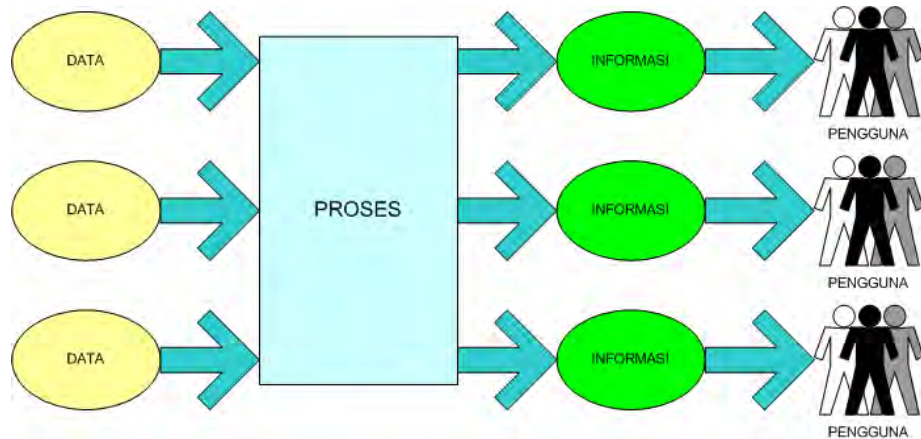
4.1.2 Informasi

Informasi :

Data yang telah diproses atau data yang mempunyai arti (McLeod, 1995).

Data yang ditempatkan agar mempunyai arti dan berguna bagi penggunaanya (O'Brien, 1999).

Kata kunci pada terminology informasi adalah “harus mempunyai arti” karena apabila tidak maka tetap sebagai data. Selain itu kepada siapa informasi itu akan disampaikan juga sangat penting untuk diperhatikan.



Gambar 4.1. Data, proses, informasi dan pengguna.

Tingginya kebutuhan pada informasi tidak hanya pada jumlah, tapi juga pada kualitasnya. Informasi yang sudah basi, tidak akurat, atau sulit dimengerti akan menjadi sesuatu yang tidak berharga bahkan kadang-kadang membahayakan kelangsungan hidup sebuah organisasi. Ada tiga dimensi kualitas pada informasi yaitu waktu (*time*), isi (*content*) dan bentuk (*form*). Tabel 4.1 menunjukkan ringkasan dari karakteristik kualitas informasi berdasarkan tiga dimensi tersebut.

Tabel 4.1. Karakteristik kualitas informasi (O'Brien, 1999).

Karakteristik	Arti
Dimensi Waktu (Time)	
- Timeliness	Informasi harus tersedia ketika dibutuhkan
- Currency	Informasi harus up-to-date ketika disampaikan
- Frequency	Informasi harus tersedia dalam frekuensi yang dibutuhkan
- Time Period	Informasi yang tersedia dapat berupa masa lalu, sekarang dan masa depan
Dimensi Isi (Content)	
- Accuracy	Informasi harus bebas dari kesalahan
- Relevance	Informasi harus sesuai dengan kebutuhan pengguna tertentu pada situasi tertentu
- Completeness	Seluruh informasi yang dibutuhkan harus tersedia
- Conciseness	Hanya informasi yang dibutuhkan yang akan disampaikan
- Scope	Informasi dapat memiliki ruang lingkup yang sempit, luas, internal atau eksternal

Tabel 4.1. Lanjutan.

Karakteristik	Arti
Dimensi Bentuk (Form)	
- Clarity	Informasi harus disampaikan dalam bentuk yang mudah dimengerti
- Detail	Informasi dapat disampaikan dalam bentuk ringkasan atau detil
- Order	Informasi yang disampaikan dapat diatur dalam urutan yang telah ditentukan sebelumnya
- Presentation	Informasi dapat disampaikan dalam bentuk narasi, grafik, atau bentuk-bentuk yang lain
- Media	Informasi dapat disampaikan dalam media cetakan kertas, tampilan video, atau media yang lain.

4.2 SISTEM KOMPUTER

Istilah komputer berasal dari bahasa Latin “Computare” yang berarti menghitung. Oleh karena itu sebenarnya setiap alat yang berfungsi sebagai alat hitung seperti mesin penjumlah, kalkulator, atau bahkan simpoa (abacus) secara teknis dapat disebut sebagai komputer. Namun dalam perkembangannya, komputer mempunyai perkembangan arti yang berbeda.

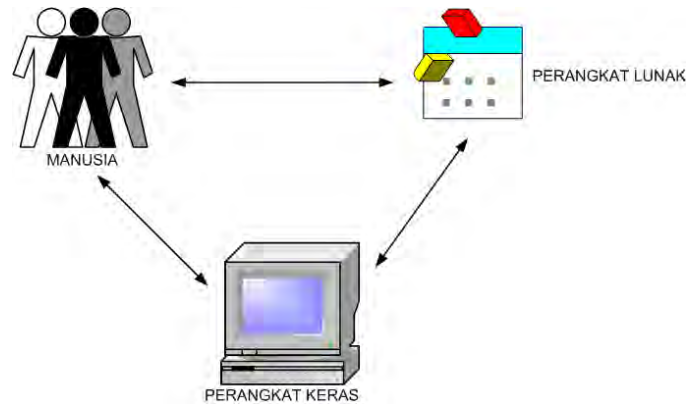
Komputer :

alat pengolah data elektronik yang bekerja dan dikontrol oleh sekumpulan instruksi (program) (Blissmer, 1985)

Sistem komputer :

kumpulan elemen-elemen yaitu manusia, perangkat keras, dan perangkat lunak yang saling berinteraksi untuk mencapai tujuan yaitu mendapatkan informasi yang berguna, kemudahan dalam bekerja, kecepatan dan tujuan lainnya.

Ada tiga komponen utama dalam sistem komputer yaitu manusia sebagai pengguna, perangkat keras dan perangkat lunak (Gambar 4.2). Apabila satu tidak ada maka sistem komputer menjadi tidak bekerja. Sebagai contoh jika hanya ada manusia dan perangkat keras, maka sistem komputer tidak bekerja karena tidak program yang membantu manusia menjalankan perangkat keras.



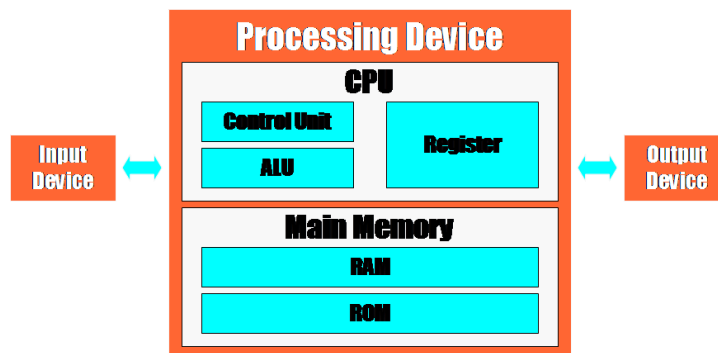
Gambar 4.2. Sistem Komputer.

4.3 PERANGKAT KERAS

Perangkat keras :

Semua bagian fisik computer.

Perangkat keras dibedakan dengan data yang berada di dalamnya atau yang beroperasi di dalamnya, dan perangkat lunak yang menyediakan instruksi buat perangkat keras untuk menyelesaikan tugasnya. Secara umum ada empat komponen dasar pada komputer yang saling terkait (Lihat Gambar 4.3).



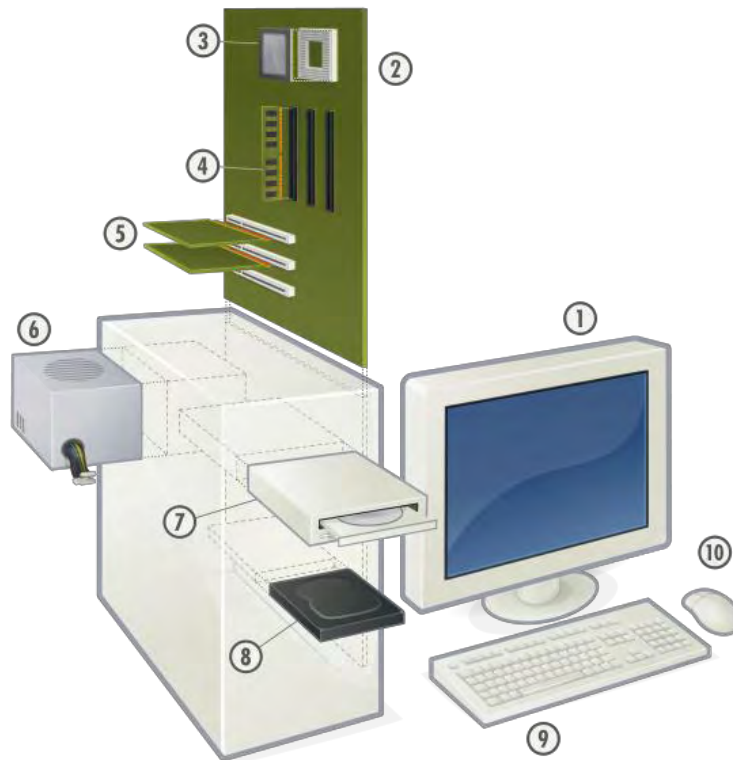
Gambar 4.3. Komponen dasar komputer

- Unit Masukan (Input), yaitu *perangkat yang memungkinkan pengguna memasukkan data atau perintah ke dalam komputer*. Contoh perangkat

yang termasuk dalam unit masukan adalah : keyboard, mouse, joystick, dan digitizer.

- Unit Keluaran (Output), yaitu *perangkat yang memungkinkan pengguna menerima informasi hasil pemrosesan oleh komputer*. Contoh perangkat yang termasuk dalam unit keluaran adalah : monitor, printer, dan plotter.
- Unit Memori Utama (Main memory), yaitu *perangkat yang digunakan untuk menyimpan data, program, dan informasi hasil pemrosesan komputer pada saat pemrosesan*. Unit memori utama terdiri dari banyak sel, yang masing-masing dapat menyimpan satu satuan informasi. Unit memori utama terdiri dari dua bagian, yaitu **ROM (Read Only Memory)** dan **RAM (Random Access Memory)**. ROM hanya dapat ditulisi sekali saja dan selanjutnya hanya dapat dibaca. RAM dapat ditulisi, dihapus dan dibaca berulang kali. Data, program, dan informasi yang sedang diproses disimpan dalam RAM ini, dan akan hilang apabila komputer dimatikan. Itu sebabnya data, program, dan informasi yang akan digunakan lagi disimpan dalam media penyimpanan tambahan (secondary storage) seperti, hard disk, disket, CD, tape dan lain-lain.
- Unit Pemrosesan Pusat (*Central Processing Unit*), yaitu bagian yang digunakan untuk memproses data, program, dan informasi pada komputer. Ada dua bagian penting dalam CPU yaitu *Arithmetic and Logical Unit* (ALU) dan *Control Unit*. Banyak orang menyebutkan ALU adalah jantung dari sebuah komputer. ALU bertanggung jawab pada dua operasi dasar yaitu operasi aritmatik dan perbandingan. Sedangkan *Control Unit* bertanggung jawab untuk menkoordinasi semua aktivitas unit-unit lain, misalnya bagaimana keyboard dapat dikenali dan bekerja sebagai unit input yang dimengerti aktivitasnya.

Secara fisik, arsitektur umum dari sebuah komputer yang biasa kita kenal (*Personal Computer/PC*) dapat dilihat pada Gambar 4.4. Sebuah PC merupakan rangkaian dari berbagai macam komponen yang memiliki fungsi masing-masing.



Keterangan gambar:

1. *Display*
2. *Motherboard*
3. *CPU*
4. *Main Memory*
5. *Expansion Cards*
6. *Power Supply*
7. *Optical Disc Drive*
8. *Secondary Storage (Hard Disk)*
9. *Keyboard*
10. *Mouse*

Gambar 4.4. Perangkat keras komputer.

Berikut ini penjelasan singkat tentang komponen-komponen fisik dalam sebuah komputer :

1. *Display*. Komponen *display* atau *monitor* termasuk dalam unit keluaran sebuah komputer. Sebuah kabel menghubungkan monitor dengan adapter video yang diinstal pada slot ekspansi *motherboard*. Komputer mengirimkan signal kepada adapter video, mengenai karakter, gambar

atau grafik apa yang harus ditampilkan. Adapter video akan mengkonversi signal menjadi sekumpulan instruksi tentang bagaimana monitor harus menampilkan teks, atau gambar pada layarnya.



Gambar 4.5. Display atau monitor.

2. *Motherboard.* *Motherboard* atau dikenal juga sebagai *mainboard*, *system board* atau *logic board* (pada Apple Computer) dan kadang disingkat sebagai *mobo* adalah pusat dari papan sirkuit utama pada sebuah sistem elektronik, seperti perangkat komputer modern. Pada komponen ini akan diletakkan (ditancapkan) komponen-komponen lain seperti memori utama, *processor*, adapter video, adapter suara dan lain-lain, sehingga terbentuk sistem komputer yang komplit dan dapat bekerja.



Gambar 4.6. Motherboard sebuah komputer.

3. CPU. *Central Processing Unit* (CPU), atau sering disebut sebagai *Processor*, adalah komponen pada komputer digital yang menginterpretasi instruksi dan memproses data pada suatu program komputer. CPU menyediakan bagian penting dari suatu sistem digital yaitu kemampuan untuk diprogram. Komponen ini merupakan komponen yang harus ada pada setiap perangkat komputer.



AMD Athlon processor

Intel processor

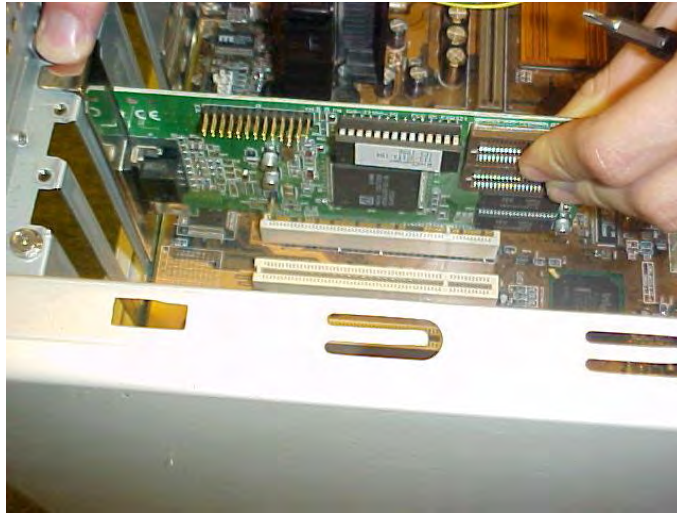
Gambar 4.7. Central Processing Unit (CPU)

4. *Main Memory*. *Main Memory* atau kadang disebut sebagai *Primary Storage*, atau *Internal Memory*, adalah memori komputer yang secara langsung dapat diakses oleh CPU tanpa menggunakan jalur input/output komputer. Komponen ini digunakan untuk menyimpan data yang sedang aktif digunakan. *Primary storage* dapat terdiri dari beberapa tipe penyimpanan seperti *main storage*, *cache memory*, dan *special registers*.



Gambar 4.8. Berbagai jenis main memory.

5. *Expansion Cards.* *Expansion card* (kartu ekspansi) adalah sebuah *printed circuit board* (PCB) yang dapat ditancapkan pada slot ekspansi yang tersedia pada *motherboard* komputer untuk menambah fungsionalitas dari komputer. Contoh expansion card antara lain kartu adapter video, kartu adapter audio, kartu adapter jaringan dan lain-lain.



Gambar 4.9. Pemasangan expansion card.

6. *Power Supply.* *Power supply* atau kadang-kadang disebut PSU (*Power Supply Unit*) adalah perangkat yang menyuplai energi listrik atau energi jenis lain pada komponen lain dalam komputer.



Gambar 4.10. Power Supply Unit.

7. *Optical Disc Drive*. *Optical Disc* adalah sebuah media penyimpanan sekunder yang berbentuk seperti piringan hitam, namun dalam ukuran yang lebih kecil. Data yang tersimpan dalam *Optical Disc* diakses ketika material yang spesifik pada *Optical Disc* disinari oleh sinar laser. Ada dua tipe utama dalam *Optical Disc* yaitu yang berbasis pada CD (*Compact Disc*) dan yang berbasis pada DVD (*Digital Versatile Disc*). Perangkat untuk membaca, menulis, atau menghapus disebut *Optical Disc Drive*.



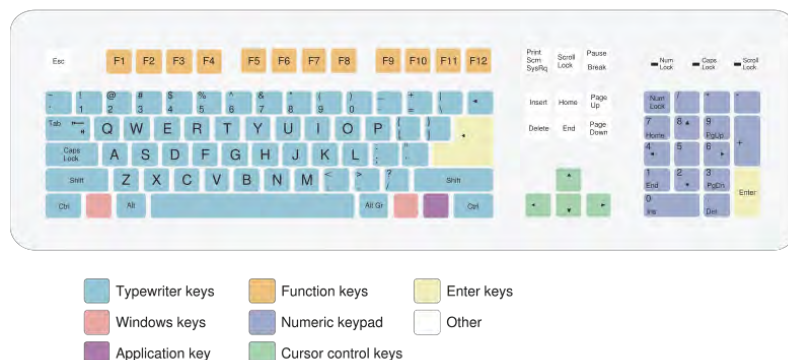
Gambar 4.11. CD-RW Drive, salah satu contoh Optical Disc Drive.

8. *Secondary Storage (Hard Disk)*. *Secondary Storage* adalah perangkat yang digunakan untuk membantu *Primary Storage (main memory)*, terutama untuk menyimpan data, program, atau informasi yang akan digunakan lagi. Berbeda dengan *primary storage*, data, program dan informasi pada *secondary storage* tidak akan hilang meskipun komputer dimatikan, kecuali apabila memang sengaja dihapus. *Secondary storage* yang paling banyak ditemui dalam setiap komputer adalah *Hard Disk*. *Hard disk* akan menyimpan data dengan menggunakan material bersifat magnetic dalam pola-pola tertentu yang merepresentasikan data.



Gambar 4.12. Hard Disk

9. **Keyboard.** *Keyboard* atau papan kunci, perangkat yang digunakan untuk menginputkan teks dan karakter pada komputer. Perangkat ini juga dapat digunakan untuk mengontrol fungsi-fungsi khusus pada komputer. Gambar 4.13 menunjukkan lay out sebuah *keyboard* yang umum kita jumpai.



Gambar 4.13. Skema umum sebuah keyboard.

10. **Mouse.** *Mouse*, biasanya terdiri dari pointing device, yang digunakan untuk mendeteksi pergerakan relative dari dua permukaan secara dua dimensi yang kemudian ditampilkan pada display. Sebagai tambahan, pada mouse seringkali ditambahkan fungsi lain, seperti "*wheels*" atau roda. Selain mendeteksi pergerakan, mouse juga berperan dalam mengeksekusi perintah dengan cara menekan tombol pada *mouse* sekali (click) atau dua kali berurutan (double click).



Gambar 4.14. Berbagai jenis mouse.

4.4 PERANGKAT LUNAK

Perangkat lunak/piranti lunak :

Program komputer yang berfungsi sebagai sarana interaksi antara pengguna dan perangkat keras.

'Penterjemah' perintah-perintah yang dijalankan pengguna komputer untuk diteruskan ke atau diproses oleh perangkat keras.

Program komputer ini, isinya dapat diubah dengan mudah. Pada komputer, perangkat lunak dimuat ke dalam RAM kemudian dieksekusi di dalam CPU. Pada level paling bawah, perangkat lunak berisi bahasa mesin yang bersifat spesifik terhadap suatu processor.

4.4.1 Klasifikasi

Ada banyak model penggolongan perangkat lunak, namun secara umum perangkat lunak dapat dibagi menjadi tiga kelompok, yaitu :

- *System Software*

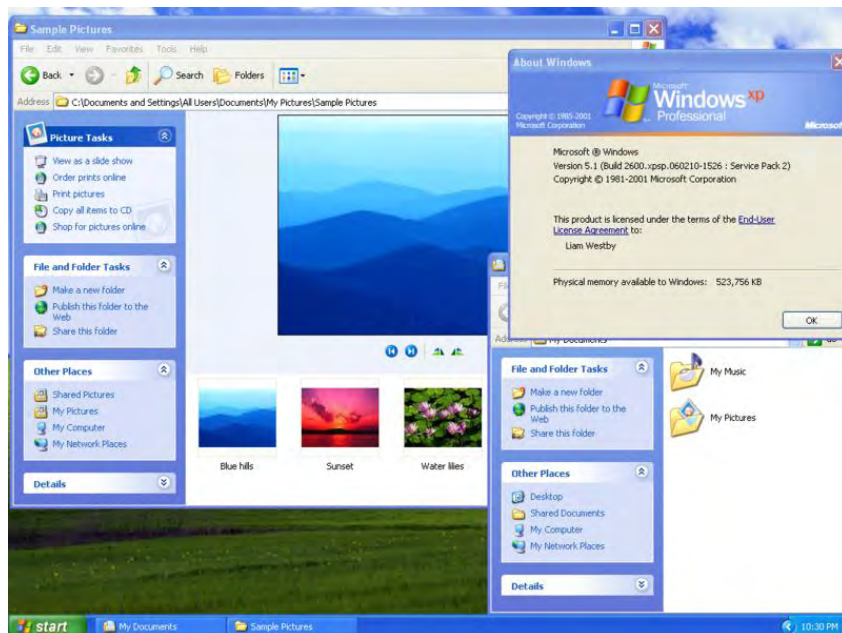
System Software :

Perangkat lunak yang digunakan untuk membantu menjalankan perangkat keras dan sistem komputer.

Tujuan dari system software :

membatasi semaksimal mungkin programmer aplikasi dari kompleksitas sebuah komputer, terutama yang berhubungan dengan akses memori dan perangkat keras secara langsung.

Termasuk dalam kelompok ini adalah sistem operasi, *driver* perangkat keras, perangkat lunak pendidagnosa, *windowing system*, *utilities* dan lain-lain. Dari kelompok ini sistem operasi merupakan perangkat lunak yang paling penting. Perangkat lunak ini bekerja sebagai antar muka antara komputer dengan dunia luar. Pada bagian hardware, sistem operasi akan mendiskripsikan perangkat keras yang ada atau terhubung dengan komputer. Sistem operasi menyediakan antar muka pada perangkat keras ini menggunakan “driver” tertentu sehingga perangkat ini dapat dikenali dan bekerja sebagai mana mestinya. Penjelasan lebih detil tentang sistem operasi dapat dilihat pada Bab 5.



Gambar 4.15. Tampilan desktop sistem operasi Windows XP.

- *Programming Software*

Programming Software :

Perangkat lunak yang menyediakan alat bantu atau fungsi yang dapat membantu programmer dalam membuat program komputer.

Software ini sangat bergantung pada bahasa pemrograman yang digunakan. Alat bantu ini meliputi editor teks, compiler, interpreter, linkers, debugger dan lain-lain. Sebuah *Integrated Development Environment* (IDE)

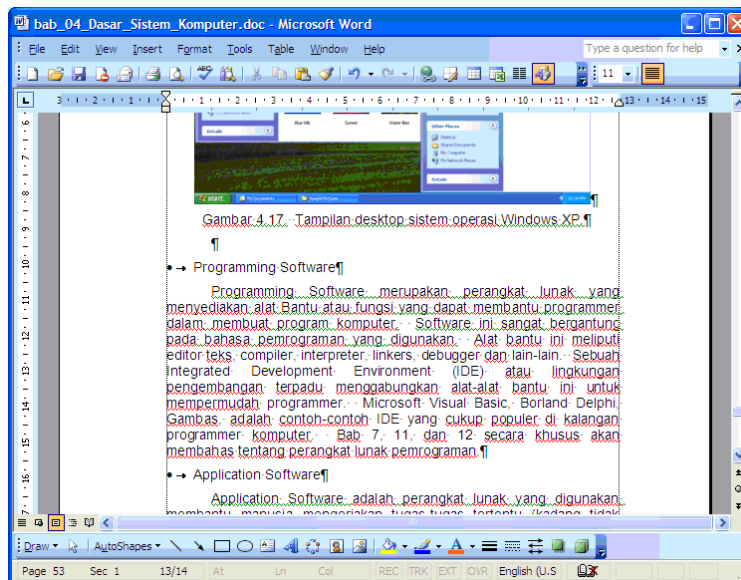
atau lingkungan pengembangan terpadu menggabungkan alat-alat bantu ini untuk mempermudah programmer. **Microsoft Visual Basic, Borland Delphi, Gambas**, adalah contoh-contoh IDE yang cukup populer di kalangan programmer komputer. Bab 7, 11, dan 12 secara khusus akan membahas tentang perangkat lunak pemrograman.

- *Application Software*

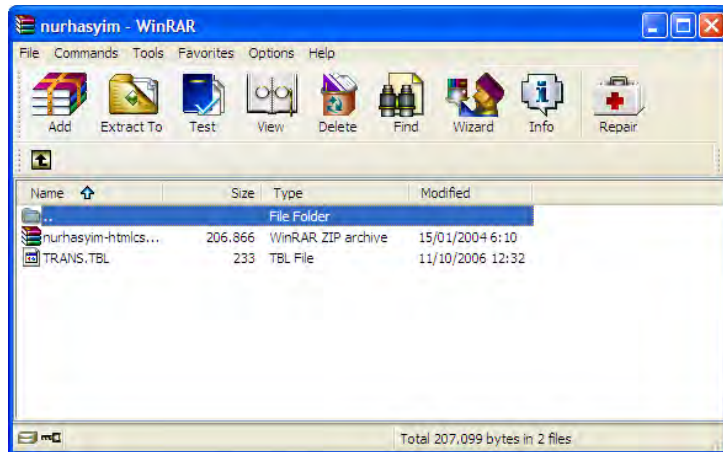
Application Software :

Perangkat lunak yang digunakan membantu manusia mengerjakan tugas-tugas tertentu (kadang tidak berhubungan dengan komputer.

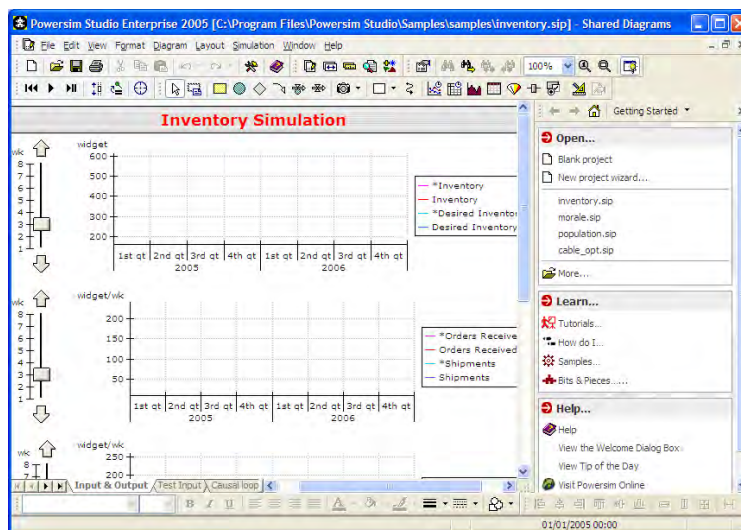
Tipe-tipe *application software* antara lain, perangkat lunak otomatisasi industri, perangkat lunak bisnis, perangkat lunak pendidikan, perangkat lunak software, database, dan game komputer. Beberapa contoh application software dapat dilihat pada Gambar-Gambar berikut ini.



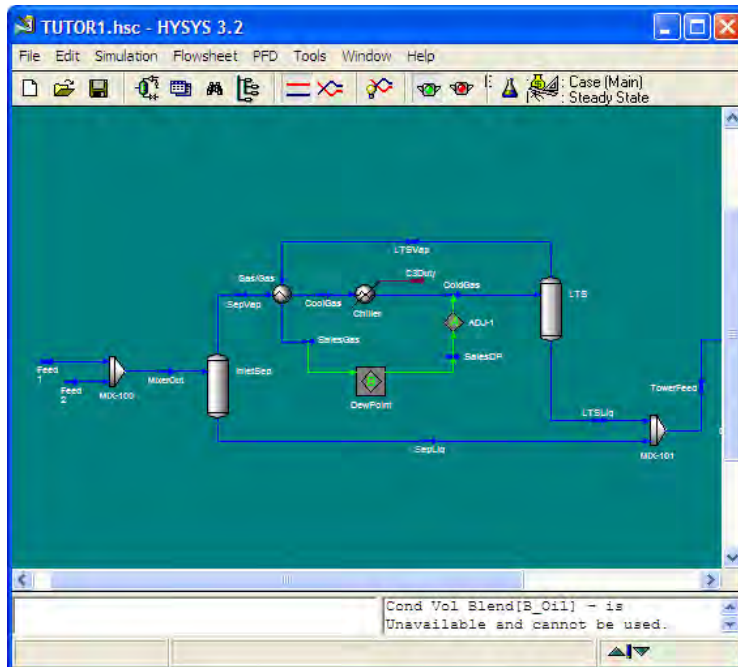
Gambar 4.16. *Application software* Microsoft Word (Software pengolah kata).



Gambar 4.17. Application software Winrar (Software kompresi dan ekstraksi file).



Gambar 4.18. Application software PowerSim (Software untuk simulasi sistem).



Gambar 4.19. *Application software* Hysis (Software untuk perancangan pabrik).

Sistem Operasi

TUJUAN

Setelah anda membaca Bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian sistem operasi.
 - Memahami fungsi-fungsi sistem operasi.
 - Memahami kerja BIOS.
 - Menjalankan proses instalasi dan booting sistem operasi.
 - Menjalankan sistem operasi dengan mode teks maupun GUI pada sistem operasi.
-

Seperti telah disebutkan pada bab terdahulu, sistem operasi termasuk dalam kelompok *system software* yaitu perangkat lunak yang berperan dalam menjalankan perangkat keras komputer dan sistem komputer secara keseluruhan.

Sistem Operasi :

perangkat lunak yang bertugas mengelola penggunaan sumberdaya dalam komputer dan menyediakan antarmuka bagi pengguna untuk mengakses sumberdaya tersebut.

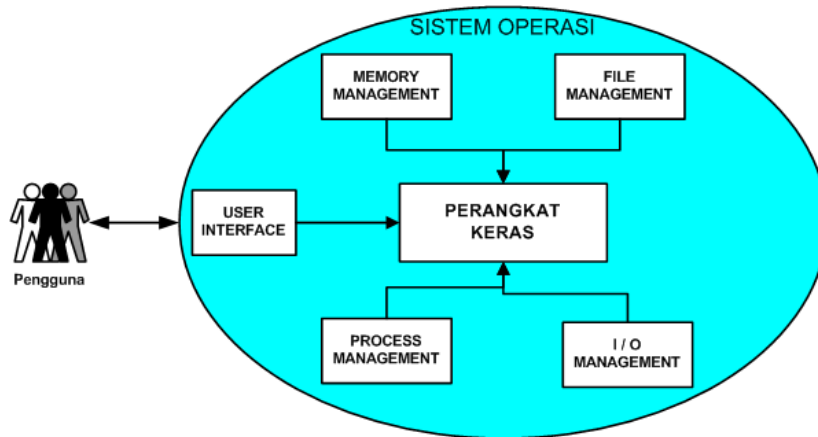
5.1 FUNGSI

Fungsi-fungsi sebuah sistem operasi secara umum dapat dilihat pada gambar 5.1.

5.1.1 Antar muka pengguna

Fungsi ini merupakan fungsi yang paling mudah dikenali oleh pengguna karena melalui fungsi ini pengguna dapat berinteraksi dengan sistem operasi, perangkat keras maupun perangkat lunak yang lain. Sistem operasi pada dasarnya menunggu input atau instruksi dari pengguna dan kemudian menerjemahkan perintah-perintah tersebut dalam bahasa yang

dimengerti oleh komputer. Antar muka pengguna menjadi tempat bagi pengguna untuk menuliskan atau menyampaikan perintah tersebut.



Gambar 5.1. Fungsi-fungsi sistem operasi.

Secara garis besar ada dua model antar muka pengguna yaitu Command Line Interface (CLI) dan Graphical User Interface (GUI). CLI memberikan fasilitas bagi pengguna untuk memberikan perintah dalam bentuk teks (IGambar 5.2.) sedangkan GUI lebih berbasis pada tampilan grafis (Gambar 5.3.). Dewasa ini hampir semua sistem operasi modern menyediakan model GUI sebagai antar muka pengguna. Beberapa menyediakan GUI yang terintegrasi dengan kernel sistem operasi, misalnya pada Microsoft Windows dan Apple Mac OS versi awal. Sedangkan yang lainnya menyediakan GUI yang bersifat modular, yaitu tidak terintegrasi langsung pada kernel sistem operasinya, seperti pada Unix, Linux dan Mac OS versi X ke atas.

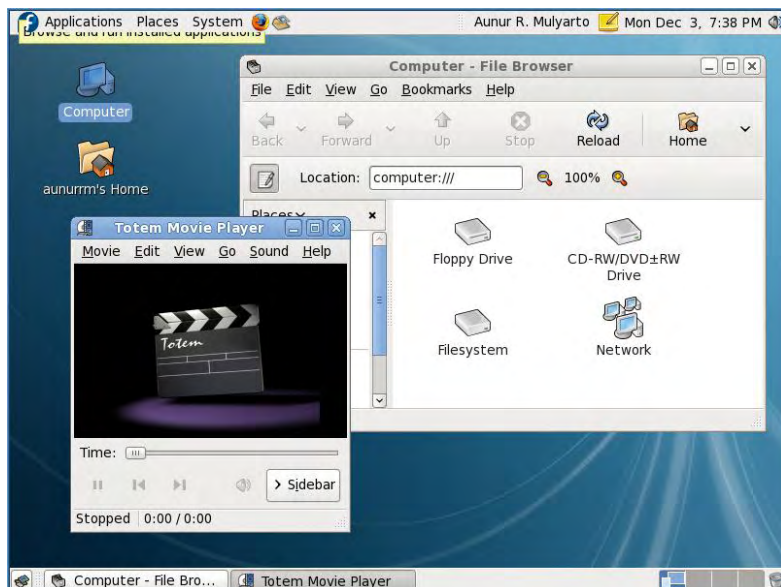
```

bash-2.05b$ pwd
/home/dstone
bash-2.05b$ cd /usr/portage/app-shells/bash
bash-2.05b$ ls -al
total 68
drwxr-xr-x 3 root root 4096 May 14 12:05 .
drwxr-xr-x 26 root root 4096 May 17 02:36 ..
-rw-r--r-- 1 root root 13710 May 3 22:35 ChangeLog
-rw-r--r-- 1 root root 2924 May 14 12:05 Manifest
-rw-r--r-- 1 root root 3720 May 14 12:05 bash-2.05b-r11.ebuild
-rw-r--r-- 1 root root 3516 May 2 20:05 bash-2.05b-r9.ebuild
-rw-r--r-- 1 root root 5083 May 3 22:35 bash-3.0-r11.ebuild
-rw-r--r-- 1 root root 4038 May 14 12:05 bash-3.0-r7.ebuild
-rw-r--r-- 1 root root 3931 May 14 12:05 bash-3.0-r8.ebuild
-rw-r--r-- 1 root root 4267 Mar 29 21:11 bash-3.0-r9.ebuild
drwxr-xr-x 2 root root 4096 May 3 22:35 files
-rw-r--r-- 1 root root 164 Dec 29 2003 metadata.xml
bash-2.05b$ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
</pkgmetadata>
bash-2.05b$ sudo /etc/init.d/bluetooth status
Password:
* status: stopped
bash-2.05b$ ping -q -c1 en.wikipedia.org
PING rr.chtpa.wikimedia.org (207.142.131.247) 56(84) bytes of data.

--- rr.chtpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=3
/dev/sda1 /nnt/usbkey
/dev/sda2 /nnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ lsmod
Module                               Size  Used by
joydev                               8256  0
ipu2200                             175112  0
ieee80211                            44228  1 ipu2200
ieee80211_crypt                      4872  2 ipu2200,ieee80211
e1000                                84468  0
bash-2.05b$ █

```

Gambar 5.2. Antar muka pengguna model CLI pada Linux / Unix.



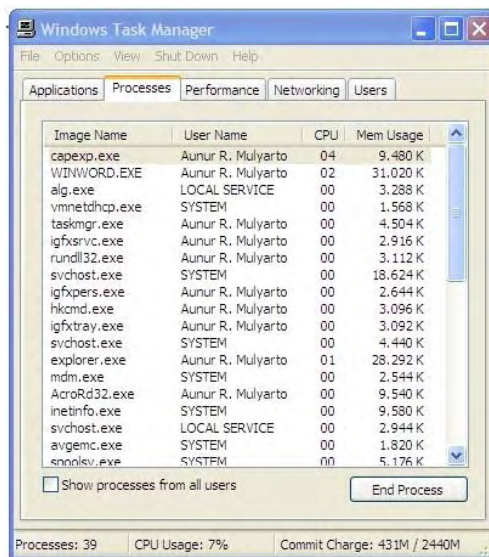
Gambar 5.3. Antar muka pengguna model GUI pada Linux Fedora.

5.1.2 Manajemen memori

Memori utama atau lebih dikenal sebagai memori adalah sebuah *array* yang besar dari *word* atau *byte*, yang ukurannya mencapai ratusan, ribuan, atau bahkan jutaan. Setiap *word* atau *byte* mempunyai alamat tersendiri. Memori utama berfungsi sebagai tempat penyimpanan instruksi/data yang akses datanya digunakan oleh CPU dan perangkat Masukan/Keluaran. Memori utama termasuk tempat penyimpanan data yang bersifat volatile -- tidak permanen -- yaitu data akan hilang kalau komputer dimatikan.

Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen memori seperti:

- Menjaga *track* dari memori yang sedang digunakan dan siapa yang menggunakannya.
- Memilih program yang akan di-*load* ke memori.



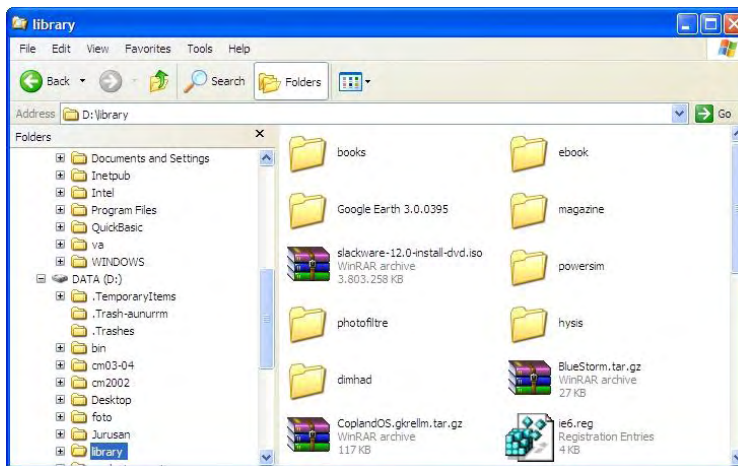
Gambar 5.4. Manajemen memori pada sistem operasi Microsoft Windows

5.1.3 Manajemen file

File (berkas) adalah kumpulan informasi yang berhubungan, sesuai dengan tujuan pembuat berkas tersebut. Umumnya file merepresentasikan program dan data. File dapat mempunyai struktur yang bersifat hirarkis (direktori, volume, dll.). Sistem operasi mengimplementasikan konsep abstrak dari file dengan mengatur media penyimpanan massa, misalnya *tapes* dan *disk*.

Sistem operasi bertanggung-jawab dalam aktivitas yang berhubungan dengan manajemen file :

- Pembuatan dan penghapusan file.
- Pembuatan dan penghapusan direktori.
- Mendukung manipulasi berkas dan direktori.
- Memetakan berkas ke *secondary-storage*.
- Mem-back-up berkas ke media penyimpanan yang tidak permanen (*non-volatile*).



Gambar 5.5. Windows Explorer sebagai sarana pengelolaan file.

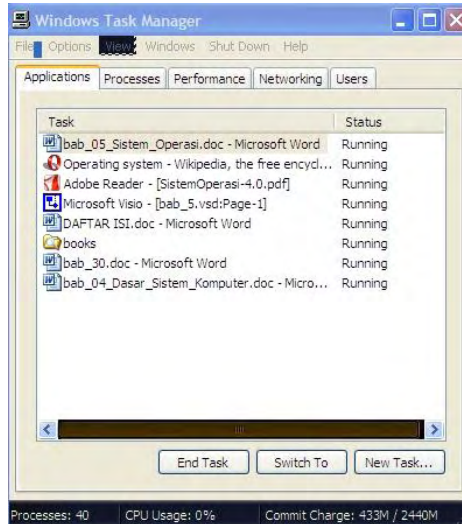
5.1.4 Manajemen proses

Proses adalah sebuah program yang sedang dieksekusi. Sebuah proses membutuhkan beberapa sumber daya untuk menyelesaikan tugasnya. Alokasi sumber daya tersebut dikelola oleh Sistem Operasi. Misalnya, penggunaan memori oleh CPU, file-file yang terbuka, dan penggunaan oleh perangkat-perangkat input/output lain. Ketika proses tersebut berhenti dijalankan, sistem operasi akan mendapatkan kembali semua sumber daya yang bisa digunakan kembali.

Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen proses seperti:

- Membuat dan menghapus proses pengguna dan sistem proses.
- Menunda atau melanjutkan proses.
- Menyediakan mekanisme untuk sinkronisasi proses.
- Menyediakan mekanisme untuk komunikasi proses.

- Menyediakan mekanisme untuk penanganan *deadlock*.



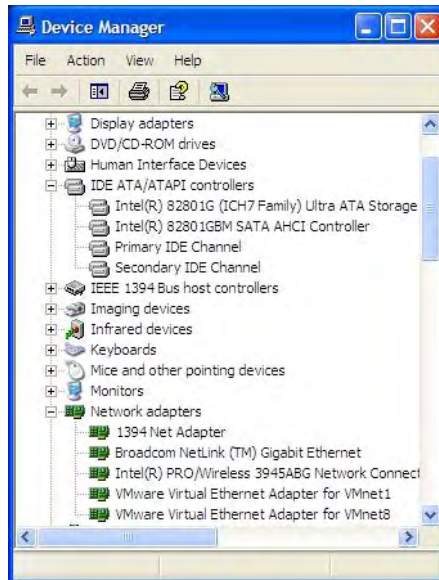
Gambar 5.6. Manajemen proses pada sistem operasi Microsoft Windows.

5.1.5 Manajemen sistem masukan dan keluaran (I / O)

Sistem ini sering disebut dengan *device manager*. Menyediakan *device driver* yang umum sehingga operasi Masukan/Keluaran dapat seragam (membuka, membaca, menulis, menutup). Contoh: pengguna menggunakan operasi yang sama untuk membaca berkas pada perangkat keras, *CD-ROM* dan *floppy disk*.

Komponen Sistem Operasi untuk sistem Masukan/Keluaran:

- Penyangga: menampung sementara data dari/ke perangkat Masukan/Keluaran.
- *Spooling*: melakukan penjadwalan pemakaian Masukan/Keluaran sistem supaya lebih efisien (antrian dsb.).
- Menyediakan *driver*: untuk dapat melakukan operasi rinci untuk perangkat keras Masukan/Keluaran tertentu.



Gambar 5.7. Manajemen I / O pada sistem operasi Microsoft Windows.

5.2 BIOS

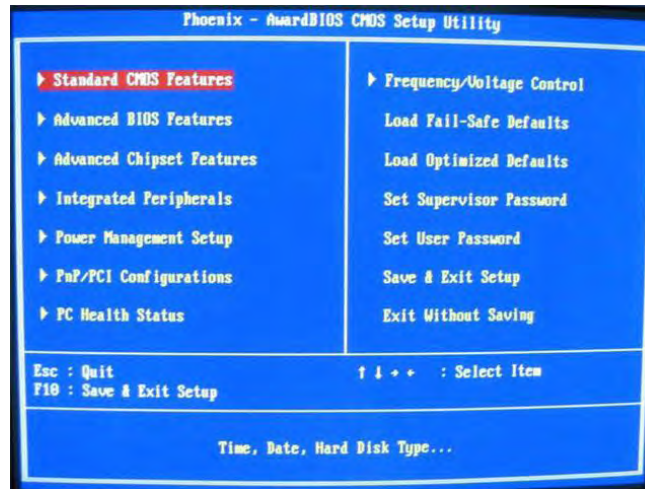
BIOS merupakan singkatan dari Basic Input/Output System. BIOS adalah kode-kode program yang pertama kali dijalankan ketika komputer dinyalakan (booting). Fungsi utama BIOS adalah untuk mengidentifikasi dan mengenali perangkat keras komputer. Biasanya BIOS akan tersimpan dalam ROM (Read Only Memory) yang ada pada motherboard suatu komputer.

Ketika komputer dinyalakan maka BIOS akan mencoba mengenali bagian-bagian komputer berikut ini:

- *clock generator.*
- *processors dan caches.*
- *chipset (memory controller and I/O controller).*
- *system memory.*
- *Semua perangkat PCI*
- *primary graphics controller.*
- *Mass storage controllers (seperti SATA and IDE controllers).*
- *Various I/O controllers (such keyboard/mouse and USB).*

Setelah dikenali maka BIOS akan memanggil program untuk boot suatu sistem operasi (boot loader).

Kita dapat melakukan setting BIOS dengan menggunakan fasilitas yang disediakan oleh BIOS. Biasanya dengan menekan tombol Del atau F2 (tergantung jenis komputernya) ketika komputer baru dinyalakan. Jika berhasil masuk maka kita akan disuguhi tampilan seperti pada Gambar 5.8. kita dapat melakukan serangkain pengaturan pada perangkat keras yang ada pada komputer.



Gambar 5.8. Tampilan BIOS utility.

5.3 JENIS-JENIS SISTEM OPERASI

Sistem operasi telah berkembang melalui jalan yang panjang. Dari yang paling sederhana sampai yang paling modern dewasa ini. Masing-masing memiliki kelebihan dan kekurangan terutama sehubungan dengan fungsi-fungsi yang dimilikinya. Pada bagian berikut ini akan dibahas beberapa sistem operasi yang banyak digunakan dan familiar bagi pengguna komputer.

5.3.1 DOS

DOS adalah singkatan dari Disk Operating System. DOS merujuk pada perangkat sistem operasi yang digunakan di banyak komputer yang menyediakan abstraksi dan pengelolaan perangkat penyimpanan sekunder dan informasinya. Misalnya penggunaan sistem file yang mengelola file-file yang ada pada perangkat penyimpanan. DOS biasanya dijalankan dari satu atau dua disc. Hal ini karena pada masa DOS digunakan media penyimpan masih sangat terbatas kemampuannya (paling besar mungkin hanya 1,4 Megabyte).

Ada banyak jenis DOS diantaranya Apple DOS, Commodore DOS, Atari DOS dan lain-lain. Jenis ini sangat bergantung dengan jenis perangkat

komputernya. Jenis DOS yang paling terkenal adalah jenis DOS yang berjalan pada mesin-mesin yang compatible dengan IBM Personal Computer.

Untuk menjalankan perintah-perintah sistem operasi, DOS menggunakan perintah berbasis teks atau CLI. Setiap kali selesai mengetikkan suatu perintah, kita harus menekan tombol ENTER untuk mengeksekusi perintah tersebut. Contoh operasi dengan menggunakan DOS dapat dilihat pada Gambar 5.9.

```
C:\>dir/w
Volume in drive C has no label.
Volume Serial Number is 5CC3-1976

Directory of C:\

[AppServ]          AUTOEXEC.BAT          CONFIG.SYS
[Documents and Settings] [Inetpub]          [Intel]
[ISACER.id]        [Program Files]      [QuickBasic]
[Val]              [WINDOWS]
                  3 File(s)          7 bytes
                  8 Dir(s)          9.171.398.656 bytes free

C:\>cd QuickBasic

C:\QuickBasic>dir/w
Volume in drive C has no label.
Volume Serial Number is 5CC3-1976

Directory of C:\QuickBasic

[.]                [QBASIC-7] [QBASIC_4.5]
                  0 File(s)          0 bytes
                  4 Dir(s)          9.171.394.560 bytes free

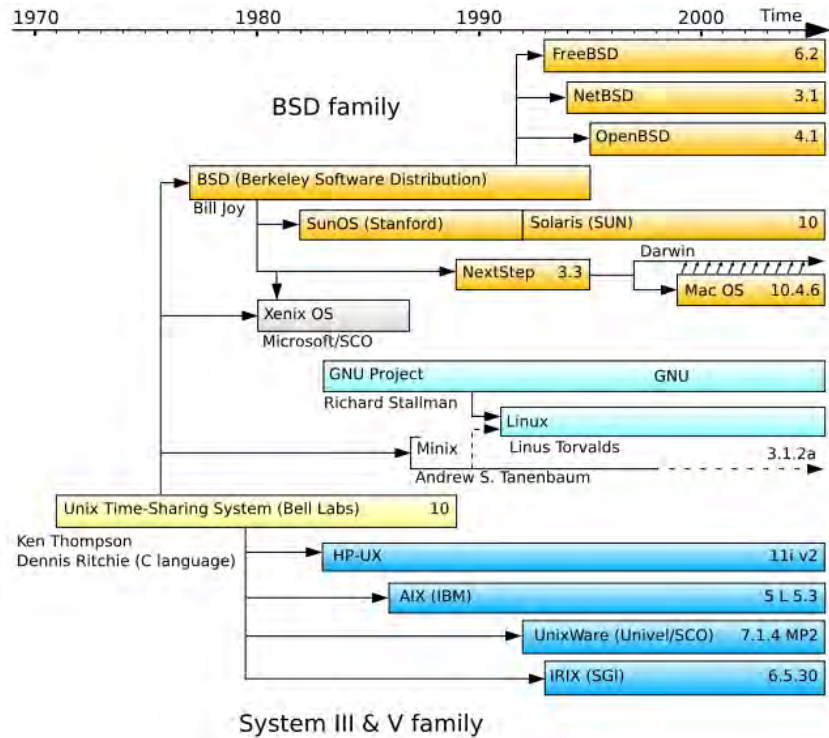
C:\QuickBasic>
```

Gambar 5.9. Contoh penggunaan DOS.

5.3.2 UNIX

UNIX adalah sistem operasi yang mula-mula dikembangkan oleh suatu kelompok di AT & T pada laboratorium Bell. Unix banyak digunakan baik untuk server maupun workstation. Lingkungan Unix dan model program client-server menunjukkan bahwa Unix lebih dikembangkan sebagai sistem operasi yang kuat di jaringan komputer dari pada sistem operasi untuk computer personal.

UNIX dirancang untuk portable, multi-tasking, dan multi-user. Konsep utama Unix antara lain banyak menggunakan file teks biasa untuk menyimpan data, menggunakan sistem file berjenjang, memperlakukan perangkat sebagai suatu file, dan menggunakan banyak program kecil yang eksekusinya pada CLI dapat digabung dengan tanda pipeline (|). Pada Gambar 5.2 di atas, tampak beberapa perintah UNIX yang digabung dengan pipeline. Konsep yang sangat solid dan stabil membuat Unix banyak dijadikan dasar sistem operasi modern. Gambar 5.10 menunjukkan bagaimana Unix merupakan dasar dari banyak sistem operasi yang ada sekarang.



Gambar 5.10. Unix dan sistem operasi turunannya.

Sistem UNIX terdiri dari beberapa komponen yang biasanya dipaket bersama. Umumnya paket-paket tersebut adalah sebagai berikut:

- *Kernel* dengan sub komponen seperti :
 - *conf* — file konfigurasi
 - *dev* — driver perangkat keras
 - *sys* — kernel sistem operasi, manajemen memori, penjadwalan proses, sistem calls dan lain-lain.
 - *h* — header files, mendefinisikan struktur kunci di dalam sistem.

```

16087 ?? IW      0:00.04 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
16200 ?? DW      0:00.00 grep -v ^ *+ conftest.erl
16610 ?? IW      0:04.82 /bin/bash /home/users/t/ta/tanders/src/net-snmp-V5-1-pa
16929 ?? I       0:00.16 sh Compile cvs/RELEASE wxGTK
17066 ?? I       0:00.00 sh Compile cvs/RELEASE wxGTK
17686 ?? IW      0:00.03 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
17774 ?? IW      0:06.28 /bin/bash /home/users/t/ta/tanders/src/net-snmp-V5-3-pa
18239 ?? I       0:00.00 sh Compile cvs/RELEASE wxGTK
19497 ?? IWs     0:00.12 sshd: tanders@notty
23894 ?? S       0:00.07 gmake
24011 ?? S       2:09.33 snmpd -d -r -U -p /tmp/snmp-test-31-11960/snmpd.pid.num
24544 ?? S       0:00.00 /bin/sh ../../../../../../bk-deps g++ -c -o ogledit_view.o
26353 ?? IW      0:00.04 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
27935 ?? DE      0:00.01 (bash)
28366 ?? IWs     0:00.13 sshd: tanders@notty
29812 ?? IWs     0:00.03 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
2336 p0 R+      0:00.00 ps ax
21906 p0 Is      0:00.01 -bash
24304 p0 S       0:00.03 sh
697 00 IWs+     0:00.01 /usr/libexec/getty Pc console
639 E1 IWs+     0:00.01 /usr/libexec/getty Pc ttyE1
734 E2 IWs+     0:00.01 /usr/libexec/getty Pc ttyE2
762 E3 IWs+     0:00.01 /usr/libexec/getty Pc ttyE3
$

```

Gambar 5.11. Manajemen memori dan penjadwalan proses pada Unix.

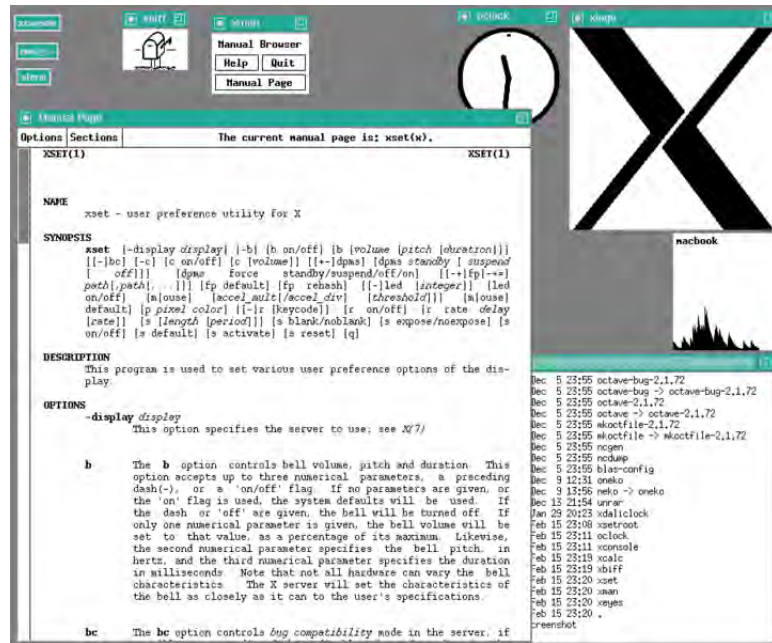
- **Development Environment:**

- *cc* —compiler untuk bahasa C
- *as* — machine-language assembler
- *ld* — linker, untuk menggabung file-file object
- *lib* — object-code libraries (diinstall di folder */lib* atau */usr/lib*) *libc*, kumpulan pustaka untuk bahasa C
- *make* — program untuk mengkompilasi kode program
- *include* — file-file header untuk pengembangan perangkat lunak dan menentukan standar interface
- *Other languages* — bahasa-bahasa pemrograman lain seperti Fortran-77, Free Pascal, dan lain-lain.

- **Commands:**

- *sh* —"Shell" untuk melakukan pemrograman berbasis CLI atau mengeksekusi perintah-perintah tertentu.
- *Utilities* — Sekumpulan perintah CLI yang berguna untuk fungsi-fungsi yang bermacam-macam, meliputi:
 - *System utilities* — Program-program untuk pengelolaan sistem seperti *mkfs*, *fsck*, dan lain-lain.
 - *User utilities* — Program-program untuk pengelolaan lingkungan kerja, seperti *passwd*, *kill*, dan lain-lain.

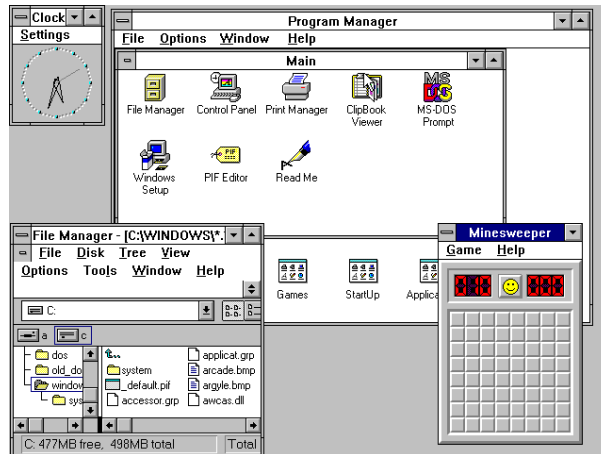
- *Document formatting* — Program untuk penyiapan dokumen seperti [nroff](#), [troff](#), [tbl](#), [eqn](#), [refer](#), dan [pic](#). Beberapa sistem Unix modern juga memasukkan aplikasi seperti [TeX](#) dan [Ghostscript](#).
- *Graphics* — Sistem Unix modern menyediakan [X11](#) sebagai sistem standard windowing dan [GUI](#).



Gambar 5.12. X windows system di UNIX.

5.3.3 Microsoft Windows

Microsoft Windows atau orang lebih sering menyebut Windows saja pada awalnya hanyalah add-on dari MS-DOS karena tingginya tuntutan pada sistem operasi yang berbasis GUI. Versi awal Windows berjalan di atas MS-DOS. Meski demikian Windows versi awal telah menunjukkan beberapa fungsi-fungsi yang umum dijumpai dalam sistem operasi, antara lain: memiliki tipe file executable tersendiri, memiliki driver perangkat keras sendiri, dan lain-lain.



Gambar 5.13. Windows versi 3.11.

Secara konsep sebenarnya Windows lebih banyak ditujukan bagi komputer personal. Pada awalnya Windows juga tidak mendukung konsep multi-tasking dan multi-user. Akomodasi terhadap jaringan atau fungsi-fungsi client-server juga tidak sekuat pada UNIX dan turunannya. Sehingga masalah yang sering muncul di sistem operasi Windows adalah masalah keamanan yang berhubungan dengan jaringan. Namun Windows memiliki kelebihan dari sisi kemudahan pemakaian. Pada versi yang terbaru (Windows Vista) konsep multi-user dan multi-tasking telah semakin matang. Selain itu tampilan GUI telah dirubah dengan banyak menggunakan efek tiga dimensi.

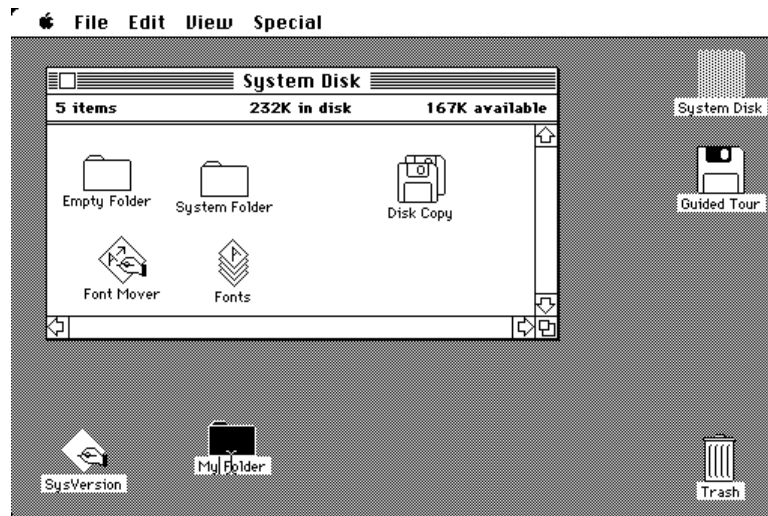


Gambar 5.14. Windows Vista.

5.3.4 Apple Mac OS

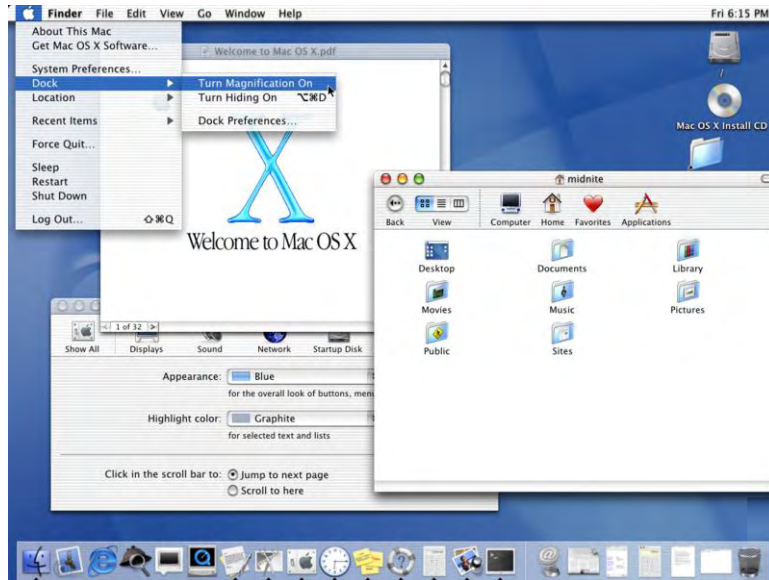
Seperti terlihat pada Gambar 5.10, Apple Mac OS merupakan turunan dari UNIX melalui jalur BSD (Berkeley Software Distribution). Oleh karena itu kekuatan dalam multi-tasking, multi-user, networking yang ada pada UNIX juga dimiliki oleh Mac OS. Mac OS adalah sistem operasi berbasis GUI. Apple merupakan pelopor dalam penggunaan GUI pada sistem operasi. Penggunaan icon, mouse dan beberapa komponen GUI merupakan sumbangan yang luar biasa bagi perkembangan sistem operasi berbasis GUI.

Versi awal dari Mac OS hampir secara penuh mengandalkan pada kemampuan GUI-nya dan sangat membatasi penggunaan CLI (Gambar 5.15). Meskipun sangat memudahkan namun ada beberapa kelemahan, antar lain: multi-tasking yang tidak berjalan sempurna, pengelolaan memori yang terbatas, dan konflik pada beberapa program yang ditanamkan. Memperbaiki sistem Mac OS kadang-kadang menjadi suatu pekerjaan yang sangat melelahkan.



Gambar 5.15. Mac OS versi awal.

Pada Mac OS X (versi terbaru), semua kelemahan pada versi lama telah coba dihilangkan. Multi-tasking telah berjalan dengan baik dan manajemen memori yang jauh lebih baik. Selain itu tampilan GUI-nya disebut-sebut sebagai yang terbaik di antara sistem operasi yang ada..



Gambar 5.16. Mac OS X.

5.3.5 Linux

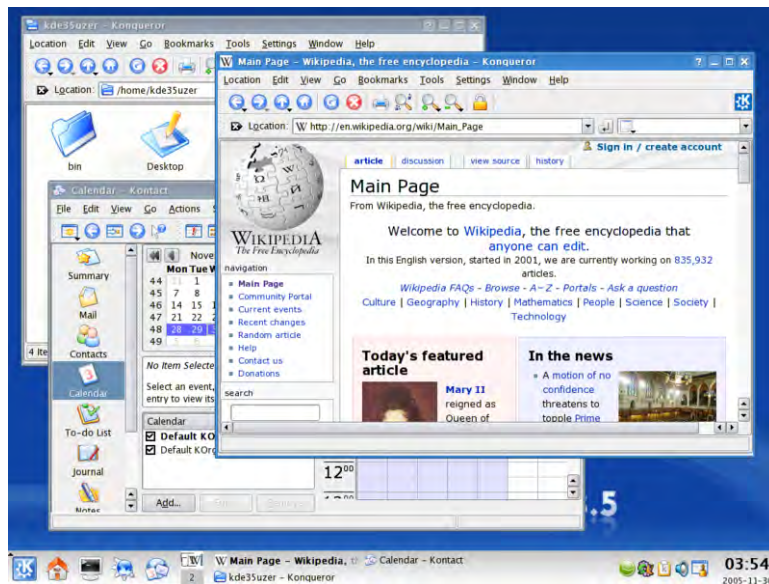
Linux sangat mirip dengan sistem-sistem UNIX, hal ini dikarenakan kompatibilitas dengan UNIX merupakan tujuan utama desain dari proyek Linux. Perkembangan Linux dimulai pada tahun 1991, ketika mahasiswa Finlandia bernama Linus Torvalds menulis Linux, sebuah kernel untuk prosesor 80386, prosesor 32-bit pertama dalam kumpulan CPU intel yang cocok untuk PC.

Dalam banyak hal, kernel Linux merupakan inti dari proyek Linux, tetapi komponen lainlah yang membentuk secara komplit sistem operasi Linux. Dimana kernel Linux terdiri dari kode-kode yang dibuat khusus untuk proyek Linux, kebanyakan perangkat lunak pendukungnya tidak eksklusif terhadap Linux, melainkan biasa dipakai dalam beberapa sistem operasi yang mirip UNIX. Contohnya, sistem operasi BSD dari Berkeley, *X Window System* dari MIT, dan proyek GNU dari *Free Software Foundation*.

Pembagian (sharing) alat-alat telah bekerja dalam dua arah. Sistem perpustakaan utama Linux awalnya dimulai oleh proyek GNU, tetapi perkembangan perpustakaannya diperbaiki melalui kerjasama dari komunitas Linux terutama pada pengalaman, ketidak efisienan, dan bugs. Komponen lain seperti GNU C Compiler, gcc, kualitasnya sudah cukup tinggi untuk dipakai langsung dalam Linux. Alat-alat administrasi network dibawah Linux berasal dari kode yang dikembangkan untuk 4.3BSD, tetapi BSD yang lebih baru, salah satunya FreeBSD, sebaliknya meminjam kode dari Linux, contohnya adalah perpustakaan matematika Intel *floating-point-emulation*.

Saat ini, Linux merupakan salah satu sistem operasi yang perkembangannya paling cepat. Kehadiran sejumlah kelompok pengembang, tersebar di seluruh dunia, yang selalu memperbaiki segala fiturnya, ikut membantu kemajuan sistem operasi Linux. Bersamaan dengan itu, banyak pengembang yang sedang bekerja untuk memindahkan berbagai aplikasi ke Linux (dapat berjalan di Linux).

Masalah utama yang dihadapi Linux dahulu adalah *interface* yang berupa teks (text based interface). Ini membuat orang awam tidak tertarik menggunakan Linux karena harus dipelajari terlebih dahulu dengan seksama untuk dapat dimengerti cara penggunaannya (tidak user-friendly). Tetapi keadaan ini sudah mulai berubah dengan kehadiran KDE dan GNOME. Keduanya memiliki tampilan desktop yang menarik sehingga mengubah persepsi dunia tentang Linux.



Gambar 5.17. Linux dengan desktop KDE.

5.4 MENJALANKAN SISTEM OPERASI

Mengenal sistem informasi saja tidak cukup. Bagi seorang yang bergerak dalam pemrograman perlu mengetahui secara lebih mendalam tentang bagaimana instalasi, booting dan menjalankan sistem operasi, dari pada pengguna biasa.

5.4.1 Instalasi

Instalasi :

Pemasangan perangkat lunak pada system computer.

Instalasi Sistem Operasi

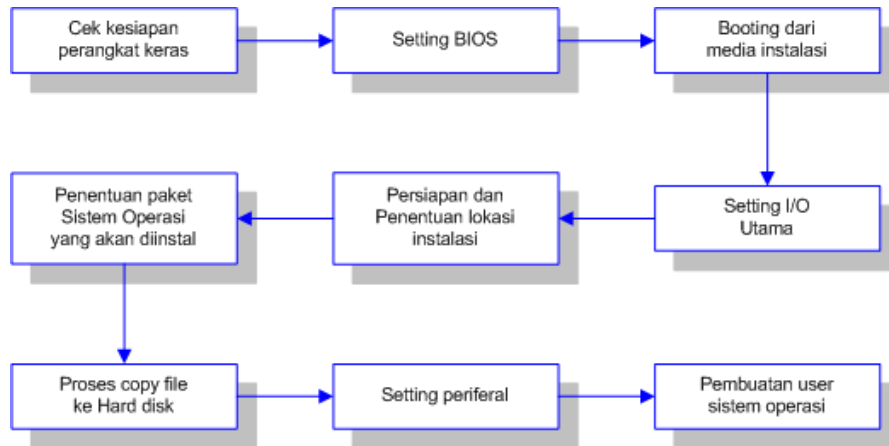
Pemasangan system operasi pada system computer

Sistem operasi akan dipasang terlebih dahulu dibanding perangkat lunak yang lain. Perangkat lunak yang lain baru bisa dijalankan setelah sistem operasi terinstal dengan benar.

Seperti telah dijelaskan sub bab 5.3, masing-masing sistem operasi memiliki ciri tersendiri. Demikian juga dengan proses instalasi sistem operasi. Proses instalasi sangat bergantung pada jenis sistem operasinya. Berdasarkan tampilan antarmukanya kita dapat membagi menjadi dua, yaitu yang berbasis GUI dan berbasis CLI. Proses instalasi berbasis GUI ada pada sistem operasi Microsoft Windows (GUI penuh pada versi Vista), Apple Mac OS ver X dan yang di atasnya, beberapa versi Linux seperti, Ubuntu dan turunannya (Xubuntu, Kubuntu, Edubuntu, dan lain-lain), Mandriva dan turunannya (PC Linux OS), dan Fedora versi terbaru. Sedangkan versi CLI ada pada Linux versi Slackware, Gentoo dan lain-lain.

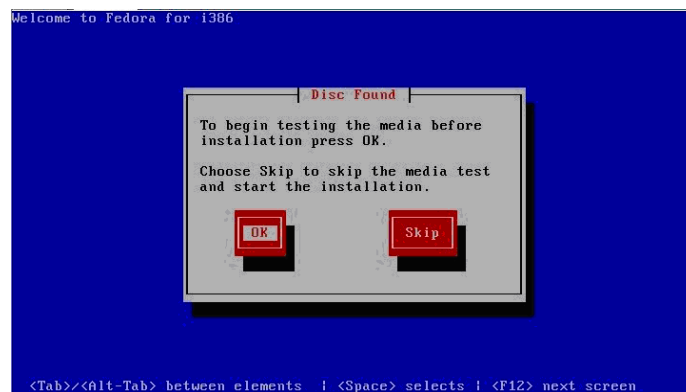
Proses instalasi juga dapat dibagi berdasarkan sumber instalasinya, yaitu bersumber dari media baik itu CD, DVD atau hard-disc dan yang bersumber dari network (jaringan). Proses instalasi dengan menggunakan media CD atau DVD merupakan metode yang paling umum digunakan. Pada bagian ini hanya akan dijelaskan tentang proses instalasi dengan sumber dari CD/DVD

Tahapan-tahapan dalam instalasi biasanya seperti terlihat pada Gambar 5.18. Tahapan-tahapan instalasi ini mungkin bervariasi antar sistem operasi. Namun secara umum tahapan dalam sistem operasi apapun tidak akan berbeda jauh.



Gambar 5.18. Tahapan-tahapan instalasi.

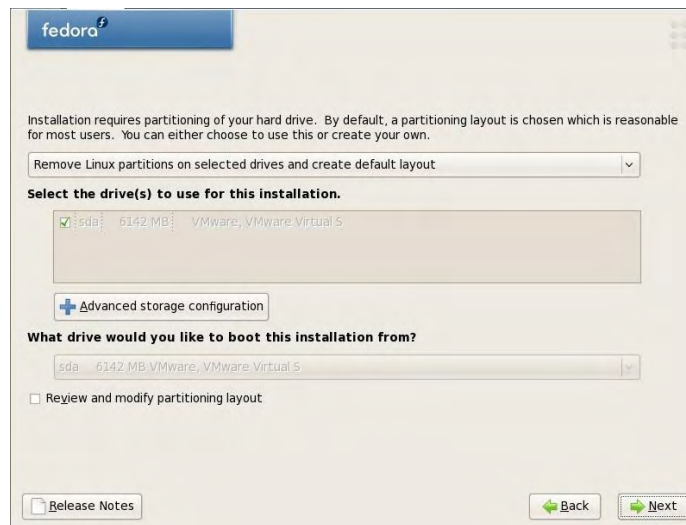
- Cek kesiapan perangkat keras. Tahap ini bertujuan untuk memastikan bahwa semua perangkat perangkat keras dan periferalnya terpasang dengan benar. Selain itu juga untuk melihat apakah spesifikasi perangkat keras komputer didukung oleh sistem operasi tersebut.
- Setting BIOS. Pada dasarnya tahapan ini adalah untuk mengkonfigurasi BIOS agar meletakkan media instalasi dalam urutan paling atas dalam prioritas booting.
- Booting dari media instalasi. Apabila setting BIOS berhasil dengan baik, maka komputer akan boot dari media instalasi. Pengertian booting dan tahapan-tahapan booting akan diuraikan pada sub bab 5.4. Gambar 5.19 merupakan screen-shot dari proses booting di bagian awal.



Gambar 5.19. Testing media instalasi.

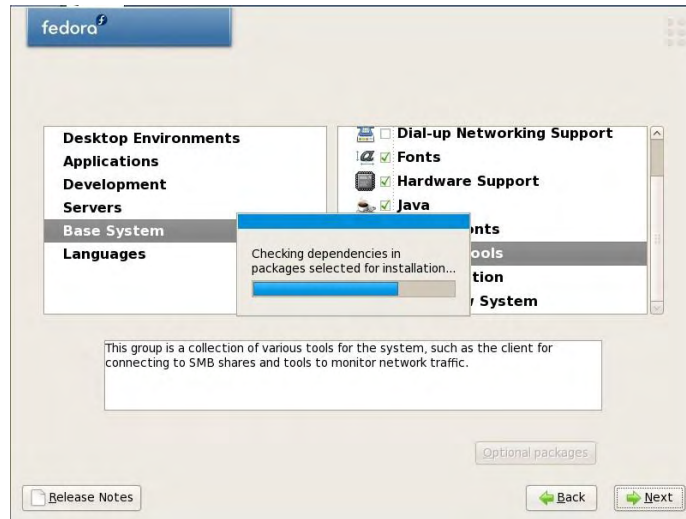
- Setting I/O utama. Tahapan ini bertujuan untuk mengatur agar perangkat input / output utama (mouse, keyboard dan video) dapat berjalan dengan baik ketika proses instalasi dilakukan.

- Persiapan dan penentuan lokasi instalasi. Media yang paling umum digunakan sebagai target instalasi adalah hard disk yang tertanam di komputer. Kita perlu mempersiapkan hard disk tersebut agar siap ditulis. Persiapan ini meliputi partisi hard disk (termasuk besarnya volume untuk masing-masing partisi) dan format partisi sesuai dengan sistem file yang disyaratkan oleh sistem operasi. Untuk Microsoft Windows, dapat menggunakan sistem file NTFS atau FAT32. Untuk linux dapat digunakan sistem file ext2, ext3, ReiserFS, dan XFS. Untuk Apple Mac OS X biasanya digunakan HFS+. Gambar 5.20 menunjukkan proses penentuan lokasi instalai pada proses instalasi Fedora Core 8.



Gambar 5.20. Proses penentuan target instalasi.

- Penentuan paket Sistem Operasi yang akan diinstal. Tahap ini kadang tidak diperlukan jika kita memilih instalasi secara default. Namun bila kita ingin menginstal sistem operasi agar sesuai dengan keinginan kita (custom installation) maka tahapan ini harus dilakukan. CD atau DVD instalasi, biasanya mempunyai paket-paket aplikasi yang dapat kita pilih ketika instalasi sistem operasi berjalan atau ketika proses instalasi telah selesai.
- Proses copy ke hard disk. Setelah penentuan paket aplikasi dilakukan, maka proses copy file instalasi ke hard disk dapat segera dilakukan. Gambar 5.21. merupakan contoh proses copy file sistem operasi.

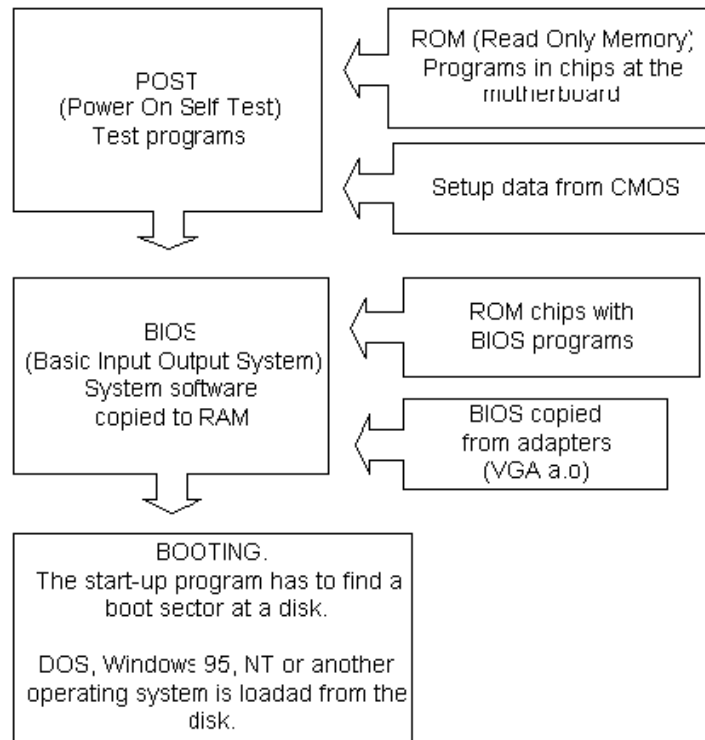


Gambar 5.21. Proses copy file pada Fedora.

- Setting peripheral lain. Tahapan ini bertujuan untuk menginstal driver bagi peripheral (kartu VGA, kartu suara, chipset motherboard dan lain-lain) pada suatu komputer agar dapat bekerja dengan optimal.
- Penentuan user. User adalah pengguna dari sistem operasi yang telah diinstal. Data dari user yang biasanya ditanyakan adalah user name dan password. Secara umum ada dua level pengguna, yaitu administrator dan user biasa. Administrator mempunyai hak pada semua bagian dari sistem operasi sedangkan user biasa mempunyai hak yang ditentukan oleh administrator.

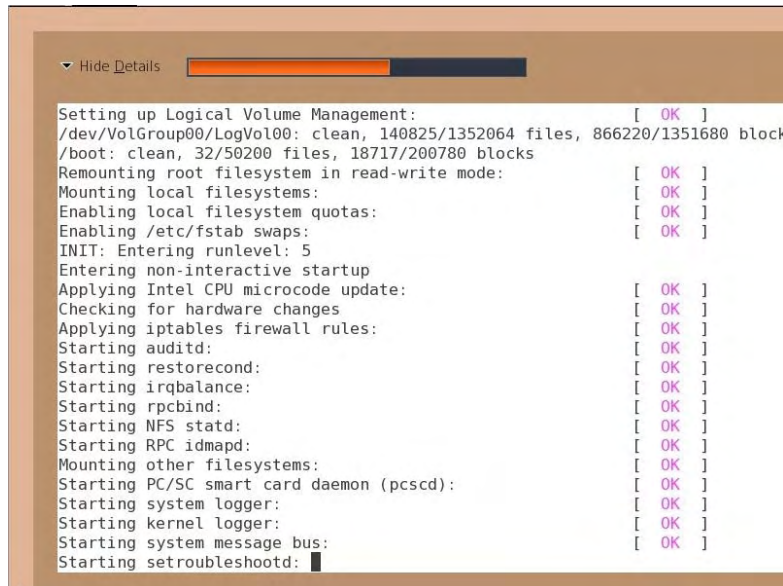
5.4.2 Booting

Booting adalah proses awal saat komputer dihidupkan. Proses awal booting dapat dijelaskan dengan menggunakan skema pada Gambar 5.22. Proses awal booting dimulai dari pembacaan dan eksekusi program yang tersimpan di ROM komputer dan data setup yang tersimpan dalam CMOS. Bagian ini disebut POST (Power On Self Test) apabila berhasil, maka perangkat lunak sistem BIOS yang berisi program BIOS dari ROM dan BIOS dari adapter (misalnya dari VGA) akan dimuat ke memori utama (RAM) dan dilanjutkan dengan pembacaan program start-up yang tersimpan di dalam boot sector hard disk. Dari sini barulah sistem operasi dimuat dari hard disk.



Gambar 5.22. Proses awal booting.

Pada sistem operasi seperti Microsoft Windows, kita tidak dapat melihat apa yang terjadi ketika sistem operasi dimuat (mulai dijalankan). Kita hanya disuguhi tampilan (biasanya logo) yang disebut sebagai boot-splash. Tetapi pada keluarga Linux, kita dapat memilih apakah proses jalannya sistem operasi ditampilkan atau tidak dengan mengkonfigurasi file boot-loader (biasanya menggunakan LILO atau Grub). Gambar 5.23 menunjukkan proses booting pada Linux Fedora.



```
▼ Hide Details
Setting up Logical Volume Management: [ OK ]
/dev/VolGroup00/LogVol00: clean, 140825/1352064 files, 866220/1351680 block
/boot: clean, 32/50200 files, 18717/200780 blocks
Remounting root filesystem in read-write mode: [ OK ]
Mounting local filesystems: [ OK ]
Enabling local filesystem quotas: [ OK ]
Enabling /etc/fstab swaps: [ OK ]
INIT: Entering runlevel: 5
Entering non-interactive startup
Applying Intel CPU microcode update: [ OK ]
Checking for hardware changes [ OK ]
Applying iptables firewall rules: [ OK ]
Starting auditd: [ OK ]
Starting restorecond: [ OK ]
Starting irqbalance: [ OK ]
Starting rpcbind: [ OK ]
Starting NFS statd: [ OK ]
Starting RPC idmapd: [ OK ]
Mounting other filesystems: [ OK ]
Starting PC/SC smart card daemon (pcscd): [ OK ]
Starting system logger: [ OK ]
Starting kernel logger: [ OK ]
Starting system message bus: [ OK ]
Starting setroubleshootd: [ OK ]
```

Gambar 5.23 Proses booting pada Linux Fedora

5.4.3 Perintah berbasis teks

Bagi banyak orang bekerja dengan perintah berbasis teks (CLI) ketika berhadapan dengan sistem operasi mungkin sangat menyulitkan karena harus menghafal perintah dan mengetikkan perintah tersebut serta tampilan yang tidak menarik. Namun sesungguhnya bekerja dengan memiliki keuntungan tersendiri, antara lain:

- eksekusi perintah relative lebih cepat.
- hemat dalam penggunaan sumberdaya (terutama CPU dan memori utama).
- tidak bergantung pada perangkat keras dengan spesifikasi tinggi (terutama pada VGA dan monitor).

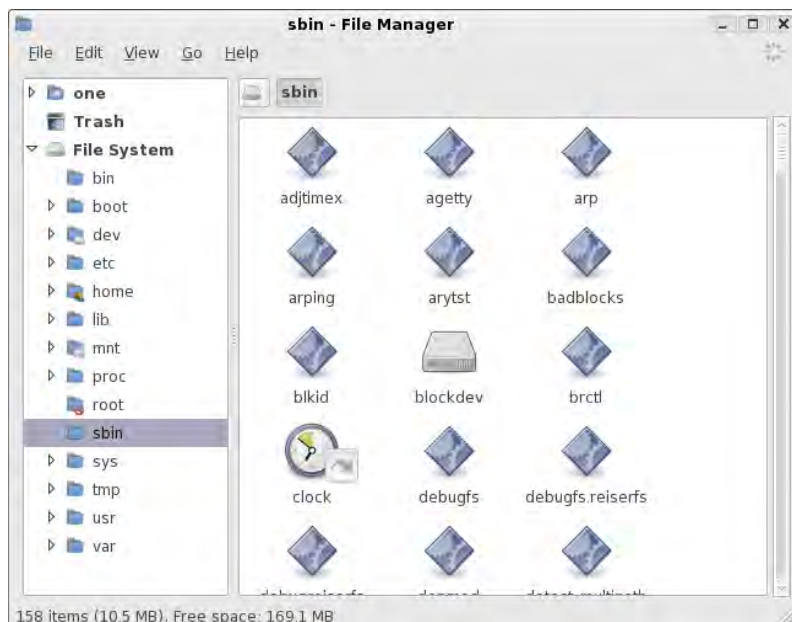
Pada sistem operasi Microsoft Windows dan Apple Mac OS X, mode CLI mungkin jarang digunakan, bahkan mungkin tidak pernah. Tetapi pada keluarga Linux dan Unix, mode CLI ini tetap merupakan bagian penting, terutama untuk administrasi sistem dan jaringan. Pada bagian ini kita akan membahas beberapa perintah yang sering digunakan pada mode CLI di sistem operasi Linux. Untuk menjalankan mode CLI ini dapat digunakan console atau terminal emulator yang tersedia di Linux, seperti *Konsole*, *xterm*, *aterm* dan lain-lain (Gambar 5.24).



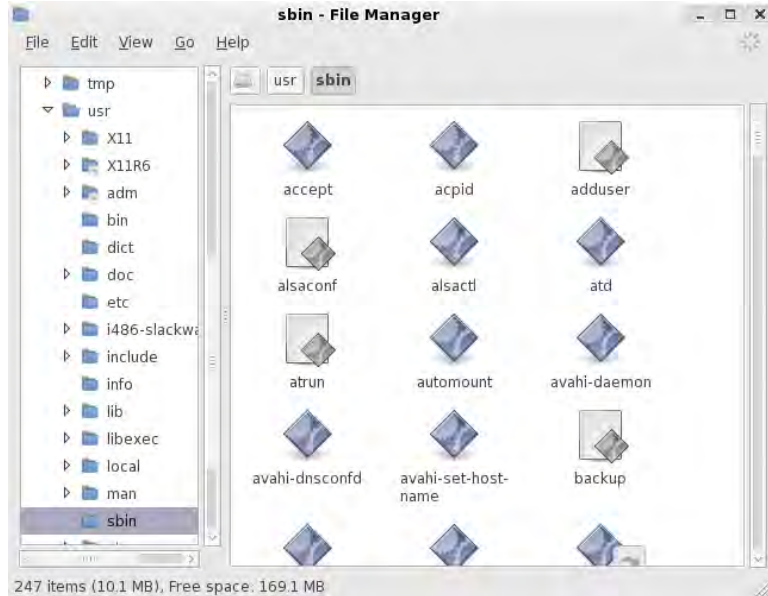
Gambar 5.24. Terminal sedang menjalankan mode CLI.

Ada dua kelompok utama dalam perintah-perintah mode CLI:

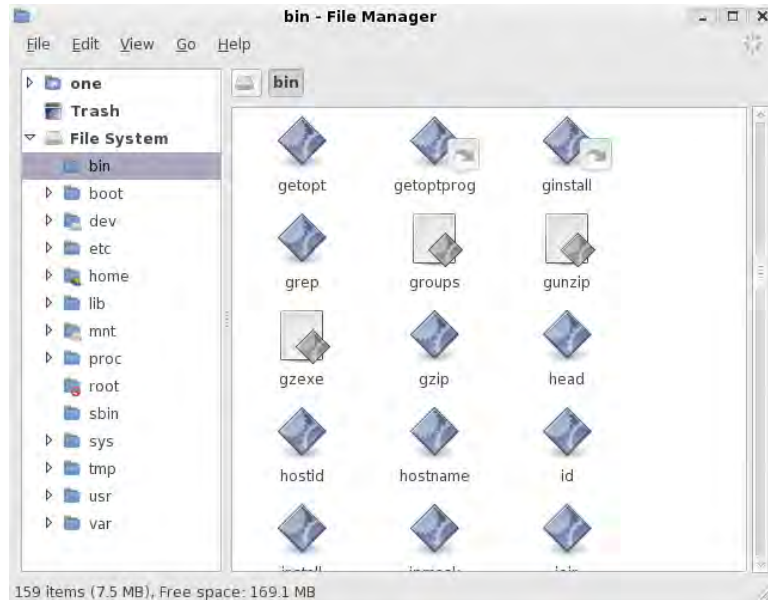
- Perintah yang berhubungan dengan administrasi sistem. Perintah-perintah yang termasuk dalam kelompok ini biasanya hanya dapat dilaksanakan oleh pengguna yang mempunyai hak sebagai administrator (root). Perintah-perintah yang termasuk kelompok ini biasanya tersimpan di direktori `/sbin` (Gambar 5.25) dan `/usr/sbin` (Gambar 5.26).
- Perintah untuk penggunaan biasa. Perintah ini dapat diakses oleh pengguna biasa. Perintah-perintah yang termasuk kelompok ini biasanya tersimpan di direktori `/bin` (Gambar 5.27) dan `/usr/bin` (Gambar 5.28).



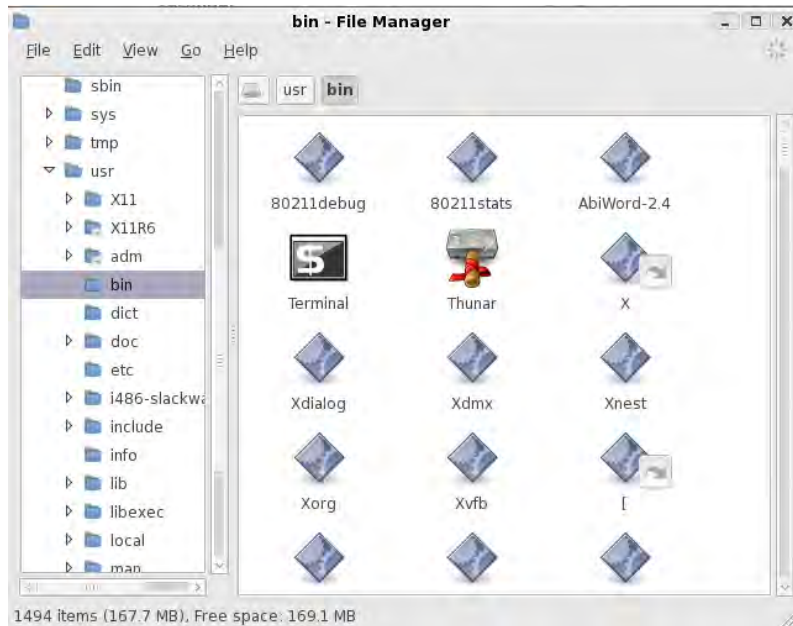
Gambar 5.25. Perintah-perintah pada direktori sbin.



Gambar 5.26. Perintah-perintah pada direktori /usr/sbin.



Gambar 5.27. Perintah-perintah pada direktori bin.



Gambar 5.28. Perintah-perintah pada direktori /usr/bin.

Berikut ini beberapa perintah-perintah penting dalam mode CLI.

- Menampilkan isi direktori

Untuk menampilkan isi direktori dapat digunakan perintah `ls` diikuti dengan argument lain. Beberapa contoh penggunaan dapat dilihat pada gambar 5.29.

```

one[~]$ ls /usr/
X11      dict          include      local      spool
X11R6    doc           info         man        src
adm      etc           lib          sbin      tmp
bin      i486-slackware-linux  libexec     share

one[~]$ ls -l
total 0
drwxr-xr-x 2 one users 85 2007-10-28 11:49 Desktop

one[~]$ ls -a
.          .bashrc      .gtkrc-2.0  .xsession-errors
..         .cache       .local      Desktop
.ICEauthority .config     .twmrc
.Xauthority  .dmenu      .vimrc
.Xresources  .gtkrc      .xinitrc

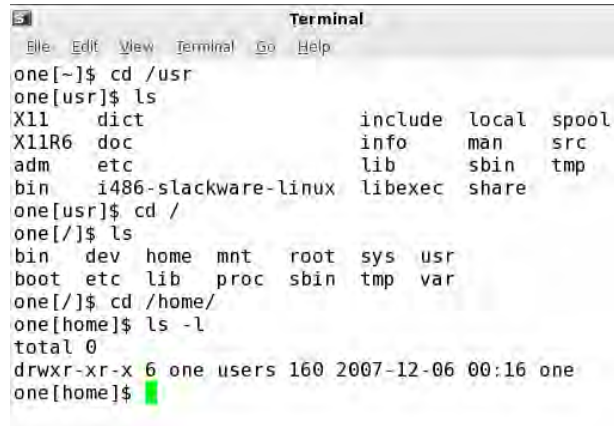
one[~]$ ls --color /usr/
X11      dict          include      local      spool
X11R6    doc           info         man        src
adm      etc           lib          sbin      tmp
bin      i486-slackware-linux  libexec     share
one[~]$

```

Gambar 5.29. Contoh penggunaan perintah `ls`.

- **Pindah direktori**

Berpindah direktori dapat dilakukan dengan perintah `cd` diikuti lokasi dimana kita mau berpindah. Beberapa contoh penggunaan dapat dilihat pada gambar 5.30.



```
one[~]$ cd /usr
one[usr]$ ls
X11      dict          include  local    spool
X11R6    doc           info     man      src
adm      etc           lib      sbin     tmp
bin      i486-slackware-linux libexec  share

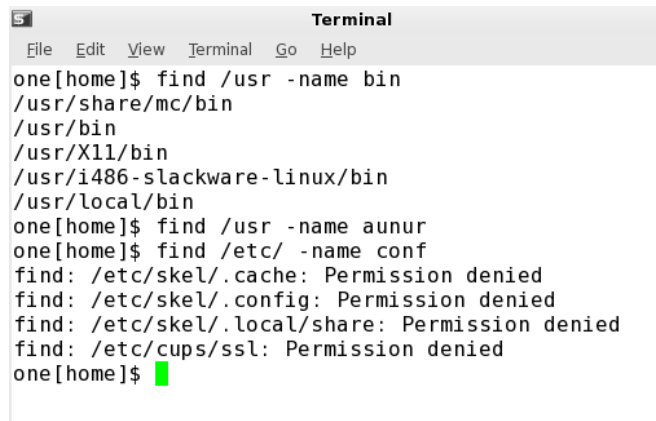
one[usr]$ cd /
one[/]$ ls
bin  dev  home  mnt  root  sys  usr
boot etc  lib  proc sbin  tmp  var

one[/]$ cd /home/
one[home]$ ls -l
total 0
drwxr-xr-x 6 one users 160 2007-12-06 00:16 one
one[home]$
```

Gambar 5.30. Contoh penggunaan perintah `cd`.

- **Mencari file**

Perintah `find` dapat digunakan untuk mencari file tertentu di lokasi yang ditentukan. Beberapa contoh penggunaan dapat dilihat pada gambar 5.31.

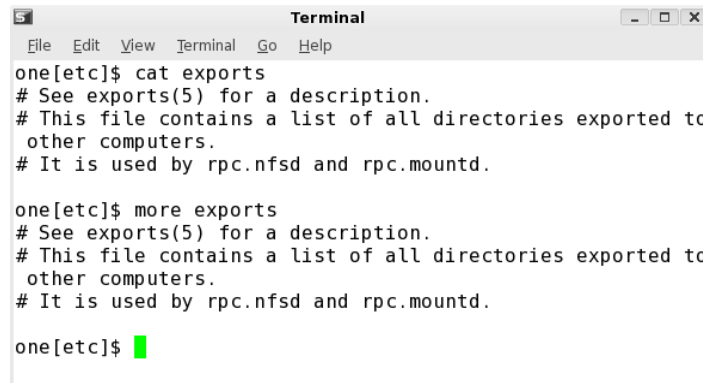


```
one[home]$ find /usr -name bin
/usr/share/mc/bin
/usr/bin
/usr/X11/bin
/usr/i486-slackware-linux/bin
/usr/local/bin
one[home]$ find /usr -name aunur
one[home]$ find /etc/ -name conf
find: /etc/skel/.cache: Permission denied
find: /etc/skel/.config: Permission denied
find: /etc/skel/.local/share: Permission denied
find: /etc/cups/ssl: Permission denied
one[home]$
```

Gambar 5.31. Contoh penggunaan perintah `find`.

- Menampilkan isi file

Untuk menampilkan isi file dapat digunakan perintah `more`, `less` atau `cat` diikuti dengan nama filenya. Beberapa contoh penggunaan dapat dilihat pada gambar 5.32.



```
Terminal
File Edit View Terminal Go Help
one[etc]$ cat exports
# See exports(5) for a description.
# This file contains a list of all directories exported to
other computers.
# It is used by rpc.nfsd and rpc.mountd.

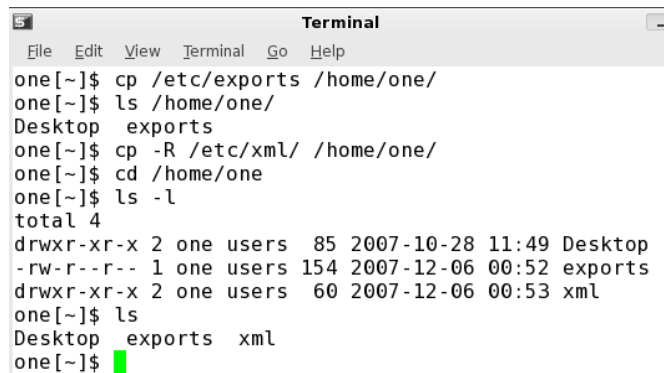
one[etc]$ more exports
# See exports(5) for a description.
# This file contains a list of all directories exported to
other computers.
# It is used by rpc.nfsd and rpc.mountd.

one[etc]$ █
```

Gambar 5.32. Contoh penggunaan perintah `cat` dan `more`.

- Menyalin file dan directory

Perintah `cp` bertujuan untuk menyalin file atau directory. Beberapa contoh penggunaan dapat dilihat pada gambar 5.33.

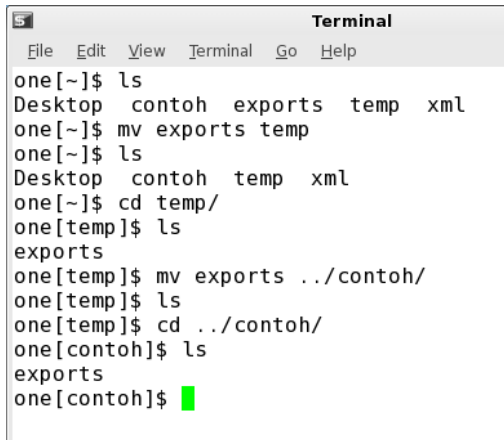


```
Terminal
File Edit View Terminal Go Help
one[~]$ cp /etc/exports /home/one/
one[~]$ ls /home/one/
Desktop  exports
one[~]$ cp -R /etc/xml/ /home/one/
one[~]$ cd /home/one
one[~]$ ls -l
total 4
drwxr-xr-x 2 one users 85 2007-10-28 11:49 Desktop
-rw-r--r-- 1 one users 154 2007-12-06 00:52 exports
drwxr-xr-x 2 one users 60 2007-12-06 00:53 xml
one[~]$ ls
Desktop exports xml
one[~]$ █
```

Gambar 5.33. Contoh penggunaan perintah `cp`.

- Memindahkan file

Untuk memindahkan file dapat digunakan perintah `mv`. Beberapa contoh penggunaan dapat dilihat pada gambar 5.34.

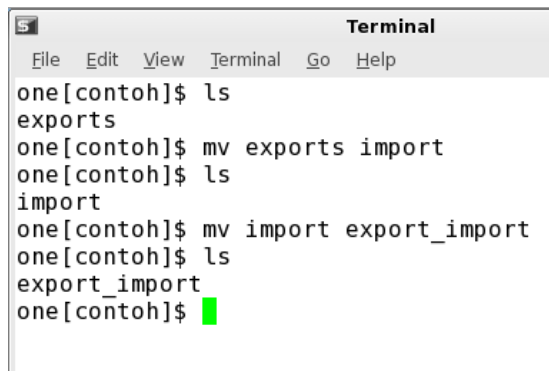
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The command history shows: one[~]\$ ls, Desktop contoh exports temp xml, one[~]\$ mv exports temp, one[~]\$ ls, Desktop contoh temp xml, one[~]\$ cd temp/, one[temp]\$ ls, exports, one[temp]\$ mv exports ../contoh/, one[temp]\$ ls, one[temp]\$ cd ../contoh/, one[contoh]\$ ls, exports, one[contoh]\$.

```
one[~]$ ls
Desktop  contoh  exports  temp  xml
one[~]$ mv exports temp
one[~]$ ls
Desktop  contoh  temp  xml
one[~]$ cd temp/
one[temp]$ ls
exports
one[temp]$ mv exports ../contoh/
one[temp]$ ls
one[temp]$ cd ../contoh/
one[contoh]$ ls
exports
one[contoh]$
```

Gambar 5.34. Contoh penggunaan perintah mv untuk memindahkan file.

- Mengganti nama file

Perintah mv dapat juga digunakan untuk mengganti nama file. Beberapa contoh penggunaan dapat dilihat pada gambar 5.35.

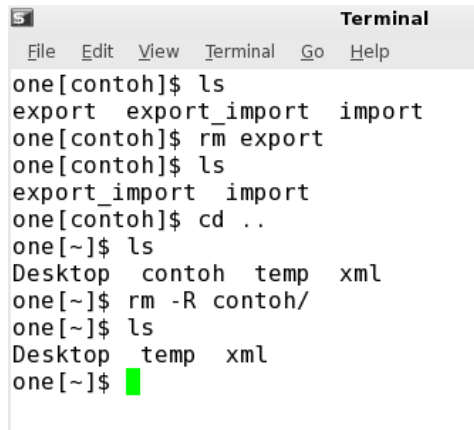
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The command history shows: one[contoh]\$ ls, exports, one[contoh]\$ mv exports import, one[contoh]\$ ls, import, one[contoh]\$ mv import export_import, one[contoh]\$ ls, export_import, one[contoh]\$.

```
one[contoh]$ ls
exports
one[contoh]$ mv exports import
one[contoh]$ ls
import
one[contoh]$ mv import export_import
one[contoh]$ ls
export_import
one[contoh]$
```

Gambar 5.35. Contoh penggunaan perintah mv untuk mengganti nama file.

- Menghapus file dan direktori

Perintah untuk menghapus file dan directory adalah rm. Beberapa contoh penggunaan dapat dilihat pada gambar 5.36.



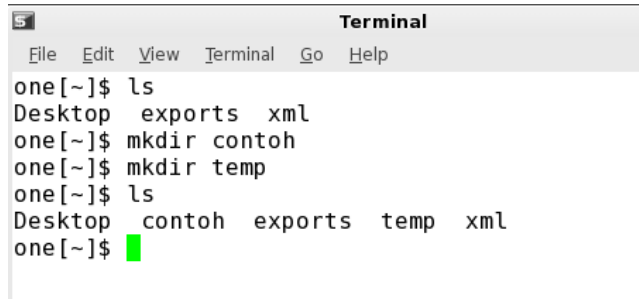
```
Terminal
File Edit View Terminal Go Help

one[contoh]$ ls
export export_import import
one[contoh]$ rm export
one[contoh]$ ls
export_import import
one[contoh]$ cd ..
one[~]$ ls
Desktop contoh temp xml
one[~]$ rm -R contoh/
one[~]$ ls
Desktop temp xml
one[~]$
```

Gambar 5.36. Contoh penggunaan perintah rm untuk menghapus file atau direktori.

- Membuat direktori

Perintah mkdir merupakan perintah untuk membuat directory baru. Beberapa contoh penggunaan dapat dilihat pada gambar 5.37.



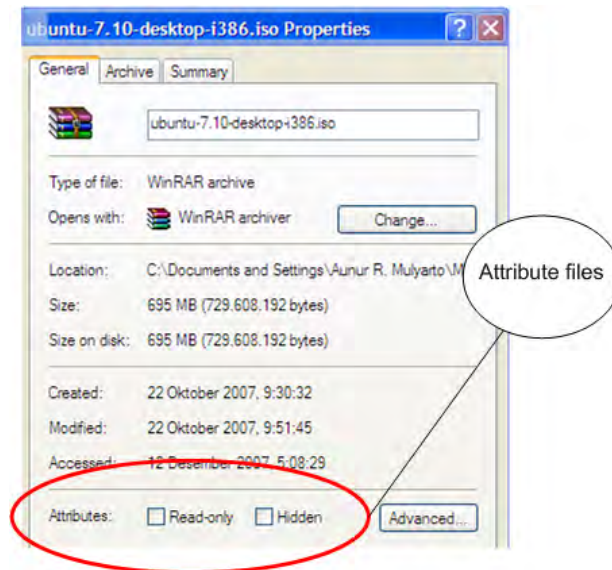
```
Terminal
File Edit View Terminal Go Help

one[~]$ ls
Desktop exports xml
one[~]$ mkdir contoh
one[~]$ mkdir temp
one[~]$ ls
Desktop contoh exports temp xml
one[~]$
```

Gambar 5.37. Contoh penggunaan perintah mkdir.

- Memahami hak akses file dan direktori

Pada sistem operasi windows, file dan direktori tidak memiliki file proteksi yang cukup karena file dan direktori hanya mempunyai attribute yang terbatas (Gambar 5.38).



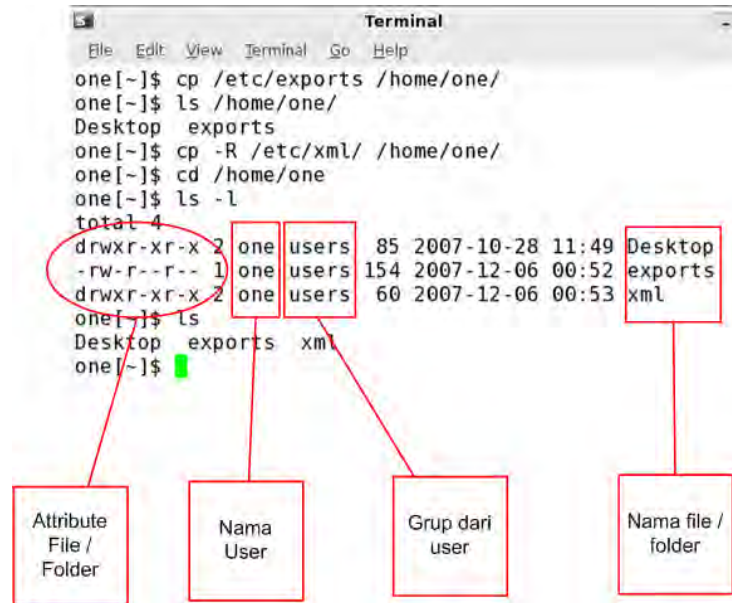
Gambar 5.38. Attribute file / folder pada Microsoft Windows

Pada gambar 5.38, tampak bahwa attribute file/direktori hanya memiliki attribute Read-only dan Hiden. Apabila kotak pilihan Read-only dipilih, maka file hanya akan dapat dibaca saja dan sebaliknya. Apabila kotak pilihan Hiden dipilih maka file/direktori akan disembunyikan sehingga tidak tampak ketika dicari dengan Windows Explorer.

Pada Unix dan keluarganya, termasuk linux, masalah attribute suatu file/direktori diatur dengan sangat ketat. Hal ini untuk meningkatkan keamanan dan memberi keleluasaan pada user untuk mengelola file dan direktori sesuai kebutuhannya.

Ada 4 bagian penting dalam suatu file / direktori, yaitu attribute, user atau (owner) pemilik dari file tersebut, grup dimana user sebagai anggota dan nama file/direktori.

Pada bagian attribute, ada penanda apakah itu direktori atau file biasa (ditandai dengan huruf d untuk direktori atau tanda - untuk file biasa). Selanjutnya ada sembilan kolom (karakter) yang menunjukkan hak akses terhadap file/direktori tersebut. Tiga kolom pertama menunjukkan hak akses owner, tiga kolom berikutnya hak akses grup dan tiga kolom terakhir adalah hak akses untuk other (user lain diluar owner dan anggota grup). Huruf **r** menunjukkan file/direktori bisa dibaca, **w** menunjukkan file/direktori bisa ditulis dan **x** menunjukkan file/direktori bisa dieksekusi. Perhatikan Gambar 5.39 berikut ini.



Gambar 5.39. Attribute file / direktori pada keluarga Unix

Gambar 5.35 menunjukkan hal sebagai berikut:

- **Desktop** dan **xml** adalah direktori karena mempunyai tanda d, sedangkan **exports** adalah file biasa karena bertanda -.
- **Desktop** dan **xml** mempunyai attribute **drwxr-xr-x** yang berarti owner (yaitu **one**) mempunyai hak untuk membaca, menulis dan mengeksekusi direktori ini. Sedangkan grup (yaitu **users**) mempunyai hak untuk membaca dan mengeksekusi saja. Other (user lain) juga mempunyai hak membaca dan mengeksekusi pada direktori ini.
- **exports** mempunyai attribute **-rw-r--r--** yang berarti owner (yaitu **one**) mempunyai hak untuk membaca dan menulis. Sedangkan grup dan other hanya mempunyai hak untuk membaca saja.

Untuk merubah attribute file/direktori dapat digunakan perintah seperti pada table berikut ini.

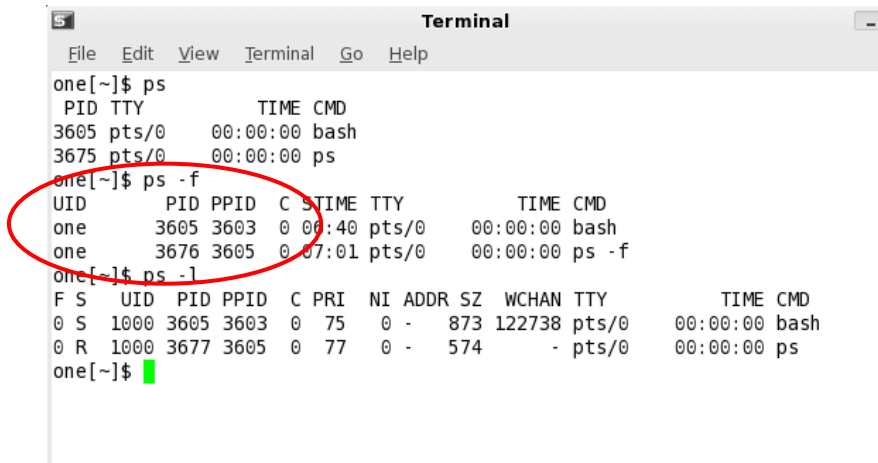
Tabel 5.1. Perintah yang berhubungan dengan pengelolaan file/direktori.

Perintah	Fungsi
chgrp [options] group file	Mengubah kepemilikan grup suatu file/direktori
chmod [options] owner file	Mengubah hak akses suatu file/direktori
chown [options] owner file	Mengubah kepemilikan owner suatu file/direktori

- Mengontrol proses

Proses merupakan bagian yang sangat penting dalam Linux sehingga perintah-perintah yang berhubungan dengan proses menjadi penting untuk diketahui.

Untuk melihat proses yang sedang berjalan dapat digunakan perintah `ps`. Perhatikan gambar 5.40 berikut ini.



```
one[~]$ ps
  PID TTY          TIME CMD
 3605 pts/0    00:00:00 bash
 3675 pts/0    00:00:00 ps
one[~]$ ps -f
  UID     PID  PPID  C  S  TIME  TTY          TIME CMD
  one     3605  3603  0  00:40 pts/0    00:00:00 bash
  one     3676  3605  0  07:01 pts/0    00:00:00 ps -f
one[~]$ ps -l
 F  S   UID     PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
 0  S   1000    3605  3603  0   75   0   -    873 122738 pts/0    00:00:00 bash
 0  R   1000    3677  3605  0   77   0   -    574   - pts/0    00:00:00 ps
one[~]$
```

Gambar 5.40. Eksekusi perintah `ps`.

Seperti terlihat pada gambar 5.40, perintah `ps` memiliki beberapa opsi (opsi selengkapnya dapat dilihat dengan mengetikkan perintah **`man ps`** pada terminal). Pada gambar tersebut ada dua proses yang sedang dijalankan oleh user **`one`** (lihat bagian **`UID`**) yaitu **`bash` dengan nomor proses (PID) 3605** dan **`ps -f` dengan PID 3676**.

Untuk menghentikan proses kita dapat menggunakan perintah **`kill`** diikuti nomor prosesnya (PID). Misalnya : **`kill 3605`** untuk menghentikan proses `bash`.

- Mengetahui ruang kosong pada disk

Kadang-kadang kita ingin mengetahui seberapa banyak sisa disk kita yang masih ada. Untuk mengetahui hal ini dapat digunakan perintah `df` seperti terlihat pada Gambar 5.41.

```
Terminal
File Edit View Terminal Go Help
one[~]$ df
Filesystem            1K-blocks      Used Available Use% Mounted on
aufs                  176800         3364   173436    2% /
one[~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
aufs                  173M   3.3M  170M   2% /
one[~]$
```

Gambar 5.41. Penggunaan perintah df.

Masih banyak sekali perintah yang digunakan dalam CLI di Linux. Pembaca disarankan untuk menggali dari sumber-sumber internet atau apabila ingin mengetahui kegunaan suatu perintah dapat digunakan perintah **man** diikuti nama perintah (Gambar 5.42).

```
Terminal
File Edit View Terminal Go Help
GREP(1)                                     GREP(1)

NAME
grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
grep [options] PATTERN [FILE...]
grep [options] [-e PATTERN] | -f FILE [FILE...]

DESCRIPTION
Grep searches the named input FILES (or standard input if no files are named, or the file name - is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

In addition, two variant programs egrep and fgrep are available. Egrep is the same as grep -E. Fgrep is the same as grep -F.

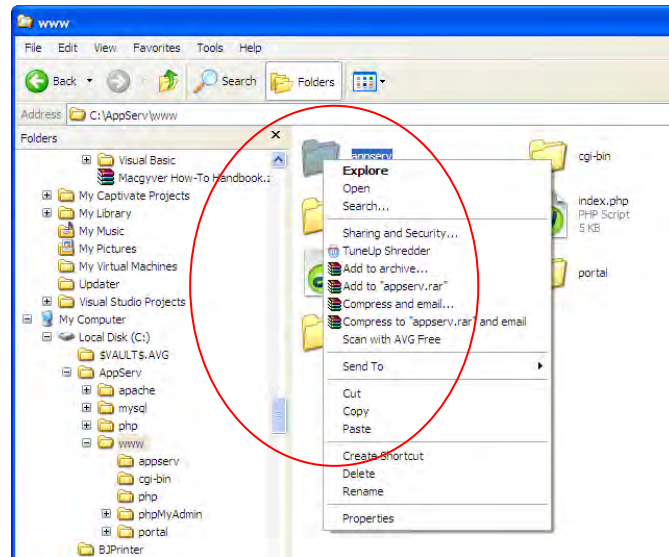
-- MOST: *stdin* (1,1) 0%
Press 'Q' to quit, 'H' for help, and SPACE to scroll.
```

Gambar 5.42. Contoh hasil eksekusi perintah man untuk melihat manual suatu perintah.

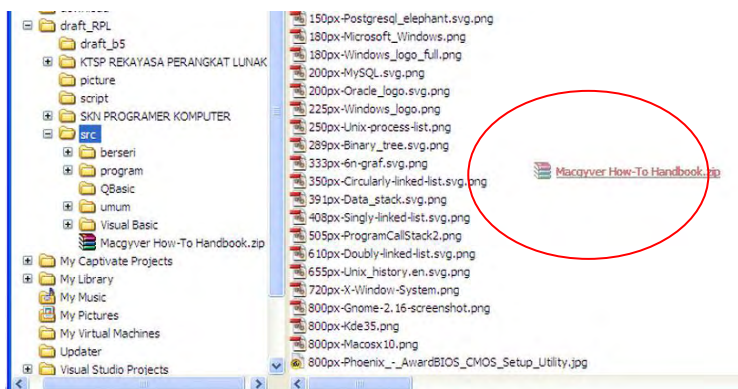
5.4.4 Bekerja dengan GUI

Secara umum bekerja dengan GUI pada sistem operasi sangat memudahkan pengguna karena pengguna hanya membutuhkan kerja mouse untuk melakukan sejumlah perintah. Mouse memiliki beberapa penggunaan, antara lain :

- klik satu kali digunakan untuk menunjuk satu file sebelum dilakukan operasi lain.
- Klik ganda (double-click) untuk mengeksekusi suatu perintah, misalnya membuka folder dan menjalankan file yang bisa dieksekusi.
- Klik kanan untuk membuka konteks menu (Gambar 5.43)
- Drag and drop untuk memindahkan file dari satu tempat ke tempat lain (Gambar 5.44).



Gambar 5.43. Membuka konteks menu dengan klik kanan.



Gambar 5.44. Drag and drop.

Jaringan Komputer

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian jaringan komputer.
 - Memahami topologi jaringan.
 - Memahami pembagian jaringan berdasarkan jangkauan.
 - Memahami protocol jaringan dan TCP / IP.
 - Menyebutkan perangkat keras jaringan
 - Menjalankan komputer dalam jaringan komputer.
-

Jaringan Komputer (network) :

system interkoneksi antar computer, terminal dan kanal komunikasi atau

kumpulan host yang dapat berkomunikasi satu dengan lainnya, yang seringkali bergantung pada pelayanan (service) dari beberapa host komputer yang dikhususkan fungsinya sebagai relay data antar komputer

Host biasanya berupa komputer, tapi tidak selalu. X terminal dan printer cerdas juga bisa dianggap sebagai suatu host. Sekelompok kecil host disebut sebagai situs.

6.1 TOPOLOGI

Topologi atau arsitektur jaringan :

Rencana induk yang didisain untuk menangani lingkungan jaringan komputer dengan prinsip terbuka, mudah, fleksibel dan efisien dengan menggunakan standar komunikasi hardware dan software.

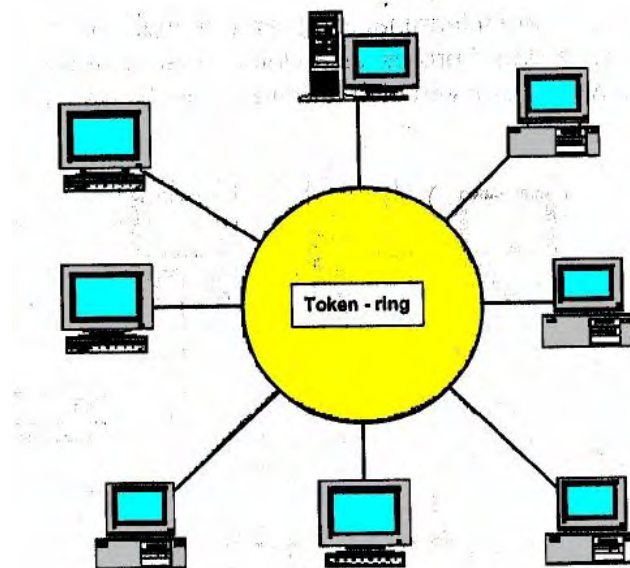
(O'Brien, 1999)

Istilah dari Topologi Jaringan mengacu pada organisasi spasial perangkat jaringan, pengkabelan fisik jaringan (Physical Routing) dan aliran paket data/paket data/informasi (messages) dari satu titik koneksi (titik koneksi) ke titik koneksi yang lain. Titik koneksi jaringan dapat berupa perangkat seperti sistim Komputer, printer, atau router, yang dihubungkan ke jaringan yang dapat mengirim dan menerima paket data/paket data.

Ada tiga topologi jaringan yang dikenal, yaitu topologi ring, bus dan star. Berikut penjelasannya.

- Topologi ring

Suatu jaringan disebut menggunakan topologi ring, jika setiap komputer dihubungkan secara melingkar. Jadi antar tetangga akan saling berhubungan. Pengiriman data dilakukan satu arah saja. Setiap perangkat bertindak sebagai pengirim dan penerima. Topologi ini jarang sekali digunakan.

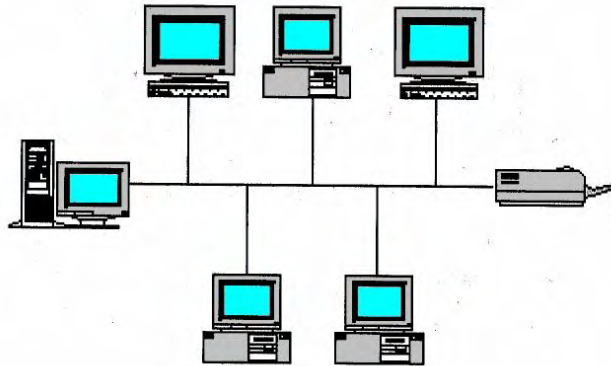


Gambar 6.1. Topologi ring.

Keuntungan dari topologi ring adalah: hemat kabel, dan dapat melayani lalu lintas data yang padat. Sedangkan kerugiannya adalah: peka kesalahan, pengembangan jaringan lebih kaku, kerusakan pada media pengirim/terminal dapat melumpuhkan kerja seluruh jaringan, dan lambat karena pengiriman menunggu giliran token.

- Topologi bus

Topologi ini pada prinsipnya adalah penggunaan bersama-sama satu kabel. Kabel ini biasa disebut sebagai backbone atau tulang punggung jaringan.



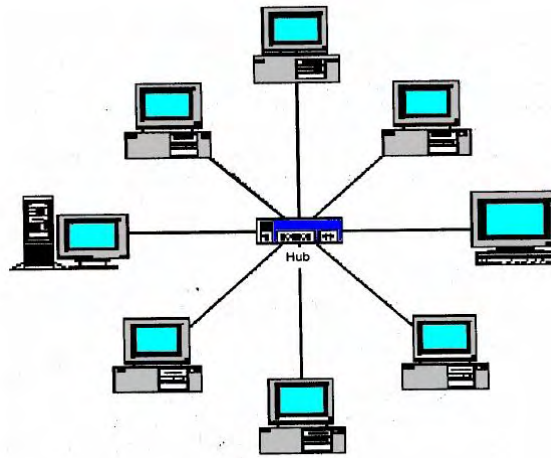
Gambar 6.2. Topologi bus.

Keuntungan jaringan komputer dengan topologi Bus adalah: hemat kabel, layout kabel sederhana, mudah dikembangkan, tidak butuh kendali pusat, dan penambahan maupun pengurangan terminal dapat dilakukan tanpa mengganggu operasi yang berjalan. Sedangkan kerugiannya adalah: deteksi dan isolasi kesalahan sangat kecil, kepadatan lalu lintas tinggi, keamanan data kurang terjamin, kecepatan akan menurun bila jumlah pemakai bertambah, dan diperlukan Repeater untuk jarak jauh.

- Topologi star

Topologi ini bersifat terpusat dengan menghubungkan semua komputer ke hub/switch. Hub menerima semua informasi / sinyal dan akan meneruskan ke alamat yang dituju. Topologi ini yang paling banyak digunakan dalam sistem jaringan.

Keuntungan dari topologi star adalah: paling fleksibel karena pemasangan kabel mudah, penambahan atau pengurangan stasiun sangat mudah dan tidak mengganggu bagian jaringan yang lain, dan kontrol yang terpusat karena memudahkan dalam deteksi dan isolasi kesalahan/kerusakan sehingga memudahkan pengelolaan jaringan. Sedangkan kerugiannya adalah: boros kabel, perlu penanganan khusus bundel kabel dan kontrol terpusat (HUB) jadi elemen kritis.



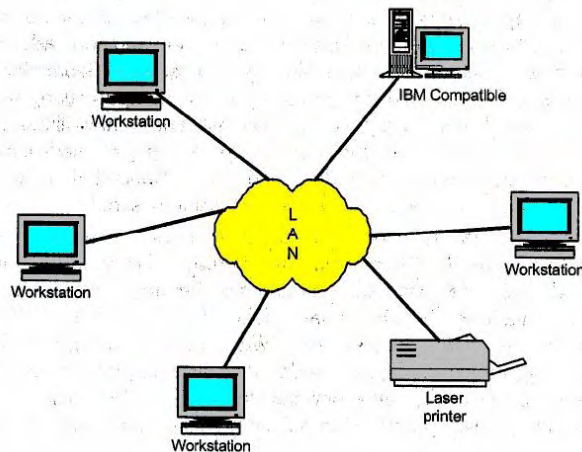
Gambar 6.3. Topologi Star.

6.2 JANGKAUAN

Berdasarkan jangkauannya jaringan komputer dapat dibedakan menjadi LAN, MAN dan WAN.

- Local Area Network (LAN)

LAN adalah jaringan komputer dan perangkat keras lainnya dalam lokasi yang tidak terlalu luas. Biasanya terbatas pada satu lingkungan kantor, sekolahan, suatu gedung dan lain-lain. LAN sudah lumrah diterapkan dalam kantor-kantor modern sekarang ini.

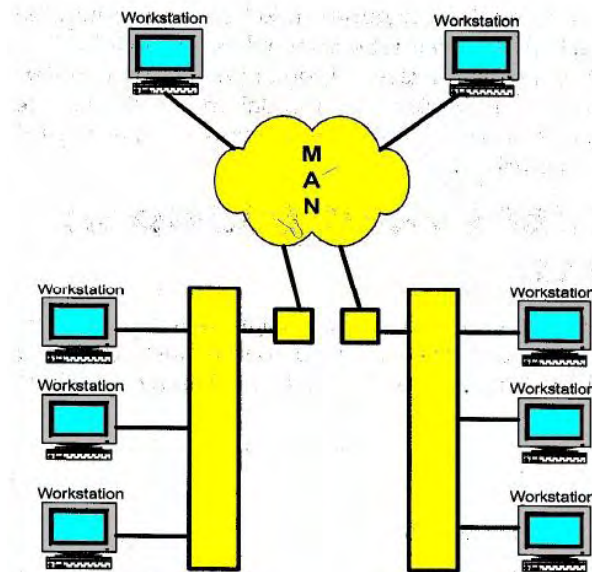


Gambar 6.4. Jaringan LAN.

LAN menggunakan berbagai macam media telekomunikasi, antara lain kabel telepon, kabel coaxial, atau perangkat wireless (tanpa kabel). Untuk dapat berkomunikasi di dalam jaringan, sebuah komputer dilengkapi dengan kartu jaringan. Sebagian besar LAN menggunakan satu atau lebih komputer yang memiliki hard disk dengan kapasitas besar, yang disebut file server atau network server. Server ini menggunakan sistem operasi jaringan yang mampu mengatur telekomunikasi dan penggunaan sumberdaya jaringan.

- Metropolitan Area Network (WAN)

MAN merupakan jaringan komputer dengan lingkup wilayah geografis yang lebih luas dari pada LAN. MAN, sesuai dengan namanya (Metropolitan) biasanya dalam lingkup satu kota. MAN menghubungkan antar LAN yang ada dalam satu kota. MAN dapat berupa akses public berkecepatan tinggi dan dapat melingkupi area seluas 80 km persegi.

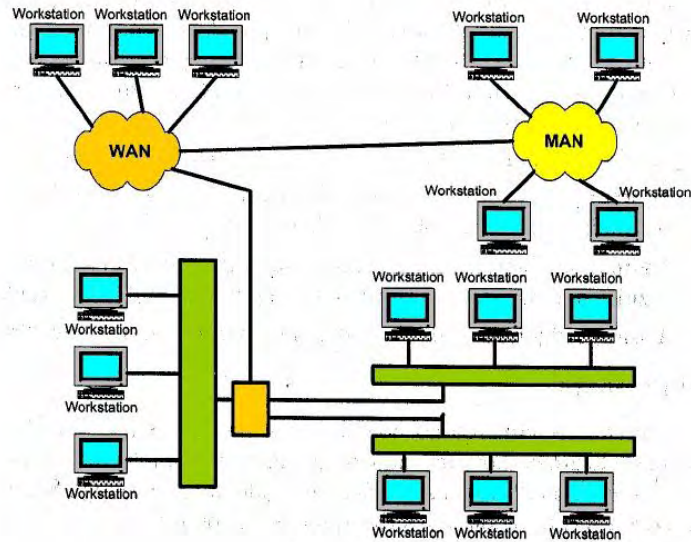


Gambar 6.5. Jaringan MAN.

- Wide Area Network (WAN)

WAN merupakan jaringan komputer dengan cakupan wilayah geografis paling luas. Dapat melewati batas kota, bahkan batas wilayah Negara sekalipun. WAN banyak digunakan perusahaan-perusahaan multi-

nasional untuk memperoleh informasi yang berhubungan dengan tenaga kerja, konsumen, dan organisasi lain di kota, atau Negara lain.



Gambar 6.6. Jaringan WAN

6.3 PROTOCOL DAN TCP/IP

Protocol :

Sekumpulan aturan dan prosedur standar untuk mengendalikan komunikasi di dalam suatu jaringan.

Adanya platform yang berbeda dalam dunia komputer membutuhkan suatu bahasa yang dimengerti oleh semua platform. Sebagai contoh ada komputer yang menggunakan sistem operasi Windows, Linux, Solaris dan ada juga yang menjalankan Mac OS. Semua komputer ini harus dapat berkomunikasi satu sama lain. Untuk itu dibutuhkan bahasa (protocol) yang sama sebagai media. TCP/IP adalah protocol yang disepakati.

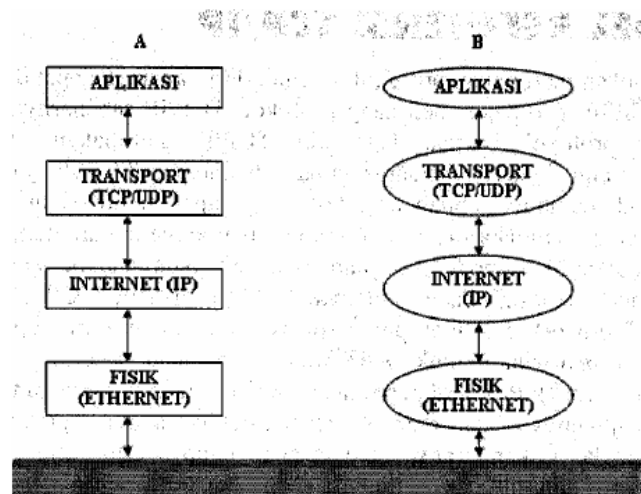
TCP/IP (Transmission Control Protocol / Internet Protocol) :

Sekumpulan protokol jaringan komputer terbuka dan bisa terhubung ke berbagai perangkat keras dan perangkat lunak.

Kebanyakan dari service yang berasosiasi secara normal dengan internet mengirimkan melalui TCP/IP. Service ini meliputi:

- transfer file melalui File Transfer Protocol (FTP)
- login remote melalui protocol Telnet
- distribusi surat elektronik melalui Simple Mail Transfer Protocol (SMTP)
- mengakses halaman web melalui Hypertext Transfer Protocol (HTTP)

TCP IP bergabung menjadi jaringan pribadi bersama membentuk internet dan World Wide Web. Skema Layer TCP/IP dapat dilihat pada gambar 6.7.



Gambar 6.7. Skema Layer TCP/IP.

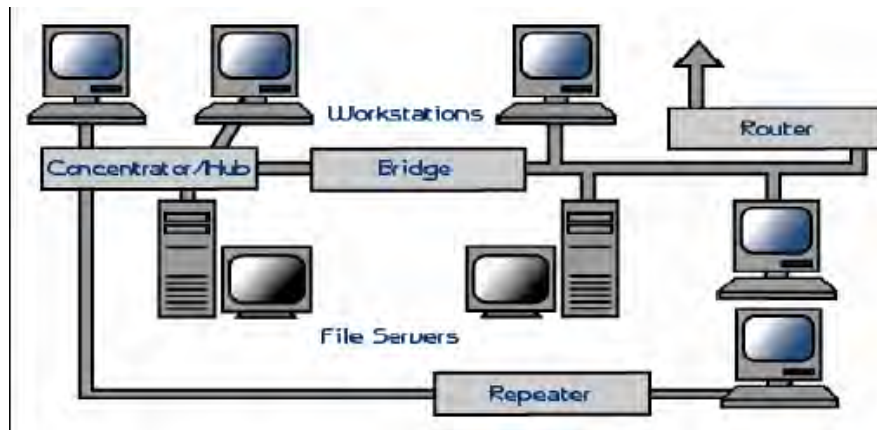
Koneksi pada jaringan ditangani oleh *host* yang dikhususkan sebagai *gateway* yang menangani paket yang masuk dan keluar dengan mengkopinya antara dua *ethernet* dan kabel optik. *Gateway* akan bertindak sebagai *forwarder*. Tata kerja dengan mengirimkan data ke sebuah *remote host* disebut *routing*, dan paket yang dikirim seringkali disebut sebagai *datagram* dalam konteks ini. Untuk memfasilitasi hal ini, pertukaran *datagram* diatur oleh sebuah protokol yang independen dari perangkat keras yang digunakan, yaitu IP (*Internet Protocol*).

Alamat IP terbagi atas 4 kelompok 8 *bit* nomor yang disebut oktet untuk memudahkan pembacaan. Sebagai contoh `quark.physics.groucho.edu` memiliki alamat IP `0x954C0C04`, yang dituliskan sebagai `149.76.12.4`. Format ini seringkali disebut notasi *quad bertitik*. Alasan lain untuk notasi ini adalah

bahwa alamat IP terbagi atas nomor jaringan, yang tercantum dalam oktet pertama, dan nomor *host*, pada oktet sisanya. Ketika mendaftarkan alamat IP ke NIC, anda tidak akan diberikan alamat untuk tiap *host* yang anda punya. Melainkan, anda hanya diberikan nomor jaringan, dan diizinkan untuk memberikan alamat IP dalam rentang yang sudah ditetapkan untuk tiap *host* sesuai dengan keinginan anda sendiri.

6.4 PERANGKAT KERAS JARINGAN

Hardware (Perangkat keras) yang dibutuhkan untuk membangun sebuah jaringan komputer yaitu, Komputer, Card Network, Hub, dan segala sesuatu yang berhubungan dengan koneksi jaringan seperti: Printer, CDRom, Scanner, Bridges, Router dan lainnya yang dibutuhkan untuk process transformasi data didalam jaringan, seperti terlihat pada gambar 6.8.



Gambar 6.8. Perangkat keras jaringan.

- NIC (Network Interface Controller)



Gambar 6.9. Network Interface Card

Network Interface Controller (NIC) biasa juga disebut sebagai Network Interface Card atau Network Card saja. Berfungsi untuk :

- Menghubungkan sebuah titik koneksi kedalam jaringan
- Mampu menyediakan MAC (Media Access Control) dan fungsi meneruskan paket data
- Bertindak sebagai sebuah bridge antara bus sistem komputer dan sebuah LAN

- Router



Gambar 6.10. Cisco Router

Router berfungsi untuk :

- Menghubungkan dua atau lebih jaringan
- Meneruskan paket antara network sesuai yang dibutuhkan
- Membuat pemilihan yg tepat diantara router alternatif

- Switches



Gambar 6.11. Switch

Switches berfungsi untuk :

- Menghubungkan jaringan atau titik koneksi jaringan untuk membentuk jaringan virtual pada sebuah basis perpaket
- Secara cepat memetakan paket menggunakan hardware yang berbasis switching

- Hub



Gambar 6.12. Hub

Hub berfungsi untuk :

- Bertindak sebagai sebuah titik pusat koneksi untuk pemasangan LAN
- Mengimplementasikan topologi jaringan logis
- Biasanya menghubungkan sebuah LAN ke sebuah WAN

6.5 BEKERJA DALAM KOMPUTER JARINGAN

Bekerja dalam komputer yang terhubung ke jaringan, saat ini bukanlah sesuatu yang aneh. Hampir semua tempat yang memiliki banyak komputer, selalu menggunakan jaringan sebagai sarana berkomunikasi. Oleh karena itu

pengetahuan dasar bagaimana dapat bekerja dalam komputer yang terhubung ke jaringan menjadi sangat penting.

6.5.1 Persiapan

Ada tiga hal penting yang harus dipersiapkan dalam koneksi ke jaringan komputer, yaitu perangkat keras, perangkat lunak dan akses ke jaringan.

- Perangkat keras

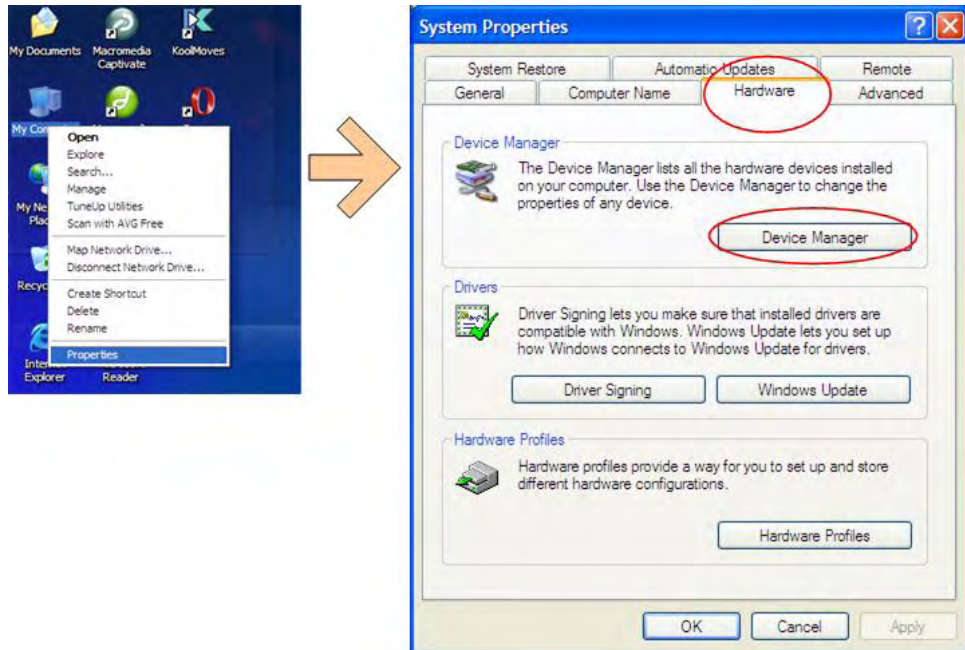
Kebutuhan perangkat keras sangat bergantung pada tipe koneksi jaringan yang akan digunakan. Untuk koneksi ke jaringan LAN maka kebutuhan utama adalah NIC yang telah terpasang dengan baik dan telah terinstal driver yang sesuai dan kabel jaringan. Untuk koneksi ke jaringan dengan cara dial-up, dibutuhkan modem dan kabel telepon analog.

Kita dapat melihat apakah perangkat keras jaringan (NIC, modem atau yang lainnya) sudah terinstall dengan benar dengan memeriksa pada daftar perangkat keras yang dikenali oleh komputer.

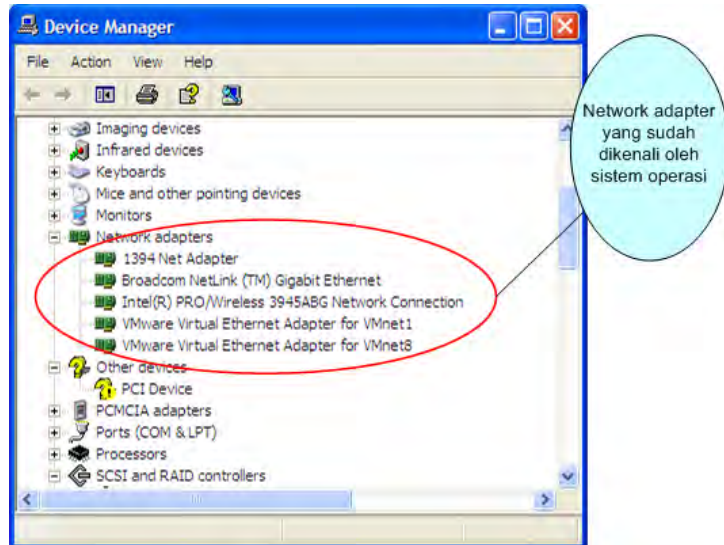
Pada sistem operasi Windows kita dapat melihat dengan cara klik kanan pada icon My Computer di desktop kemudian pilih Properties pada menu yang muncul (Gambar 6.13). Pada jendela System Properties pilih tab Hardware kemudian klik pada Device Manager (Gambar 6.13), sehingga akan muncul jendela Device Manager (Gambar 6.14).

Pada Gambar 6.14 terlihat bahwa network adapter yang digunakan oleh komputer sudah dikenali dengan baik. Apabila kita menemukan gambar tanda tanya pada suatu perangkat lunak berarti perangkat keras tersebut belum dikenali dengan baik (lihat Gambar 6.14).

Pada sistem operasi Linux dan keluarganya kita dapat memeriksa apakah perangkat keras sudah dikenali atau tidak dengan cara mengetikkan perintah **lspci** (Gambar 6.15) dan **ifconfig** (Gambar 6.16).



Gambar 6.13. Membuka system properties.



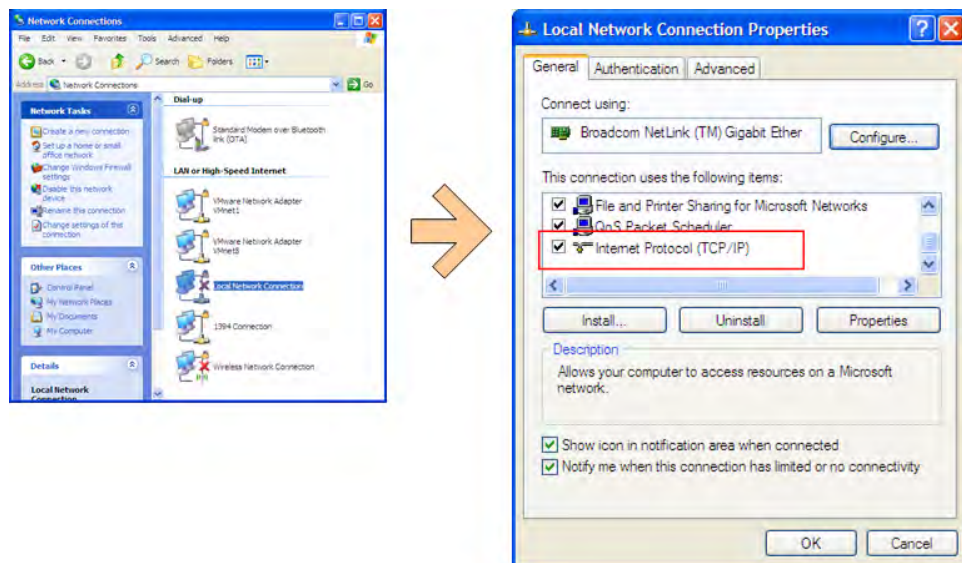
Gambar 6.14. Device manager.

```
root@one# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443 (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443 (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4
00:07.2 USB Controller: Intel Corporation 82371AB/EB/MB PIIX4
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0f.0 VGA compatible controller: VMware Inc [VMware SVGA II] PCI Display Adapter
00:10.0 SCSI storage controller: LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)
00:11.0 Ethernet controller: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10)
00:12.0 Multimedia audio controller: Ensoniq ES1371 [AudioPCI-97] (rev 02)
root@one#
```

Gambar 6.15. Output perintah lspci untuk memeriksa network adapter..

- o Perangkat lunak

Perangkat lunak utama, selain sistem operasi adalah apakah paket TCP/IP sudah terinstall dengan benar pada komputer. Pada sistem operasi windows dapat dilakukan dengan double klik pada tipe koneksi, kemudian setelah jendela properties muncul cek apakah sudah ada TCP/IP yang sudah terinstal (Gambar 6.16).



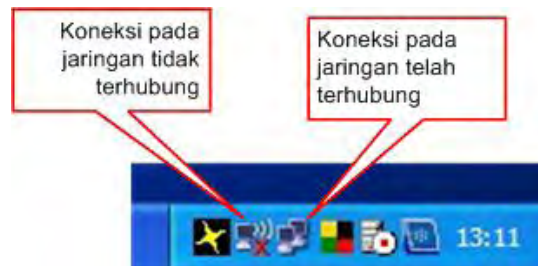
Gambar 6.16. Memeriksa protocol TCP/IP.

- Akses ke jaringan

Akses jaringan ini berhubungan dengan hak atau kewenangan kita dalam jaringan komputer yang akan kita masuki, terutama pemberian alamat IP (IP Address) dan password untuk masuk ke jaringan. Pada jaringan yang menerapkan DHCP maka kita tidak perlu khawatir karena no IP akan diberikan langsung ketika komputer berhubungan ke jaringan. Apabila tidak menggunakan DHCP maka kita harus memberikan no IP static yang diberikan oleh administrator jaringan pada komputer.

6.5.2 Konfigurasi koneksi jaringan

Pada sistem operasi modern sekarang ini koneksi ke jaringan bukan pekerjaan yang menyulitkan karena hampir semua koneksi telah dijalankan otomatis oleh sistem operasi. Pada jaringan LAN yang menggunakan DHCP, komputer yang menggunakan sistem operasi Windows (versi 2000 dan yang lebih baru) maupun Linux akan secara otomatis terkoneksi ke jaringan dan memperoleh no IP *dynamic*. Pada sistem operasi Windows, untuk memeriksa apakah komputer sudah tersambung ke jaringan kita bisa melihat pada *systray (notification area)* yang terletak di bagian kanan bawah desktop (Gambar 6.17).



Gambar 6.17. Kondisi koneksi jaringan.

6.5.3 Berbagi file, printer, dan sumber daya lain

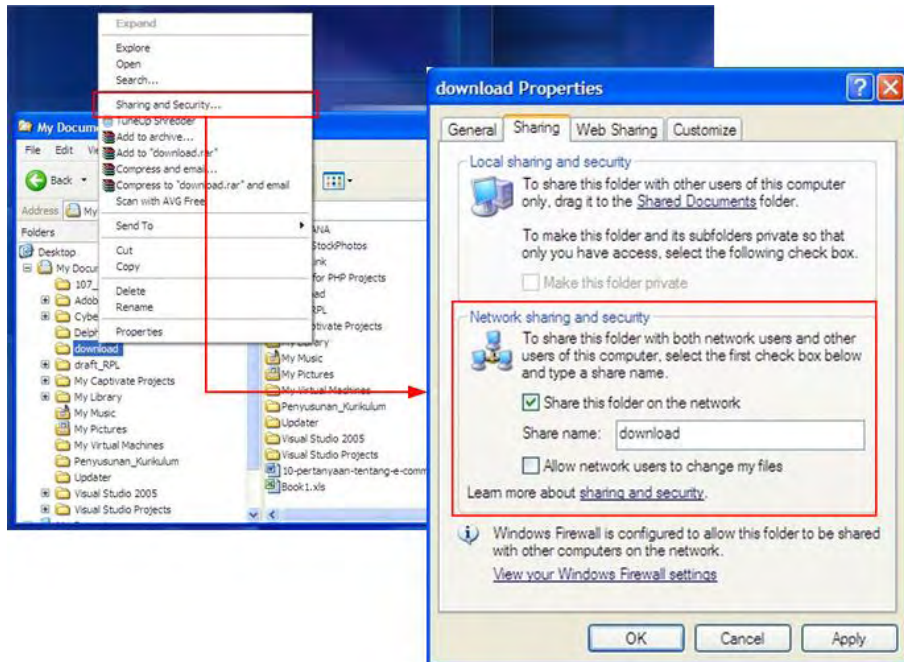
Kelebihan utama dari jaringan komputer adalah penggunaan secara bersama berbagai macam sumber daya, seperti: file, printer, media perekam (CD-RW atau DVD-RW), scanner dan lain-lain. Pada bagian berikut ini akan dijelaskan bagaimana berbagi file dan printer. Sumber daya yang lain dapat digunakan secara bersama-sama dengan cara yang tidak jauh berbeda dengan file dan printer.

- Berbagi file

Direktori atau file yang ada pada komputer kita dapat diatur agar dapat digunakan oleh komputer lain di dalam jaringan. Demikian pula

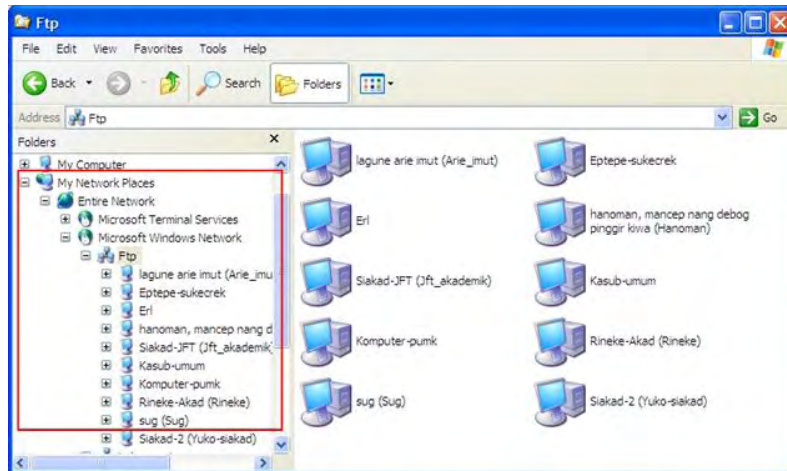
sebaliknya kita dapat menggunakan direktori atau file pada komputer lain di jaringan.

Untuk membagi (share) direktori atau file kita dapat menempuh cara berikut. Buka **Windows Explorer**, kemudian klik kanan pada direktori atau file yang akan kita share dan pilih **Sharing and Security**. Setelah muncul jendela **properties**, pilih tab **Sharing** dan pada bagian **Network sharing and security** cek pada **Share this folder on the network** dan beri nama untuk direktori yang di-share (Gambar 6.18).



Gambar 6.18. Mengatur file sharing.

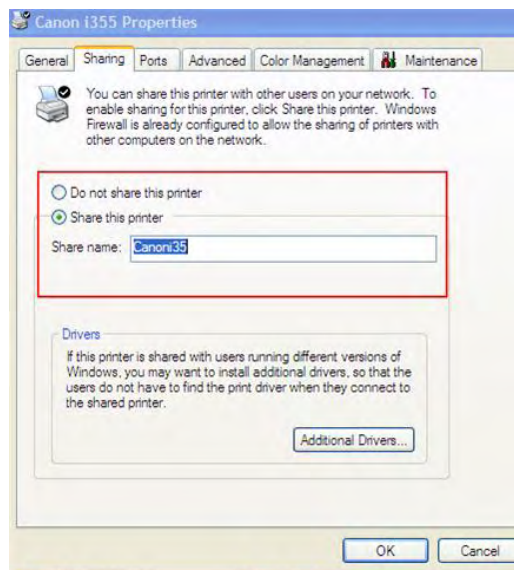
Untuk bisa mengakses direktori atau file di komputer lain, kita bisa membuka Windows Eksplorer kemudian klik pada My Network Places -> Entire Network -> Microsoft Windows Network. Kita akan mendapati tampilan seperti pada Gambar 6.19. Dari sini kita bisa melihat sumber daya apa yang dibagi pada masing-masing komputer yang terhubung ke jaringan dengan cara double klik pada nama komputer.



Gambar 6.19. Menjelajah komputer yang ada di jaringan.

o Berbagi printer

Untuk dapat berbagi printer yang ada di komputer, cara yang hampir sama dapat kita lakukan. Pertama kita buka jendela Printers and Faxes dengan cara Start -> Settings -> Printers and Faxes. Setelah jendela terbuka, klik kanan pada printer yang akan di share dan pilih Sharing Setelah jendela properties printer terbuka, pilih tab Sharing dan klik pada Share this printer serta beri nama (Gambar 6.20).



Gambar 6.20. Printer sharing.

Dasar-Dasar Pemrograman

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian program, programming language, pemrogramming, dan programmer.
 - Menyebutkan klasifikasi bahasa pemrograman.
 - Memahami prinsip kerja interpreter dan compiler
 - Menerapkan tahapan-tahapan pemrograman dalam penyelesaian masalah.
 - Mengetahui beberapa teknik pemrograman.
 - Memilih bahasa pemrograman yang sesuai untuk berdasarkan criteria utama.
-

7.1 PROGRAM DAN BAHASA PEMROGRAMAN

Program :

Kumpulan instruksi yang dikenal (dimengerti) oleh komputer dan disusun menurut urutan yang logis.

Bahasa Pemrograman (*Programming Language*) :

Bahasa yang digunakan untuk menulis program komputer.

Pemrograman (*programming*) :

Kegiatan membuat program komputer.

Pemrogram (*programmer*) :

Orang yang melakukan pemrograman komputer.

Seringkali kita dibingungkan dengan istilah belajar pemrograman dan belajar bahasa pemrograman. Dua istilah ini hampir mirip, namun sebenarnya memiliki arti yang sangat berbeda.

Belajar pemrograman :

Belajar tentang metode pemecahan masalah kemudian menuangkannya dalam suatu notasi yang mudah dipahami.

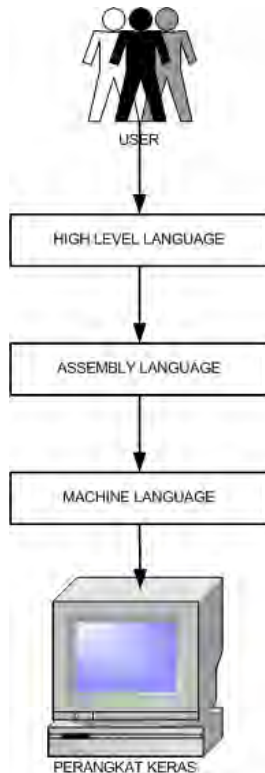
Belajar bahasa pemrograman :

Belajar memakai suatu bahasa, aturan tata bahasanya (syntax), instruksi-instruksi dan bagaimana memanfaatkan instruksi-instruksi tersebut dalam membuat program.

7.2 KLASIFIKASI BAHASA PEMROGRAMAN

Apabila kita melakukan pencarian di internet dengan kata kunci “programming language” maka kita akan menjumpai puluhan bahkan mungkin lebih dari seratus bahasa pemrograman, mulai dari yang sudah berusia puluhan tahun seperti FORTRAN, COBOL, BASIC, PASCAL, C, JAVA, dan lain-lain, sampai yang berusia baru beberapa tahun atau bulan seperti RUBY, PHYTON, dan bahasa-bahasa script yang banyak digunakan pada pemrograman berbasis web. Bahkan dalam satu bahasa seringkali timbul beberapa varian yang berkembang dan memiliki karakteristik tersendiri, sebagai contoh bahasa BASIC dalam perkembangannya memiliki varian seperti MBASIC, GW-BASIC yang kemudian berkembang menjadi bahasa untuk pemrograman visual seperti pada Visual Basic, Visual Basic.NET, Gambas (pada platform GNU-Linux).

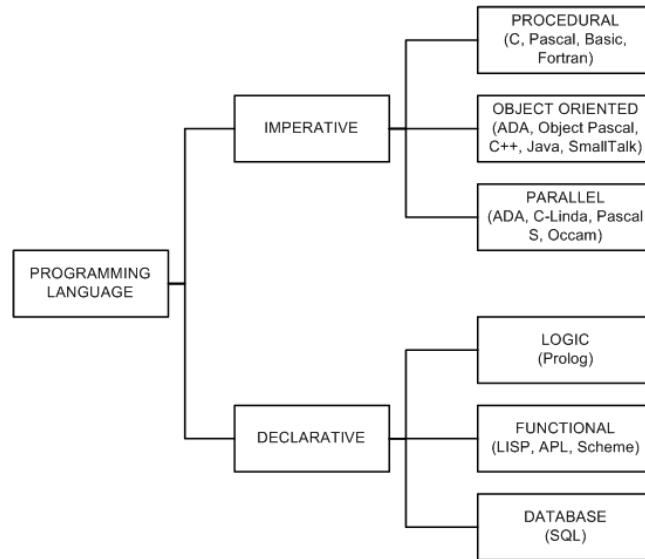
Berdasarkan kedekatannya dengan perangkat keras, klasifikasi bahasa pemrograman dapat dilihat pada Gambar 7.1. Pada gambar tersebut terlihat semakin mendekati perangkat keras, semakin mudah perangkat keras menerjemahkan instruksi-instruksinya namun bagi manusia semakin sulit untuk membaca. Semakin mendekati manusia semakin sulit bagi komputer untuk menerjemahkan instruksi namun semakin mudah manusia mengerti.



Gambar 7.1. Klasifikasi bahasa pemrograman berdasarkan kedekatannya dengan perangkat keras.

- Machine language (bahasa mesin) adalah bahasa pemrograman yang langsung dapat dimengerti oleh komputer. Bahasa ini ditulis dalam kombinasi angka 0 dan 1 yang menunjukkan ada atau tidaknya arus tegangan. Atau biasa dikenal sebagai sistem bilangan biner. Untuk membuat program dalam machine language kita harus mengetahui secara rinci lokasi serta isi dari sel-sel memori yang digunakan.
- Assembly language (bahasa assembly) mirip dengan machine language namun sudah menggunakan kode-kode singkatan perintah. Agar komputer dapat mengerti perintah dalam assembly language, maka dibutuhkan suatu penerjemah yang disebut sebagai assembler yang akan menerjemahkan program dalam assembly language menjadi machine language. Meskipun sudah menggunakan kode-kode perintah, namun bahasa ini masih cukup sulit untuk diikuti dan tidak praktis.
- High Level Language (bahasa tingkat tinggi) lebih mendekati bahasa manusia dan tidak tergantung pada jenis komputer yang digunakan. Bahasa pemrograman yang umum dipakai adalah bahasa dari kelompok ini. Pascal, Basic, Fortran, C dan bahasa-bahasa pemrograman modern berada pada kelompok ini. Program yang ditulis dengan menggunakan bahasa ini harus diterjemahkan dulu ke dalam machine language. Ada dua tipe penerjemah yaitu interpreter dan compiler.

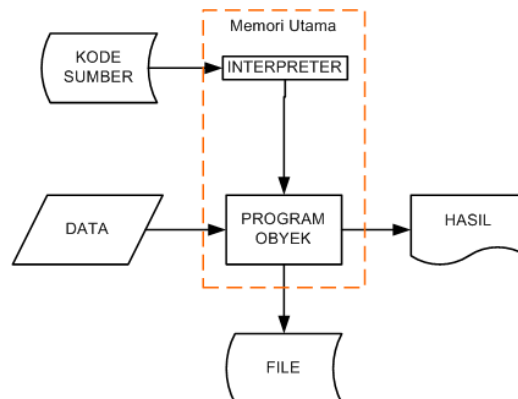
Klasifikasi lain dari bahasa pemrograman adalah berdasarkan tujuan dan fungsinya (Gambar 7.2).



Gambar 7.2. Klasifikasi bahasa pemrograman berdasarkan tujuan dan fungsinya.

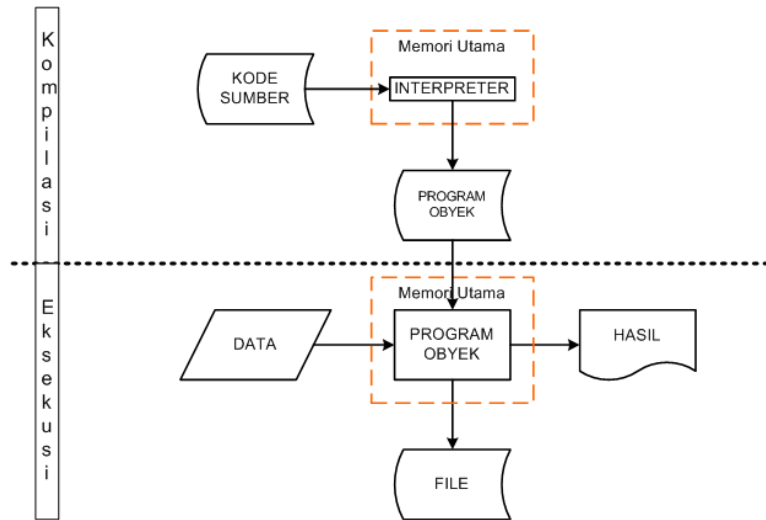
7.3 INTERPRETER DAN COMPILER

Interpreter adalah penerjemah bahasa pemrograman tingkat tinggi ke bahasa mesin dengan cara membaca instruksi-instruksi dari program (kode) sumber kemudian diterjemahkan dan langsung dilaksanakan satu persatu (Gambar 7.3). Jadi kita tidak memperoleh hasil terjemahan kode sumber dalam bahasa mesin. Setiap kali kita menjalankan program kita menggunakan kode sumbernya.



Gambar 7.3. Proses eksekusi interpreter.

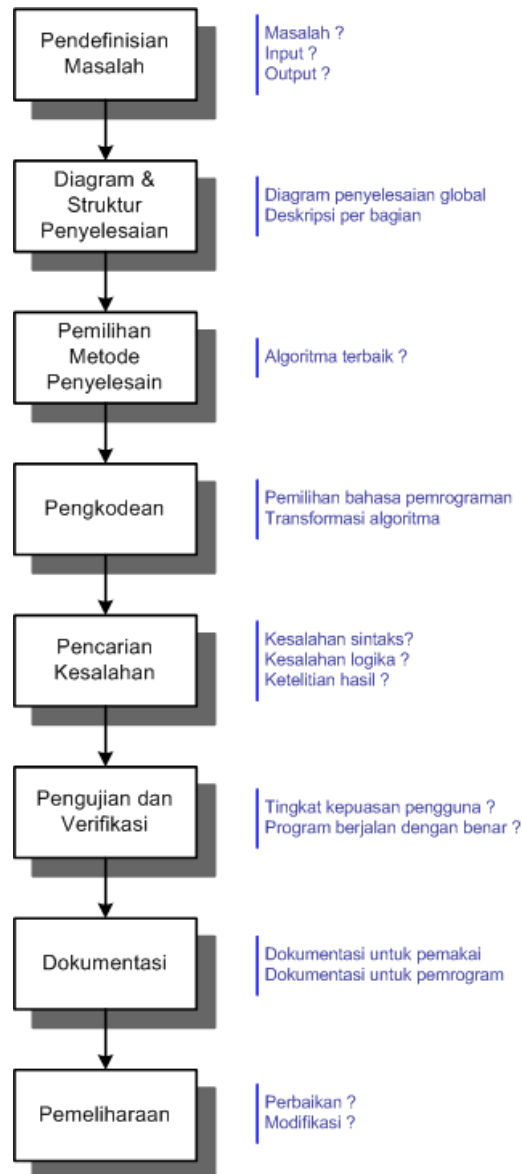
Compiler melakukan penerjemahan dengan cara membaca secara keseluruhan kode sumber dan menerjemahkan sekaligus menjadi satu set lengkap instruksi dalam bahasa mesin, sebelum instruksi-instruksi tersebut dieksekusi (Gambar 7.4). Hasil penerjemahan ini biasa disebut sebagai program obyek. Setiap kali kita ingin menjalankan program kita tidak perlu lagi kode sumbernya tapi cukup dengan program obyek saja.



Gambar 7.4. Proses eksekusi compiler.

7.4 TAHAPAN-TAHAPAN PEMROGRAMAN

Seperti telah disebutkan, pemrograman komputer dapat dilakukan dengan menggunakan berbagai bahasa pemrograman, dari bahasa tingkat rendah sampai bahasa tingkat tinggi. Pemrograman komputer merupakan bagian dari disiplin ilmu Software Engineering yang meliputi beberapa aspek seperti disain, konstruksi, dan pengendalian mutu. Pada aspek disain dan konstruksi, ada beberapa pendekatan yang digunakan untuk membantu memecahkan permasalahan, antara lain: pemrograman prosedural, pemrograman berbasis obyek, pemrograman berorientasi obyek, pemrograman terstruktur dan lain-lain. Masing-masing pendekatan memiliki ciri dan karakter tersendiri. Namun secara umum pemrograman komputer mempunyai tahapan-tahapan yang hampir serupa untuk menyelesaikan suatu masalah seperti terlihat pada Gambar 7.5.



Gambar 7.5. Tahapan-tahapan dalam pemrograman.

- ***Pendefinisian Masalah***

Pada tahap ini dilakukan upaya untuk memperjelas masalah apa sebenarnya yang akan diselesaikan. Selain itu batasan masalah juga harus diperjelas agar mempermudah pada tahapan-tahapan berikutnya. Setelah itu baru diidentifikasi input apa saja yang diperlukan dan output apa yang diinginkan.

- ***Pembuatan diagram dan struktur cara penyelesaian***

Tahapan ini dimulai dengan membuat diagram penyelesaian secara global kemudian dilanjutkan dengan mendeskripsikan tugas atau sub-tugas dari masing-masing bagian pada diagram tersebut. Cara ini ditempuh untuk mempermudah baik programmer maupun pengguna untuk melihat secara utuh cara penyelesaian suatu masalah. Penggunaan pendekatan seperti disebutkan di atas akan sangat berpengaruh pada tahapan ini.

- ***Pemilihan metode penyelesaian***

Setelah diagram dan struktur cara penyelesaian terbentuk, maka bagaimana metode penyelesaian yang terbaik dapat ditentukan. Dalam hal ini akan dipilih algoritma apa yang paling sesuai untuk memecahkan masalah.

- ***Pengkodean***

Tahap pengkodean secara prinsip berarti menterjemahkan algoritma ke dalam kode pada bahasa pemrograman tertentu.

- ***Pencarian kesalahan***

Pada pemrograman komputer sederhana yang hanya terdiri dari beberapa baris kode, kesalahan mungkin tidak dijumpai. Namun pada pemrograman yang kompleks kesalahan akan selalu muncul. Jarang sekali programmer dapat membuat program komputer yang kompleks sekali jadi.

- ***Pengujian dan verifikasi***

Tahapan ini bertujuan untuk menjamin bahwa sampai tingkat kepuasan tertentu, program berjalan benar dan untuk semua kemungkinan akan memberikan hasil yang benar.

- ***Pendokumentasian***

Pendokumentasian secara tertulis merupakan tahapan yang sering kali tidak dilakukan dengan baik, terutama oleh programmer itu sendiri. Hal ini karena programmer lebih fokus pada bagaimana menyelesaikan sebuah masalah, sehingga dokumentasi kadang diabaikan. Ada dua jenis dokumentasi tertulis, yaitu dokumen (manual) bagi pengguna dan dokumen bagi programmer. Dokumen (manual) bagi pengguna berisi petunjuk bagaimana mengoperasikan program itu secara jelas. Sedangkan dokumen bagi programmer berisi petunjuk atau komentar tentang bagaimana untuk mengubah, memperbaiki program tersebut dikemudian hari.

- **Pemeliharaan**

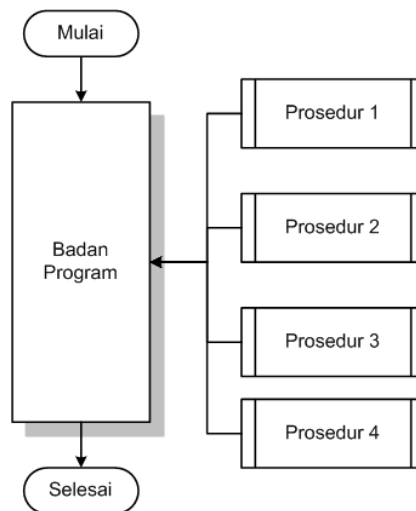
Pemeliharaan program dilakukan dengan cara memperbaiki kekurangan yang mungkin ditemukan kemudian atau memodifikasi atas dasar perubahan dalam spesifikasi masalah.

7.5 TEKNIK PEMROGRAMAN

Teknik pemrograman atau dikenal juga sebagai *programming paradigm* adalah style (pendekatan) dasar dalam pemrograman hubungannya dengan bagaimana solusi-solusi dari masalah diformulasikan dalam bahasa pemrograman. Berikut ini beberapa *programming paradigm* yang populer di kalangan programmer.

7.5.1 Pemrograman Prosedural

Pemrograman prosedural didasarkan pada konsep prosedur. Prosedur kadang-kadang disebut sebagai *routines*, *subroutines*, *methods* atau fungsi merupakan bagian tertentu yang berisi urutan langkah untuk menyelesaikan masalah tertentu. Prosedur-prosedur ini dapat dipanggil setiap waktu selama eksekusi program dilakukan. Gambar 7.6 menunjukkan bagaimana konsep pemrograman prosedural dilakukan.



Gambar 7.6. Pemrograman prosedural.

Keuntungan dari pendekatan dengan pemrograman prosedural adalah:

- Kemampuan untuk menggunakan kembali kode yang sama pada berbagai tempat tanpa harus menulis ulang.

- Mudah dalam mengikuti alur pemrograman.
- Kuat sebagai dasar pemrograman modular

Bahasa pemrograman yang dapat digunakan dalam pemrograman prosedural haruslah dapat digunakan untuk membuat prosedur dan memanggilnya kembali. BASIC, PASCAL, C, COBOL, FORTRAN, PHP, Perl adalah bahasa-bahasa pemrograman yang dapat digunakan untuk pemrograman procedural.

7.5.2 Pemrograman Terstruktur

Pemrograman terstruktur sebenarnya merupakan bagian dari pemrograman prosedural. Pada level pengkodean, pemrograman terstruktur biasanya mudah diikuti karena struktur aliran yang bersifat urut dan berjenjang. Tujuan utama teknik pemrograman terstruktur adalah :

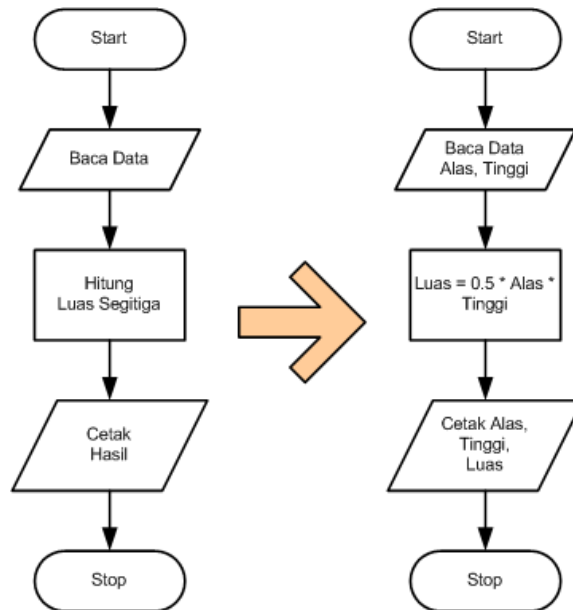
- membuat program yang benar
- membuat program yang dapat dipercaya
- membuat program yang mudah dibaca, dipahami dan dikoreksi sehingga mudah ketika dimodifikasi di masa datang.

Ada 3 tiga komponen dalam pemrograman terstruktur, yaitu top-down design, modular design, dan structured coding.

- top-down design

Top down design dimulai dengan penentuan tujuan program secara menyeluruh dan bukan pada bagaimana cara mencapainya. Setelah tujuan terdefinisi dengan jelas, barulah dibuat garis besar proses yang akan dilaksanakan. Kemudian secara bertahap garis besar ini diuraikan menjadi tahapan-tahapan yang lebih rinci, sehingga akhirnya semua bagian program selesai. Contoh pada Gambar 7.7 memperlihatkan bagaimana top-down design dilakukan.

Pada Gambar 7.7, tujuan dari program adalah mencari luas segitiga. Gambar sebelah kiri menunjukkan garis besar proses yang akan dilakukan program. Sedangkan gambar sebelah kanan merupakan detail proses untuk masing-masing bagian. Pada persoalan yang sederhana seperti di atas model top-down design tidak banyak memberikan manfaat, namun jika masalah semakin kompleks, model ini akan sangat bermanfaat.



Gambar 7.7. Top-Down Design.

- modular design

Modular design adalah membagi program dalam modul-modul yang lebih kecil yang lebih sederhana. Sebenarnya ketika melakukan top-down design kita juga sekaligus sudah merencanakan dan melakukan modular design.

Setiap modul program terdiri dari dari sekumpulan pernyataan yang memiliki fungsi/kegunaan tertentu di dalam program. Secara umum, apapun program yang akan dibuat, biasanya mengandung bagian-bagian berikut ini:

- Dokumentasi program
- Bagian pemasukan data
- Bagian pemeriksaan data
- Bagian manipulasi dan perhitungan terhadap data
- Bagian pembaharuan file (file update)
- Pengubahan data (editing)
- Pembuatan laporan (report Generator)

Dengan menggunakan modular design, kita dapat membuat bagian-bagian di atas sebagai suatu modul yang terpisah-pisah.

- structured coding

Structured coding atau kodifikasi terstruktur terdiri dari dua hal:

- Cara penulisan yang terstruktur

Cara penulisan yang terstruktur akan sangat memudahkan dalam membaca program dan memperbaiki jika ada kesalahan. Perhatikan contoh berikut:

Contoh 7.1.

Kode 1

```
If JlhAnak > 3 Then
PTKP = PTKP + (3 * 120000)
Else
PTKP = PTKP + (JlhAnak * 120000)
End If
```

Kode 2

```
If JlhAnak > 3 Then
    PTKP = PTKP + (3 * 120000)
Else
    PTKP = PTKP + (JlhAnak * 120000)
End If
```

Perhatikan kedua potongan kode di atas dan kita lihat bagaimana penulisan yang terstruktur memberikan hasil yang berbeda. Kode 2 jauh lebih mudah dibaca dari pada Kode 1.

Cara penulisan yang terstruktur dapat dilakukan dengan:

- Menggunakan identitas atau judul atau keterangan pada baris, namun tidak boleh berlebihan.
- Menggunakan indentasi seperti terlihat contoh 7.1.

- Penggunaan pernyataan secara terstruktur.

Penggunaan pernyataan terstruktur yang dimaksud disini adalah penggunaan satu atau lebih struktur algoritma yang akan kita pelajari di Bab 9, yaitu struktur berurutan, struktur pemilihan dan struktur pengulangan.

Bahasa pemrograman yang dapat digunakan untuk pemrograman *procedural*, hampir seluruhnya dapat digunakan untuk pemrograman terstruktur. Buku ini sebagian besar akan membahas bagian algoritma dan

pemrograman berdasarkan konsep pemrograman terstruktur dan pemrograman *procedural*.

7.5.3 Pemrograman Berorientasi Obyek

Pemrograman Berorientasi Obyek (Object Oriented Programming – OOP) adalah programming paradigm yang menggunakan obyek dan interaksinya untuk merancang aplikasi dan program komputer. OOP tidak banyak digunakan sebelum awal tahun 1990an. Tapi sekarang menjadi sesuatu yang sudah lumrah digunakan. Bahasa-bahasa pemrograman seperti keluarga dotNet dari Microsoft (Visual Basic.Net, Visual C#, dan Visual J), Borland Delphi, Java, Phyton, PHP versi 5 ke atas, C++ dan banyak lainnya merupakan bahasa pemrograman yang mendukung konsep OOP.

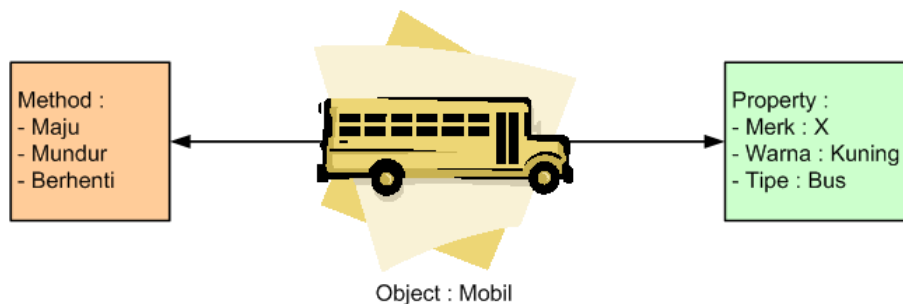
Beberapa prinsip yang menjadi dasar OOP akan dijelaskan berikut ini.

- Class

Class atau Kelas mendefinisikan karakteristik abstrak dari sesuatu (object) termasuk atribut atau sifat-sifat dari sesuatu dan apa yang dapat dikerjakan oleh sesuatu (method). Sebagai contoh, mobil adalah sebuah kelas yang memiliki attribut warna, merek, tipe dan lain-lain. Mobil juga punya method antara lain, maju, mundur dan berhenti (lihat Gambar 7.8).

- Object

Object adalah contoh dari kelas yang sudah didefinisikan. Atribut dan method dari kelas secara otomatis akan menurun pada object namun dengan kekhususan. Sebagai ilustrasi kita perhatikan Gambar 7.8. pada gambar tersebut, kita bisa identifikasi kelasnya adalah mobil dengan atribut dan methodnya. Objectnya adalah sebuah mobil sedan dengan merk Toyota, dan warnanya adalah merah. Sedan itu juga memiliki method maju, mundur dan berhenti. Pada kasus ini mobil sedan disebut sebagai instance dari kelas mobil.



Gambar 7.8. Class, object, attribute dan method.

- Method

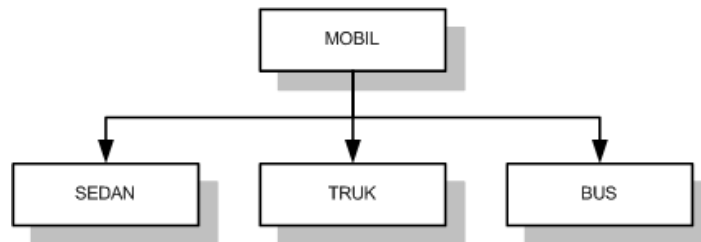
Method adalah kemampuan dari object atau kelas. Atau dengan kata lain adalah apa yang bisa dilakukan oleh suatu object. Pada contoh diatas mobil sedan mampu maju, mundur dan berhenti.

- Message Passing

Message parsing atau kadang disebut sebagai interfacing adalah proses dimana suatu object mengirim data pada object lain atau meminta object lain untuk menjalankan suatu method.

- Inheritance

Inheritance atau pewarisan adalah prinsip pewarisan sifat dari orang tua ke anak atau turunannya yang diterapkan pad kelas. Orang tua memiliki atribut dan method yang lebih umum dibandingkan dengan anak atau turunannya (Gambar 7.9).



Gambar 7.9. Pewarisan.

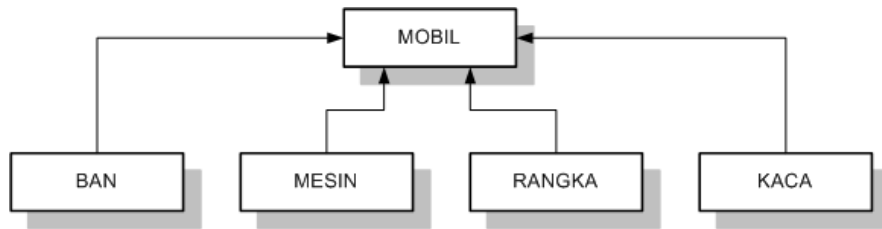
Seperti terlihat pada gambar 7.9, mobil adalah orang tua dan mungkin dapat dibagi menjadi anak kelas seperti sedan, truk dan bus. Misalnya mobil memiliki atribut warna dan method maju, maka baik sedan, truk dan bus secara otomatis akan memiliki atribut dan method yang sama. Hal ini sangat menguntungkan dalam pemrograman, karena kita hanya perlu mengetikkan kode sekali saja untuk semua. Selain atribut dan method yang dimiliki oleh orang tuanya, anak kelas juga dapat memiliki atribut dan method yang hanya berlaku untuk dirinya.

- Encapsulation

Prinsip encapsulation adalah prinsip menyembunyikan detail dari sebuah kelas terhadap object yang berinteraksi terhadapnya. Sebagai contoh ketika kita menjalankan mobil, sebenarnya kita sedang berinteraksi dan kita meminta kepada mobil untuk menjalankan methodnya seperti maju, mundur atau berhenti. Kita berinteraksi hanya dengan beberapa bagian dari mobil (*interface*) seperti persneling, setir, pijakan gas, pijakan rem dan bagian lain. Tapi detail proses yang terjadi didalam mobil bagaimana bisa maju, mundur atau berhenti kita tidak perlu tahu.

- Abstraction

Abstraction atau disebut juga *composition* merupakan prinsip penyederhanaan dari realitas kompleks dengan cara memodelkan kelas sesuai dengan masalahnya. Untuk lebih memperjelas pengertian mari kita lihat Gambar 7.10. Pada gambar tersebut terlihat sebuah mobil jika dipecah-pecah bagian-bagiannya kita akan dapatkan seperti ban, mesin, rangka mobil, kaca, dan lain-lain dan hal ini berlaku sebaliknya. Jika kita gabungkan bagian-bagian tersebut maka kita akan mendapatkan sebuah kelas mobil.



Gambar 7.10. Contoh abstraction

7.6 PEMILIHAN BAHASA PEMROGRAMAN

Pemilihan bahasa pemrograman yang digunakan dalam pengkodean, kadang-kadang menjadi suatu hal yang pelik, karena ada banyak hal yang berpengaruh dalam pemilihan ini. Secara umum faktor-faktor yang mempengaruhi adalah :

- Arsitektur sistem yang dipilih. Hal ini berpengaruh, terutama pada perangkat keras yang dipilih. Seringkali bahasa pemrograman yang dipilih harus menyesuaikan dengan hardware yang tersedia.
- Platform sistem operasi. Sistem operasi memberikan pengaruh besar pada kompatibilitas. Suatu program yang ditulis dan dijalankan pada sistem operasi tertentu mungkin tidak kompatibel bila dijalankan pada sistem operasi yang lain. Sebagai contoh aplikasi yang dibuat dengan Microsoft Visual Basic pada platform sistem operasi Microsoft Windows tidak akan dapat dijalankan langsung pada sistem operasi lain seperti GNU-Linux atau UNIX.
- Platform pengembangan yang disepakati. Pilihan yang tersedia saat ini dalam platform pengembangan adalah *desktop application* dan *internet/web application*. *Desktop application* cocok bila sistem informasi tidak digunakan banyak orang (atau hanya untuk satu komputer tersebut) atau jarang membutuhkan koneksi dengan perangkat lain. Sedangkan *web application* cocok bila sistem informasi harus diakses oleh banyak orang dengan lokasi tersebar.

Pengkodean untuk *desktop application* jelas berbeda dengan pengkodean *web application*.

- Biaya yang harus dikeluarkan. Perangkat lunak untuk pengkodean sangat beragam harganya. Dari yang gratis sampai dengan yang harganya ribuan dolar. Masing-masing memiliki keunggulan dan kerugian. Analisis yang seksama terhadap kebutuhan dan dihubungkan dengan biaya yang tersedia mungkin harus dilakukan agar diperoleh manfaat yang optimal.

Tipe Data, Variabel, Konstanta, Dan Operator

TUJUAN

Setelah anda membaca Bab ini, diharapkan anda akan mampu :

- Memilih tipe data yang sesuai dalam pemrograman.
 - Memahami cara penggunaan variabel.
 - Memahami cara penggunaan konstanta.
 - Memahami cara penggunaan operator.
-

8.1. TIPE DATA

Tipe Data :

Jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.

Ketepatan pemilihan tipe data pada variabel atau konstanta akan sangat menentukan pemakaian sumberdaya oleh aplikasi yang dihasilkan. Salah satu tugas penting seorang programmer adalah memilih tipe yang sesuai untuk menghasilkan program yang efisien dan berkinerja tinggi.

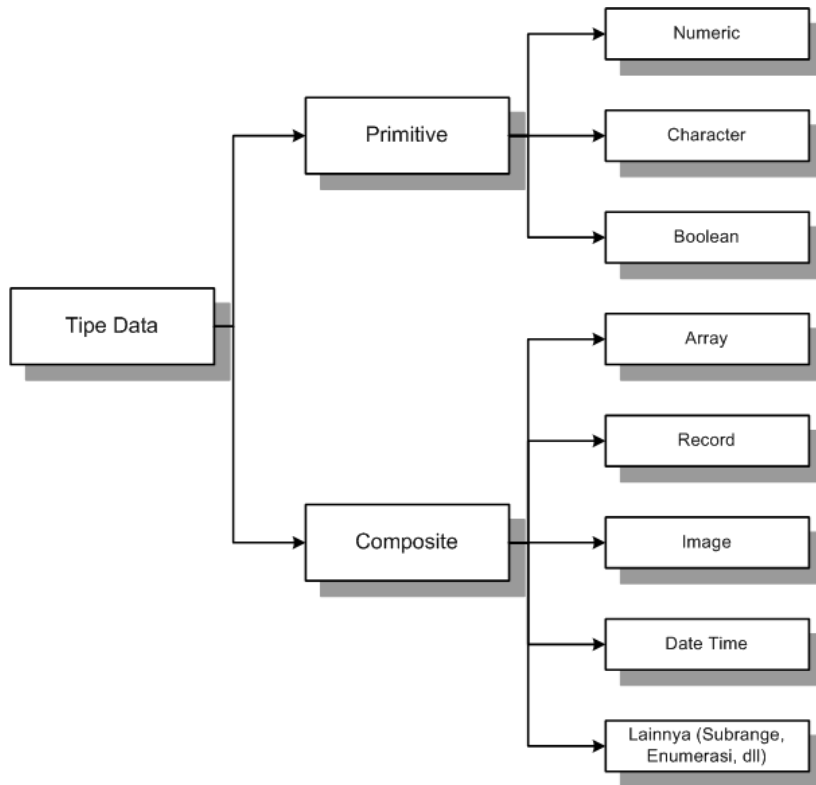
Ada beberapa kelompok tipe data yang tersedia, tetapi secara umum dapat dikelompokkan menjadi dua kelompok utama yaitu, *primitive type* dan *composite type*. Gambar 8.1 memperlihatkan pembagian kelompok ini.

Primitive type :

Tipe data dasar yang tersedia secara langsung pada suatu bahasa pemrograman.

Composite type :

Tipe data bentukan yang terdiri dari dua atau lebih primitive type.



Gambar 8.1. Pengelompokan tipe data.

8.1.1. Numeric

Tipe data numeric digunakan pada variabel atau konstanta untuk menyimpan nilai dalam bentuk bilangan atau angka. Semua bahasa pemrograman menyediakan tipe data numeric, hanya berbeda dalam jenis numeric yang diakomodasi. Beberapa jenis yang termasuk dalam tipe data numerik antara lain:

- *Integer* (bilangan bulat)
- *Single* (bilangan pecahan dengan presisi yang terbatas)
- *Double* (bilangan pecahan dengan presisi yang lebih akurat)

Tipe data *Integer* dapat dibagi menjadi dua bagian yaitu *Integer* yang mampu menyimpan bilangan bulat dari -32768 sampai dengan 32767 dan *Long* yang mampu menyimpan bilangan bulat dari -2147483648 sampai dengan 2147483647.

Tipe data *Single* dan *Double* berasal dari kata *single-precision* dan *double precision*. Pemilihan antara dua tipe ini menjadi penting karena sangat berhubungan dengan sumberdaya (memori) dan akurasi. *Single* memiliki

presisi yang lebih buruk dari *Double*, namun memori yang digunakan juga lebih sedikit, demikian juga sebaliknya. Perhatikan contoh berikut ini :

```
Dim A As Single, B As Double
A = 1 / 3
Print A
A = A * 100000
Print A
B = 1 / 3
Print B
B = B * 100000
Print B
```

Apabila kode program tersebut dijalankan maka hasilnya adalah sebagai berikut :

```
0.3333333          \ (nilai A = 1 / 3)
33333.34           \ (nilai A = A * 100000)
0.333333333333333 \ (nilai B = 1 / 3)
33333.3333333333   \ (nilai B = B * 100000)
```

Dari contoh tersebut terlihat perbedaan presisi ketika kita menggunakan Single dan Double pada operasi matematik yang sama.

8.1.2. Character

Bersama dengan tipe data numeric, character merupakan tipe data yang paling banyak digunakan. Tipe data character kadang disebut sebagai char atau string. Tipe data string hanya dapat digunakan menyimpan teks atau apapun sepanjang berada dalam tanda petik dua (“...”) atau petik tunggal (‘...’). Perhatikan contoh berikut ini :

```
Dim SomeText As String
SomeText = "Sesuatu yang tertunda"
SomeText = " "
SomeText = "15000"
```

Pada contoh ini kita mendeklarasikan variabel *SomeText* sebagai string, kemudian kita isikan nilai-nilai pada variabel ini (lihat baris 2, 3 dan 4). Isi dari variabel *SomeText* boleh terdiri dari angka, huruf, karakter, atau apapun. Sepanjang semuanya berada di dalam tanda petik dua maka akan selalu dianggap sebagai teks. Sebagai contoh pada baris ke 3, variabel *SomeText* kita isi dengan nilai “15000”. Apabila tanpa tanda petik dua (menjadi 15000) maka nilai yang kita masukkan menjadi tidak cocok karena 15000 bertipe data numeric. Kita juga tidak bisa secara langsung melakukan operasi matematika seperti penjumlahan, pengurangan, pengalian dan pembagian pada variabel *SomeText* yang berisi “15000”.

8.1.3. Boolean

Tipe data Boolean digunakan untuk menyimpan nilai True/False (Benar/Salah). Pada Visual Basic nilai -1 menunjukkan True dan 0 melambangkan False, meskipun biasanya nilai selain 0 menunjukkan nilai True. Tipe data ini banyak digunakan untuk pengambilan keputusan pada struktur kendali IF ... THEN atau IF ... THEN ... ELSE. Untuk mendeklarasikan sebuah variabel dengan tipe data Boolean, pada Visual Basic dilakukan dengan cara berikut:

```
Dim Test1 AS Boolean
```

8.1.4. Array

Array atau sering disebut sebagai larik adalah tipe data yang sudah terstruktur dengan baik, meskipun masih sederhana. Array mampu menyimpan sejumlah data dengan tipe yang sama (homogen) dalam sebuah variable. Setiap lokasi data array diberi nomor indeks yang berfungsi sebagai alamat dari data tersebut. Penjelasan tentang array akan disampaikan lebih detail pada Bab 10 tentang Algoritma Lanjutan.

8.1.5. Record

Seperti halnya Array, Record adalah termasuk tipe data komposit. Namun, berbeda dengan array, tipe data record mampu menampung banyak data dengan tipe data berbeda-beda (heterogen). Sebagai ilustrasi array mampu menampung banyak data namun dengan satu tipe data yang sama, misalnya integer saja. Sedangkan dalam record, kita bisa menggunakan untuk menampung banyak data dengan tipe data yang berbeda, satu bagian integer, satu bagian lagi character, dan bagian lainnya Boolean. Biasanya record digunakan untuk menampung data suatu obyek. Misalnya, siswa memiliki nama, alamat, usia, tempat lahir, dan tanggal lahir. Nama akan menggunakan tipe data string, alamat bertipe data string, usia bertipe data single (numeric), tempat lahir bertipe data string dan tanggal lahir bertipe data date. Berikut ini contoh pendeklarasian record dalam Delphi.

```
Type TRecord_Siswa = Record
    Nama_Siswa      : String[30]
    Alamat          : String[50]
    Usia            : Real
EndRecord
```

8.1.6. Image

Image atau gambar atau citra merupakan tipe data grafik. Misalnya grafik perkembangan jumlah siswa SMK, foto keluarga kita, video perjalanan

dan lain-lain. Pada bahasa-bahasa pemrograman modern terutama yang berbasis visual tipe data ini telah didukung dengan sangat baik.

8.1.7. Date Time

Nilai data untuk tanggal (Date) dan waktu (Time) secara internal disimpan dalam format yang spesifik. Variabel atau konstanta yang dideklarasikan dengan tipe data Date dapat digunakan untuk menyimpan baik tanggal maupun jam. Tipe data ini masuk dalam kelompok tipe data composite karena merupakan bentukan dari beberapa tipe data. Berikut ini contoh tipe data dalam Visual Basic :

```
Dim WaktuLahir As Date
WaktuLahir = "01/01/1997"
WaktuLahir = "13:03:05 AM"
WaktuLahir = "02/23/1998 13:13:40 AM"
WaktuLahir = #02/23/1998 13:13:40 AM#
```

8.1.8. Tipe data lain

- **Subrange**

Tipe data subrange merupakan tipe data bilangan yang mempunyai jangkauan nilai tertentu sesuai dengan yang ditetapkan programmer. Biasanya tipe data ini mempunyai nilai batas minimum dan nilai batas maksimum. Tipe data ini didukung dengan sangat baik dalam Delphi. Berikut ini contoh deklarasi tipe data subrange dalam Delphi.

```
Type
    BatasIndeks = 1..20
    RentangTahun = 1950..2030
Var
    Indeks          : BatasIndeks
    Tahun           : RentangTahun
```

- **Enumerasi**

Tipe data ini merupakan tipe data yang mempunyai elemen-elemen yang harus disebut satu persatu dan bernilai konstanta integer sesuai dengan urutannya. Nilai konstanta integer elemen ini diwakili oleh suatu nama variable yang ditulis di dalam kurung. Tipe data ini juga dijumpai pada Delphi dan bahasa pemrograman deklaratif seperti SQL. Berikut ini contoh deklarasi tipe data subrange dalam Delphi.

```
Type
    Hari_dlm_Minggu = (Nol, Senin, Selasa, Rabu,
                       Kamis, Jumat, Sabtu, Minggu)
    Nama_Bulan = (Nol, Januari, Pebruari, Maret,
```

```

                                April, Mei, Juni, Juli, Agustus,
                                September, Oktober, Nopember,
                                Desember)
Var
    No_Hari           : Hari_dlm_Minggu
    No_Bulan          : Nama_Bulan

```

Pada contoh di atas tipe data `Hari_dlm_Minggu` termasuk enumerasi dengan rentang nilai Nol, Senin sampai dengan Minggu dan nilai data dari 0, 1, sampai dengan 7. Sedangkan tipe data `Nama_Bulan` termasuk enumerasi dengan rentang nilai Nol, Januari sampai dengan Desember dan nilai data dari 0, 1, sampai dengan 12.

- **Object**

Tipe data object digunakan untuk menyimpan nilai yang berhubungan dengan obyek-obyek yang disediakan oleh Visual Basic, Delphi dan bahasa pemrograman lain yang berbasis GUI. Sebagai contoh, apabila kita mempunyai form yang memiliki control Command button yang kita beri nama `Command1`, kita dapat mendeklarasikan variabel sebagai berikut :

```

Dim A As CommandButton
Set A = Command1
A.Caption = "HEY!!!"
A.FontBold = True

```

Pada contoh ini variabel `A` dideklarasikan bertipe data Object yaitu `CommandButton`. Kemudian kita set variabel `A` dengan control Command button yang ada pada form (`Command1`). Dengan cara ini kita dapat mengakses seluruh property, method dan event obyek `Command1` dengan menggunakan variabel `A`.

- **Variant**

Tipe data hanya ada di Visual Basic. Tipe ini adalah tipe data yang paling fleksibel di antara tipe data yang lain, karena dapat mengakomodasi semua tipe data yang lain seperti telah dijelaskan.

8.2. VARIABEL

Variabel :

Tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan.

Setiap variabel akan mempunyai nama (identifier) dan nilai.

Variabel *UserName* misalnya, dapat mempunyai nilai “joni” atau “nijo” atau terserah pada kita. Sedangkan variabel *HargaTotal* dapat berisi nilai 30000 atau 25000. *UserName* dan *HargaTotal* adalah nama variabel sedangkan “joni”, “nijo”, 30000 dan 25000 adalah nilai dari masing-masing variabel.

Pada sebagian besar bahasa pemrograman, variabel harus dideklarasikan lebih dulu untuk mempermudah compiler bekerja. Apabila variabel tidak dideklarasikan maka setiap kali compiler bertemu dengan variabel baru pada kode program akan terjadi waktu tunda karena compiler harus membuat variabel baru. Hal ini memperlambat proses kerja compiler. Apabila variabel telah dideklarasikan lebih dulu, compiler akan tahu seluruh variabel yang digunakan dan tipe data masing-masing variabel. Hal ini akan menghasilkan kode program yang kompak, efisien dan optimal ketika proses kompilasi dijalankan.

Salah satu kritik paling besar pada bahasa pemrograman yang bersifat interpreter seperti BASIC, Python atau PHP, adalah karena bahasa-bahasa ini tidak mewajibkan pemrogram untuk mendeklarasikan seluruh variabelnya. Pada Visual Basic, pemrogram masih dibolehkan untuk tidak mendeklarasikan variabel, namun sangat disarankan untuk selalu mendeklarasikan seluruh variabel yang akan kita gunakan dalam kode program. Sedangkan pada bahasa pemrograman Pascal (Delphi) dan C, seluruh variabel harus dideklarasikan lebih dulu.

Ada dua cara untuk mendeklarasikan variabel, yaitu :

- **Deklarasi secara eksplisit**

Cara ini dilakukan dengan menggunakan pernyataan `Dim` diikuti dengan nama variabel dan tipe datanya seperti contoh berikut :

```
Dim UserName As String
Dim meters As Integer
```

UserName dan *meters* adalah nama variabel sedangkan *String* dan *Integer* adalah tipe data masing-masing variabel. Hal ini berarti, variabel *UserName* hanya boleh diisi dengan nilai variabel yang berupa teks dan

variabel meters hanya boleh diisi dengan nilai berupa bilangan bulat (Integer). Penjelasan tentang tipe data dapat dilihat pada bagian II dari modul ini.

- ***Deklarasi secara implisit***

Kita dapat mendeklarasikan variabel tanpa harus menentukan tipe datanya. Oleh Visual Basic, variabel ini akan ditentukan tipenya sebagai tipe data **variant** (tipe data generic yang dapat mengakomodasi seluruh tipe data. Ketika kita menggunakan variabel tersebut dalam kode program dan memberikan nilai pada variabel tersebut, Visual Basic akan mengatur tipe data variabel tersebut sesuai dengan nilai yang kita masukkan. Cara ini disebut deklarasi secara implicit. Perhatikan contoh berikut ini.

```
Dim Var1, Var2
Var1 = "Terima kasih"
Var2 = 50.45
```

Var1 dan Var2 adalah dua buah variabel yang tidak kita deklarasikan tipe datanya. Namun ketika kita mengisi nilai masing-masing variabel (lihat baris 2 dan 3) maka secara tidak langsung kita juga menentukan tipe data masing-masing variabel. Variabel Var1 menjadi bertipe String karena berisi teks, sedangkan Var2 menjadi bertipe Single karena berisi bilangan dengan pecahan.

Ketika kita mendeklarasikan variabel tertentu maka kita harus mengikuti aturan-aturan penamaan yang berlaku untuk bahasa pemrograman tersebut.

Aturan-aturan yang harus dipenuhi dalam penamaan variabel :

- *Harus diawali dengan huruf.*
- *Tidak boleh menggunakan spasi. Spasi bisa diganti dengan karakter underscore (_).*
- *Tidak boleh menggunakan karakter-karakter khusus (seperti : ., +, -, *, /, <, >, dll).*
- *Tidak boleh menggunakan kata-kata kunci yang sudah dikenal oleh bahasa pemrograman (seperti : dim, as, string, integer, dll).*
- *Harus unik ketika berada dalam scope-nya.*

Sebuah variabel mempunyai ruang-lingkup (scope) dan waktu-hidup (lifetime). Scope dari sebuah variabel adalah bagian dari program aplikasi yang dapat mengenali dan memanipulasi variabel tersebut. Sedangkan lifetime adalah seberapa lama nilai dalam suatu variabel itu akan tetap ada.

Berdasarkan scope dan lifetime-nya variabel dapat dibagi menjadi dua kelompok yaitu variabel global dan variable local.

Variabel global (Public) :

Variabel yang dapat dikenali dan dimanipulasi pada seluruh bagian program. Nilai data yang tersimpan didalamnya akan hidup terus selama program berjalan.

Variabel local :

Variabel yang hanya dapat dikenali dan dimanipulasi oleh satu bagian program saja, misalnya pada satu prosedur atau fungsi. Nilai data yang tersimpan didalamnya hanya hidup selama bagian program tersebut dijalankan.

8.3. KONSTANTA

Konstanta :

Variabel yang nilai datanya bersifat tetap dan tidak bisa diubah

Sebagai contoh, jika kita membuat program perhitungan matematik yang menggunakan nilai pi (3.14159) yang mungkin akan muncul dibanyak tempat pada kode program, kita dapat membuat pi sebagai konstanta. Penggunaan konstanta pi akan lebih memudahkan penulisan kode program dibanding harus mengetikkan nilai 3.14159 berulang-ulang. Penulisan deklarasi konstanta di dalam kode program adalah sebagai berikut:

```
Const <nama_kontanta> As <type_data> = <nilai_data>
```

Tipe data yang digunakan dalam konstanta sama persis yang digunakan dalam pendeklarasian variabel. Kita bisa menggunakan numeric, string, boolean, date, object atau variant sebagai tipe data konstanta.

Contoh :

```
Const pi As Single = 3.14159  
Area = 2 * pi * Radius
```

Kita dapat mendeklarasikan pi sebagai variabel, namun cara ini tidak disarankan karena nilai pi tidak berubah-ubah selama eksekusi program dan penggunaan konstanta lebih cepat pemrosesannya dari pada variabel.

Beberapa petunjuk hubungannya dengan ruang lingkup (scope) dari suatu user-defined konstanta antara lain:

- Jika suatu konstanta hanya ada dalam suatu procedure, maka konstanta tersebut dideklarasikan hanya dalam procedure tersebut
- Jika suatu konstanta berlaku pada semua procedure dalam suatu module, maka konstanta tersebut dideklarasikan pada bagian deklarasi di module tersebut
- Jika suatu konstanta berlaku pada semua aplikasi, maka konstanta tersebut harus dideklarasikan dengan keyword `Public` sebelum kata `Const`.

8.4. OPERATOR

Operator :

Simbol yang digunakan dalam bahasa pemrograman untuk melakukan suatu operasi terhadap nilai data. Simbol operator bisa berupa karakter ataupun kata khusus.

8.4.1. Operator aritmatika

Operator aritmatika digunakan untuk operasi matematis terhadap nilai data. Notasi-notasi yang digunakan tampak pada tabel berikut ini :

Tabel 8.1. Operator Aritmatika.

Simbol	Operasi Matematis	Contoh
$\wedge$	pemangkatan	$5 \wedge 2$ hasilnya 25
*	perkalian	$5 * 2$ hasilnya 10
/	pembagian (hasil pecahan)	$5 / 2$ hasilnya 2,5
\	pembagian (hasil bulat)	$5 \setminus 2$ hasilnya 2
Mod	sisa pembagian	$5 \text{ Mod } 2$ hasilnya 1
+	penjumlahan	$5 + 2$ hasilnya 7
-	pengurangan	$5 - 2$ hasilnya 3
&	penggabungan string	$5 \& 2$ hasilnya 52

Notasi/symbol untuk operator ini memiliki hirarki kekuatan, artinya apabila ada dua atau lebih operator digunakan bersama-sama, maka operator dengan hirarki yang lebih tinggi akan dieksekusi lebih dulu. Hirarki kekuatan operator secara berturut-turut adalah $\wedge$, $*$ dan $/$, $\setminus$, mod, $+$ dan $-$. Operator $*$ dan $/$ adalah setara. Demikian juga operator $+$ dan $-$. Perhatikan contoh berikut ini.

$$5 * 2 + 3 = 13$$

$$4 \wedge 2 - 5 = 11$$

Pada contoh pertama, tanda * (kali) memiliki kekuatan yang lebih tinggi dari tanda + (tambah) sehingga operasi perkalian dilakukan terlebih dahulu daripada penjumlahan. Hasil yang diperoleh adalah 10 ditambah 3 bukan 5 dikali 5. Sedangkan pada contoh kedua, tanda ^ (pangkat) mempunyai urutan kekuatan lebih tinggi dari – (kurang) sehingga hasil akhirnya adalah 11 (yaitu dari 16 dikurangi 5). Untuk merubah urutan proses aritmatika dapat diatur dengan menggunakan tanda kurung. Perhatikan contoh berikut :

$$5 * (2 + 3) = 25$$

$$4 ^ (2 - 5) = 0.015625$$

Pada contoh ini kita menggunakan angka dan operator yang sama seperti contoh sebelumnya, namun dengan menggunakan tanda kurung untuk merubah urutan proses. Pada contoh pertama proses yang pertama dilakukan adalah penjumlahan 2 dan 3, kemudian baru dikalikan 5. Hasil akhirnya adalah 25 (bandingkan dengan contoh sebelumnya, yaitu 13). Sedangkan pada contoh kedua proses pertama adalah 2 dikurangi 5 kemudian baru hasilnya digunakan sebagai angka pangkat untuk 4 (menjadi 4^3). Hasil yang diperoleh adalah 0.015625, berbeda jauh dengan contoh sebelumnya yaitu 11. Dari kedua contoh ini dapat dilihat bahwa tanda kurung memiliki kekuatan lebih tinggi dari operator lainnya. Apabila ada lebih dari satu tanda kurung berjajar maka tanda kurung paling dalam yang akan diproses terlebih dahulu. Perhatikan contoh berikut :

$$5 * ((2 + 2) / 8) = 2.5$$

8.4.2. Operator perbandingan

Operator perbandingan digunakan untuk operasi yang membandingkan nilai data. Simbol-simbol yang digunakan :

Tabel 8.2. Operator perbandingan.

Simbol	Operasi Perbandingan	Contoh
<	lebih kecil	$5 < 2$ hasilnya FALSE
>	lebih besar	$5 > 2$ hasilnya TRUE
<=	lebih kecil atau sama dengan	$5 <= 2$ hasilnya FALSE
>=	lebih besar atau sama dengan	$5 >= 2$ hasilnya TRUE
=	sama dengan	$5 = 2$ hasilnya FALSE
<>	tidak sama dengan	$5 <> 2$ hasilnya TRUE

8.4.3. Operator logika

Operator logika digunakan untuk operasi yang membandingkan suatu perbandingan. Simbol-simbol yang digunakan :

Tabel 8.3. Operator logika.

Simbol	Operasi Logika	Contoh
Or	atau	$(5 < 2)$ Or $(5 > 2)$ hasilnya TRUE
And	dan	$(5 < 2)$ And $(5 > 2)$ hasilnya FALSE
Not	Tidak	Not $(5 < 2)$ hasilnya TRUE

Algoritma Pemrograman Dasar

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian algoritma dan karakteristiknya.
 - Memahami cara penulisan algoritma.
 - Membuat algoritma dasar dengan struktur berurutan.
 - Membuat algoritma dasar dengan struktur pemilihan.
 - Membuat algoritma dasar dengan struktur perulangan
-

9.1. PENGERTIAN ALGORITMA

Algoritma :

Urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis.

Masalah dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Secara informal konsep algoritma sering kali disetarakan dengan sebuah resep. Sebuah resep biasanya memiliki daftar bahan atau bumbu yang akan digunakan, urutan pengerjaan dan bagaimana hasil dari urutan pengerjaan tersebut. Apabila bahan yang digunakan tidak tertera (tidak tersedia) maka resep tersebut tidak akan dapat dikerjakan. Demikian juga jika urutan pengerjaannya tidak beraturan, maka hasil yang diharapkan tidak akan dapat diperoleh.

Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan kriteria yang sama. Tingkat kerumitan dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Umumnya, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat kerumitan yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan tingkat kerumitan yang tinggi.

Sebagai ilustrasi, berikut ini adalah algoritma sederhana untuk mencari luas segitiga apabila nilai alas dan tinggi diketahui :

Contoh 9.1

1. Start
2. Baca data alas dan tinggi.
3. Luas adalah alas kali tinggi kali 0.5
4. Tampilkan Luas
5. Stop

Algoritma di atas adalah algoritma yang sangat sederhana, hanya ada lima langkah. Pada algoritma ini tidak dijumpai perulangan ataupun pemilihan. Semua langkah dilakukan hanya satu kali.

Sekilas algoritma di atas benar, namun apabila dicermati maka algoritma ini mengandung kesalahan yang mendasar, yaitu tidak ada pembatasan pada nilai data untuk alas dan tinggi. Bagaimana jika pengguna memasukkan nilai alas atau tinggi dengan bilangan 0 atau bilangan negatif ? Tentunya hasil yang keluar menjadi tidak sesuai dengan yang diharapkan. Dalam kasus seperti ini struktur pemilihan akan sangat membantu, misalnya dengan melakukan pengecekan pada input yang masuk. Apabila input nilai alas dan tinggi kurang dari 0 maka program tidak akan dijalankan, sebaliknya maka program akan dapat dieksekusi. Sehingga algoritma di atas dapat dirubah menjadi sebagai berikut :

Contoh 9.2

1. Start
2. Baca data alas dan tinggi.
3. Periksa data alas dan tinggi, jika nilai data alas dan tinggi lebih besar dari nol maka lanjutkan ke langkah ke 4 jika tidak maka stop
4. Luas adalah alas kali tinggi kali 0.5
5. Tampilkan Luas
6. Stop

Dari ilustrasi di atas dapat diambil sebuah dua kesimpulan pokok tentang algoritma. Pertama, **algoritma harus benar**, Kedua **algoritma harus berhenti**, dan setelah berhenti, **algoritma memberikan hasil yang benar**.

Menurut Knuth (1973) algoritma harus memiliki sebagai berikut :

Ciri-ciri penting Algoritma:

- *Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas.*
- *Setiap langkah harus didefinisikan dengan tepat dan tidak ambigu.*
- *Algoritma memiliki nol atau lebih masukan (input)*
- *Algoritma memiliki nol atau lebih keluaran (output)*
- *Algoritma harus efektif dan efisien.*

9.2. CARA PENULISAN ALGORITMA

Ada tiga cara penulisan algoritma, yaitu :

- **Structured English (SE)**

SE merupakan alat yang cukup efisien untuk menggambarkan suatu algoritma. Basis dari SE adalah Bahasa Inggris, namun kita dapat memodifikasi dengan Bahasa Indonesia sehingga kita boleh menyebutnya sebagai Structured Indonesia (SI). Algoritma seperti pada contoh 9.1 dan 9.2 merupakan algoritma yang ditulis menggunakan SI. Karena basisnya adalah bahasa sehari-hari, maka SE atau SI lebih tepat untuk menggambarkan suatu algoritma yang akan dikomunikasikan kepada pemakai sistem.

- **Pseudocode**

Pseudocode mirip dengan SE. Karena kemiripan ini kadang-kadang SE dan Pseudocode dianggap sama. *Pseudo* berarti imitasi atau tiruan atau menyerupai, sedangkan *code* menunjuk pada kode program. Sehingga pseudocode adalah kode yang mirip dengan instruksi kode program sebenarnya. Pseudocode berbasis pada bahasa pemrograman yang sesungguhnya seperti BASIC, FORTRAN atau PASCAL. Pseudocode yang berbasis bahasa PASCAL merupakan pseudocode yang sering digunakan. Kadang-kadang orang menyebut pseudocode sebagai PASCAL-LIKE algoritma. Apabila contoh 9.1 ditulis dalam pseudocode berbasis bahasa BASIC akan tampak sebagai berikut :

Contoh 9.3

1. **Start**
2. **READ** alas, tinggi
3. **Luas** = 0.5 * alas * tinggi
4. **PRINT** Luas
5. **Stop**

Pada contoh 9.3 tampak bahwa algoritma sudah sangat mirip dengan bahasa BASIC. Pernyataan seperti READ dan PRINT merupakan keyword yang ada pada bahasa BASIC yang masing-masing menggantikan kata “baca data” dan “tampilkan”. Dengan menggunakan pseudocode seperti di atas maka proses penterjemahan dari algoritma ke kode program menjadi lebih mudah.

- **Flowchart**

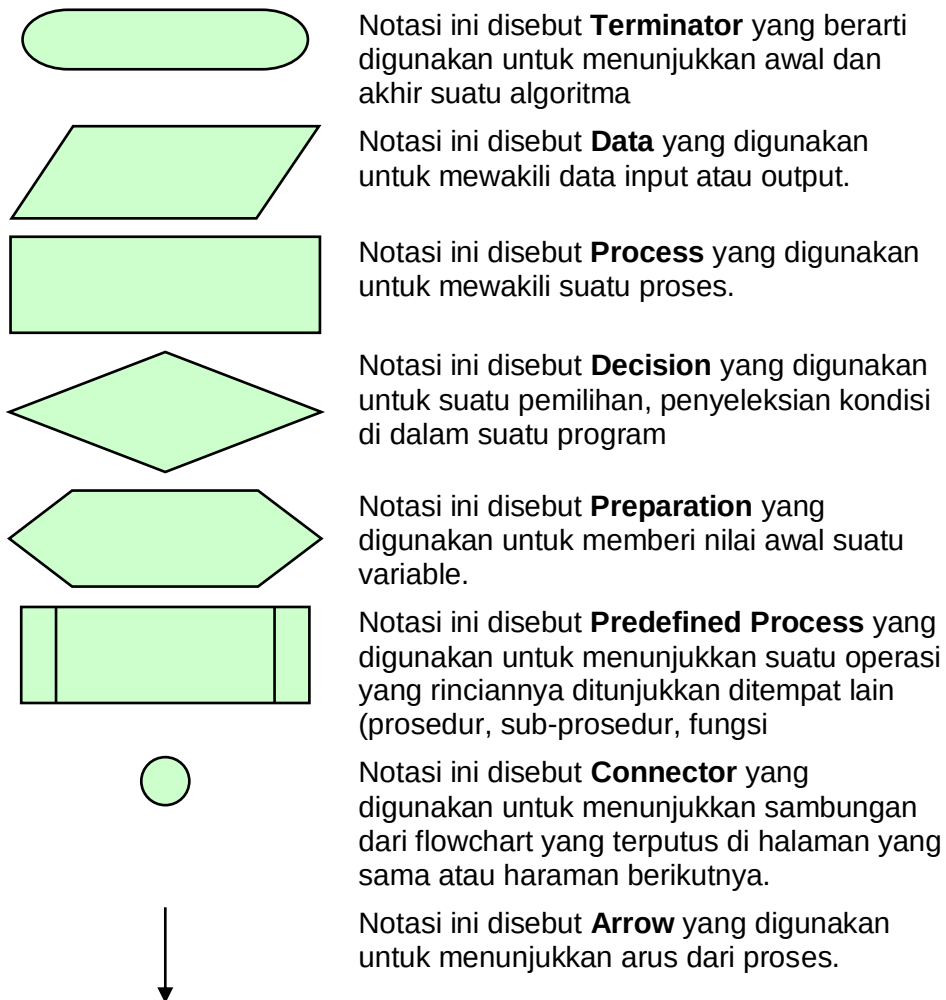
Flowchart atau bagan alir adalah skema/bagan (chart) yang menunjukkan alir (flow) di dalam suatu program secara logika. Flowchart merupakan alat yang banyak digunakan untuk menggambarkan algoritma dalam bentuk notasi-notasi tertentu. Secara lebih detail bagian ini akan dibahas pada bagian berikutnya.

9.3. FLOWCHART

Flowchart :

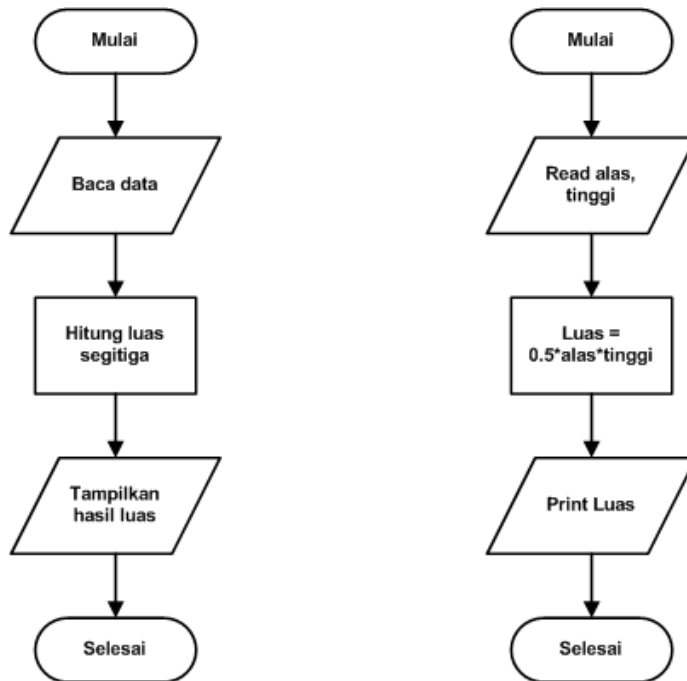
Skema atau gambar yang merepresentasikan suatu proses tertentu.

Flowchart banyak digunakan dalam presentasi untuk menjelaskan secara visual sebuah urutan langkah agar peserta lebih mudah memahami. Ada beberapa bentuk flowchart antara lain System Flowchart, Document Flowchart, Schematic Flowchart, Program Flowchart, dan Process Flowchart. Pada bagian ini, hanya akan dibahas tentang Program Flowchart, karena flowchart ini yang paling erat kaitannya dengan algoritma pemrograman. Pada Program Flowchart ada beberapa notasi penting yang digunakan untuk membuat algoritma sebagaimana tercantum pada Gambar 9.1.



Gambar 9.1. Notasi Flowchart

Program Flowchart dapat terdiri dari dua macam, yaitu bagan alir logika program (*program logic flowchart*) dan bagan alir program komputer terinci (*detailed computer program flowchart*). Bagan alir logika program digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika dan biasanya dipersiapkan oleh seorang analis system. Sedangkan bagan alir program komputer terinci digunakan untuk menggambarkan instruksi-instruksi program komputer secara terinci dan biasanya dipersiapkan oleh seorang programmer. Apabila contoh 9.1 dibuat program flowchartnya maka akan tampak pada gambar 9.2.



Bagan alir logika program

Bagan alir program komputer terinci

Gambar 9.2. Program flowchart.

9.4. STRUKTUR BERURUTAN

Ada tiga struktur dasar yang digunakan dalam membuat algoritma yaitu struktur berurutan (*sequence*), struktur pemilihan/keputusan (*decision*) dan struktur pengulangan (*iteration*). Sebuah algoritma biasanya akan menggabungkan ketiga buah struktur ini untuk menyelesaikan masalah.

Struktur berurutan terdiri satu atau lebih instruksi. Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya, yaitu sebuah instruksi dieksekusi setelah instruksi sebelumnya selesai dieksekusi. Urutan instruksi menentukan keadaan akhir dari algoritma. Bila urutannya diubah, maka hasil akhirnya mungkin juga berubah. Menurut Goldshlager dan Lister (1988) struktur berurutan mengikuti ketentuan-ketentuan sebagai berikut:

Ketentuan-ketentuan struktur urut :

- *tiap instruksi dikerjakan satu persatu*
- *tiap instruksi dilaksanakan tepat sekali, tidak ada yang diulang*
- *urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam algoritmanya*
- *akhir dari instruksi terakhir merupakan akhir algoritma.*

Goldshlager dan Lister (1988)

Contoh 9.4.

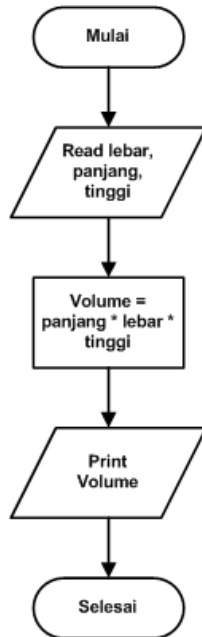
Buat flowchart untuk menghitung luas bangun berikut :

- balok
- lingkaran

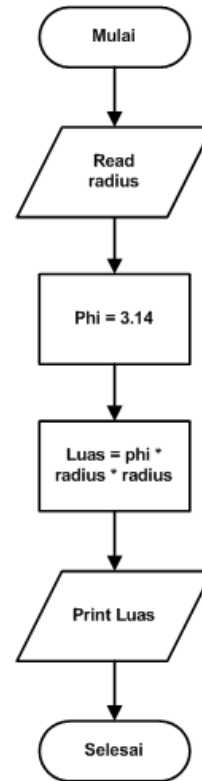
(rumus-rumus untuk masing-masing bangun dapat dicari pada buku geometri)

Penyelesaian:

Soal ini merupakan permasalahan dengan algoritma struktur berurutan karena tidak ada proses pemilihan atau pengulangan. Berikut flowchat untuk bagian a dan b.



a. flowchart menghitung volume balok



b. flowchart menghitung luas lingkaran

Gambar 9.3. Flowchart untuk penyelesaian contoh 9.4.

Contoh 9.5.

Ada dua gelas A dan B, gelas A berisi larutan berwarna merah, gelas B berisi larutan berwarna kuning. Pertukarkan isi dari kedua gelas itu sedemikian rupa sehingga gelas A berisi larutan kuning dan gelas B berisi larutan merah. Buatlah algoritma penyelesaiannya dengan menggunakan SI (Structured Indonesia).

Penyelesaian:

Penyelesaian ini masih menggunakan algoritma dengan struktur urut. Untuk mempertukarkan isi dua buah gelas maka diperlukan satu gelas tambahan (misalkan gelas C) sebagai tempat sementara sehingga cairan dalam kedua gelas tidak akan bercampur. Berikut adalah jawaban soal tersebut :

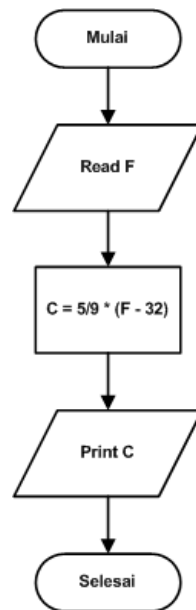
Mulai
Tuangkan isi gelas A ke dalam gelas C
Tuangkan isi gelas B ke dalam gelas A
Tuangkan isi gelas C ke dalam gelas B
Stop

Contoh 9.6.

Buat flowchart untuk mengubah temperatur dalam Fahrenheit menjadi temperatur dalam Celcius dengan rumus $^{\circ}\text{C} = 5/9 \times (^{\circ}\text{F} - 32)$.

Penyelesaian:

Soal ini juga masih menggunakan algoritma dengan struktur urut. Berikut flowchart penyelesaian masalah :



Gambar 9.4. Flowchart penyelesaian untuk contoh 9.6

9.5. STRUKTUR PEMILIHAN / PERCABANGAN

Sebuah program tidak selamanya akan berjalan dengan mengikuti struktur urut, kadang-kadang kita perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Peristiwa ini kadang disebut sebagai percabangan/pemilihan atau keputusan. Pada struktur seperti ini program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi. Pada proses seperti ini notasi flowchart Decision dapat digunakan.

9.5.1. Struktur pemilihan sederhana

Struktur pemilihan sederhana biasanya melibatkan pernyataan If ... Then dan If ... Then ... Else. Beberapa contoh struktur pemilihan sederhana dapat dilihat berikut ini.

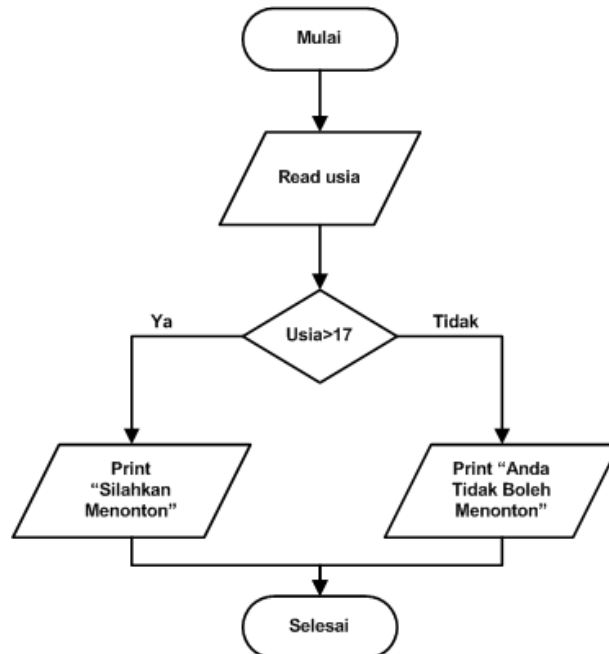
Contoh 9.7.

Sebuah aturan untuk menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun maka penonton diperbolehkan dan apabila kurang dari 17 tahun maka penonton tidak diperbolehkan nonton. Buatlah flowchart untuk permasalahan tersebut.

Penyelesaian:

Permasalahan diatas merupakan tipikal permasalahan yang menggunakan struktur pemilihan. Perhatikan pada pernyataan *jika .. maka ...*

Flowchart penyelesaian masalah tampak pada Gambar 9.5. Pada gambar 9.5. tampak penggunaan notasi Decision. Pada notasi ini terjadi pemeriksaan kondisi, yaitu apakah usia lebih dari 17 tahun atau tidak. Jika jawaban ya maka program akan menghasilkan keluaran teks “Silahkan Menonton”, sedangkan jika input usia kurang dari 17 tahun maka program akan menghasilkan keluaran teks “Anda Tidak Boleh Menonton”.



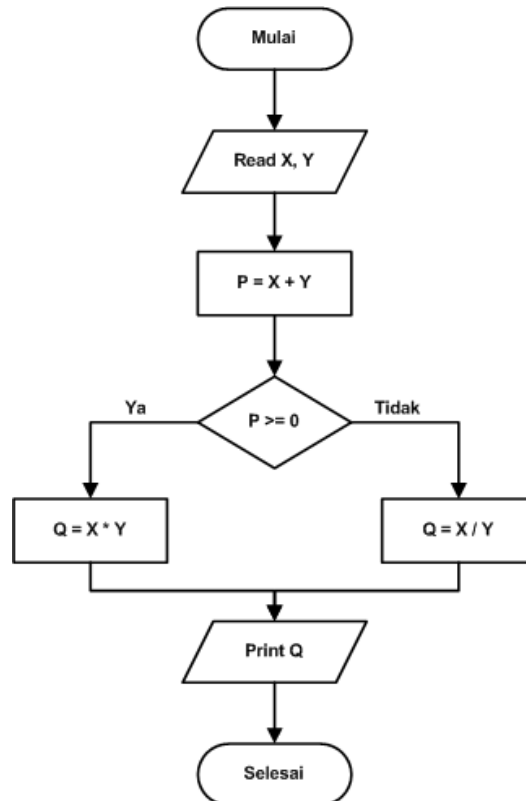
Gambar 9.5. Flowchart penyelesaian masalah nonton film.

Contoh 9.8.

Dalam suatu perhitungan nilai $P = X + Y$. Jika P positif, maka $Q = X * Y$, sedangkan jika negative maka nilai $Q = X/Y$. Buatlah flowchart untuk mencari nilai P dan Q

Penyelesaian:

Pada soal ini kita mulai berkenalan dengan struktur kendali pemilihan (percabangan) namun masih dalam taraf sederhana. Input yang dibutuhkan adalah nilai X dan Y , sedangkan proses pemeriksaan kondisi dilakukan pada nilai P apakah positif (termasuk 0) ataukah negative. Berikut ini flowchart penyelesaian masalah :



Gambar 9.6. Flowchart penyelesaian untuk contoh 9.8.

9.5.2. Struktur pemilihan bersarang (nested)

Struktur pemilihan bersarang merupakan pengembangan dari struktur pemilihan sederhana. Struktur ini kita jumpai bila ada satu kondisi pemilihan yang berada dalam kondisi pemilihan yang lain.

Contoh 9.9.

Sebuah usaha fotokopi mempunyai aturan sebagai berikut :

- jika yang fotokopi statusnya adalah langganan, maka berapa lembar pun dia fotokopi, harga perlembar Rp. 75,-
- jika yang fotokopi bukan langganan, maka jika dia fotokopi kurang dari 100 lembar harga perlembar Rp. 100,-. Sedangkan jika lebih atau sama dengan 100 lembar maka harga perlembar Rp. 85,-.

Buat flowchart untuk menghitung total harga yang harus dibayar jika seseorang memfotokopi sejumlah X lembar.

Penyelesaian:

Pada kasus kedua ini, ada dua percabangan. Yang pertama adalah pemeriksaan apakah status orang pelanggan atau bukan. Kedua, apabila status seseorang bukan pelanggan, maka dilakukan pemeriksaan berapa jumlah lembar fotokopi, apakah lebih dari 100 lembar atau tidak.

Pada soal ini kita juga menjumpai apa yang disebut sebagai nested. Perhatikan pernyataan pada syarat kedua dari persoalan di atas.

***jika** yang fotokopi bukan langganan, maka **jika** dia fotokopi kurang dari 100 lembar harga perlembar Rp. 100*

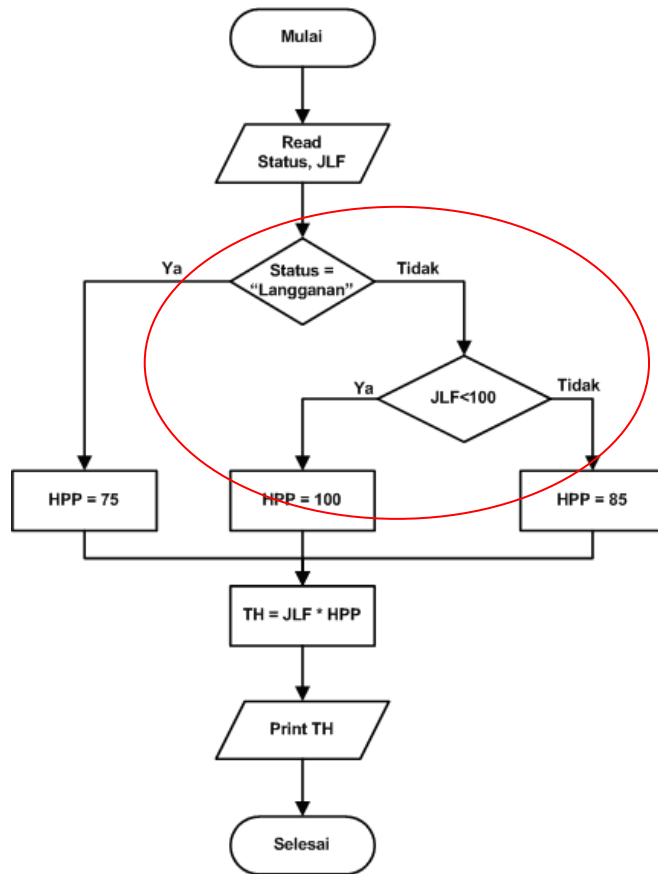
pernyataan jika yang kedua berada di dalam jika yang pertama.

Flowchart penyelesaian masalah ini dapat dilihat pada Gambar 9.7.

Input yang dibutuhkan untuk permasalahan ini adalah status orang yang fotokopi dan jumlah lembar yang difotokopi. Sehingga variable input yang digunakan adalah:

- **Status** untuk status orang yang fotokopi
- **JLF** untuk jumlah lembar yang difotokopi

Selain itu terdapat variable dengan nama HPP yang digunakan untuk menyimpan harga per lembar dan TH untuk menyimpan nilai total harga. Perhatikan, variable Status bertipe data string, sehingga penulisannya harus menggunakan tanda “ ”.



Gambar 9.7. Flowchart penyelesaian untuk contoh 9.9.

Contoh 9.10.

Dalam penilaian kelulusan siswa pada mata pelajaran Pemrograman Web, diterapkan aturan sebagai berikut :

- Jika nilai ujian tengah semester (UTS) lebih besar dari 70 maka siswa dinyatakan lulus dan tidak perlu mengikuti ujian akhir semester (UAS) dan Nilai Akhir sama dengan nilai UTS.
- Jika nilai UTS kurang atau sama dengan 70 maka siswa harus mengikuti UAS. Siswa dinyatakan lulus jika Nilai Akhir lebih besar atau sama dengan 60 dimana

$$\text{Nilai Akhir} = (\text{nilai UTS} \times 40\%) + (\text{nilai UAS} \times 60\%)$$

Buatlah flowchart penyelesaian masalah tersebut apabila output yang diinginkan adalah NIM, Nama Siswa, Nilai Akhir dan Status Kelulusan.

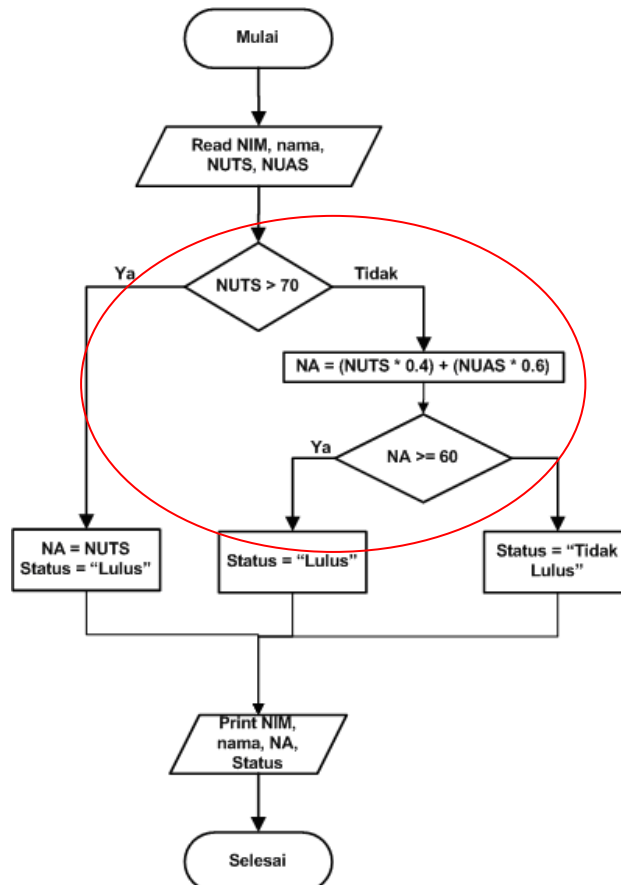
Penyelesaian:

Pada kasus ini, ada dua percabangan. Yang pertama adalah pemeriksaan apakah nilai UTS siswa lebih dari 70. Kedua, apabila nilai UTS tidak lebih dari 70, maka dilakukan pemeriksaan apakah nilai akhir lebih dari 60.

Input yang dibutuhkan untuk permasalahan ini adalah NIM, nama siswa, nilai UTS, dan nilai UAS. Sehingga variable input yang digunakan adalah:

- NIM untuk Nomor induk siswa
- nama untuk nama siswa
- NUTS untuk nilai ujian tengah semester
- NUAS untuk nilai ujian akhir semester

Selain itu terdapat variable dengan nama NA yang digunakan untuk menyimpan nilai akhir dan Status untuk menyimpan status kelulusan. Perhatikan, variable Status bertipe data string, sehingga penulisannya harus menggunakan tanda “ ”.



Gambar 9.8. Flowchart penyelesaian untuk contoh 9.10.

9.6. STRUKTUR PENGULANGAN

Salah satu kelebihan komputer dibandingkan dengan manusia adalah kemampuannya untuk mengerjakan tugas atau suatu instruksi berulang kali tanpa merasa lelah, bosan, atau malas. Dalam banyak penyelesaian masalah seringkali kita dihadapkan pada sejumlah pekerjaan yang harus diulang beberapa kali, demikian juga dalam sebuah algoritma. Satu atau sekumpulan instruksi kadang harus diulang sejumlah kali, atau sampai kondisi penghentian pengulangan tercapai. Struktur pengulangan satu atau lebih instruksi ini disebut struktur pengulangan atau looping.

Struktur pengulangan terdiri dari dua bagian :

1. Kondisi pengulangan, yaitu syarat yang harus dipenuhi untuk melaksanakan pengulangan. Syarat ini biasanya dinyatakan dalam ekspresi Boolean yang harus diuji apakah bernilai benar (true) atau salah (false)
2. Badan pengulangan (loop body), yaitu satu atau lebih instruksi yang akan diulang

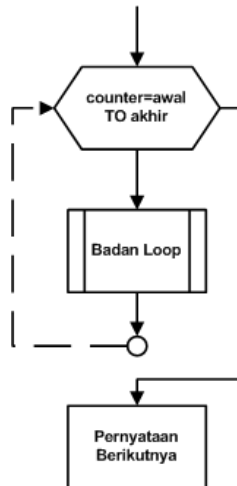
Pada struktur pengulangan, biasanya juga disertai bagian inisialisasi dan bagian terminasi. Inisialisasi adalah instruksi yang dilakukan sebelum pengulangan dilakukan pertama kali. Sedangkan terminasi adalah instruksi yang dilakukan setelah pengulangan selesai dilaksanakan. Bagian inisialisasi dan terminasi bersifat opsional, namun inisialisasi umumnya digunakan, misalnya untuk memberi nilai awal sebuah variable.

Ada beberapa bentuk pengulangan yang dapat digunakan, masing-masing dengan syarat dan karakteristik tersendiri. Beberapa bentuk dapat dipakai untuk kasus yang sama, namun ada bentuk yang hanya cocok untuk kasus tertentu saja. Pemilihan bentuk pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan bentuk pengulangan yang tepat bergantung pada masalah yang akan diprogram. Dalam modul ini akan dijelaskan dua bentuk pengulangan yang didukung oleh tidak hanya oleh Visual Basic. Bentuk-bentuk pengulangan yang akan dijelaskan ini memiliki beberapa kemiripan dengan bahasa pemrograman modern lainnya seperti Delphi, PHP, Java, dan lain-lain. Kedua bentuk pengulangan itu adalah :

9.6.1. Struktur pengulangan dengan For

Pengulangan dengan menggunakan For ... Next, merupakan salah satu teknik pengulangan yang paling tua dalam bahasa pemrograman. Hampir semua bahasa pemrograman menyediakan metode ini, meskipun sintaksnya mungkin berbeda. Pada struktur For ... Next kita harus tahu terlebih dahulu seberapa banyak badan loop akan diulang. Struktur ini menggunakan sebuah variable yang biasa disebut sebagai loop's counter, yang nilainya akan naik

atau turun selama proses pengulangan. Sintaks dan flowchart untuk struktur For ... Next adalah sebagai berikut :



Gambar 9.9. Struktur algoritma pengulangan dengan For.

Dalam mengeksekusi sebuah pengulangan dengan For ... Next, urutan langkah-langkah adalah sebagai berikut :

1. Menetapkan nilai counter sama dengan start
2. Memeriksa apakah nilai counter lebih besar daripada end. Jika benar maka keluar dari proses loop. Apabila increment bernilai negative , maka VB akan memeriksa apakah nilai counter lebih kecil daripada end. Jika benar maka keluar dari proses pengulangan.
3. Mengeksekusi pernyataan yang ada di badan loop
4. Menaikkan nilai counter sesuai dengan jumlah yang ditentukan pada argument increment. Apabila argument increment tidak ditetapkan maka secara default nilai counter akan dinaikkan 1.
5. Ulang kembali mulai langkah no 2.

Satu hal yang penting yang harus kita perhatikan adalah nilai counter selalu ditetapkan diawal dari pengulangan. Apabila kita mencoba merubah nilai end pada badan loop, maka tidak akan berdampak pada berapa banyak pengulangan akan dilakukan. Pada contoh berikut ini proses pengulangan hanya akan dilakukan 10 kali bukan 100 kali :

Contoh 9.11.

```
endValue = 10
For i = 1 To endValue
    endValue = 100
Next i
```

Nilai counter dapat kita rubah di dalam badan loop untuk mengurangi atau menaikkan jumlah ulangan. Perhatikan contoh berikut ini:

Contoh 9.12.

```
For i = 0 To 10
    i = i - 1
Next i
```

Berapa kalikah pengulangan akan dilakukan? Pengulangan pada contoh tersebut tidak akan pernah berhenti karena nilai counter yaitu i tidak pernah bertambah naik. Berikut ini contoh penggunaan pengulangan dengan For ... Next.

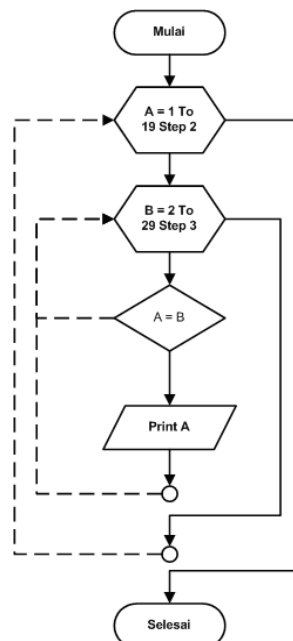
Contoh 9.13.

Diketahui dua buah himpunan sebagai berikut :

Himpunan $A = (1, 3, 5, \dots, 19)$ dan Himpunan $B = (2, 5, 8, \dots, 29)$

Buatlah flowchart untuk mencari anggota himpunan A yang juga merupakan anggota himpunan B (perpotongan antara himpunan A dan himpunan B).

Penyelesaian:

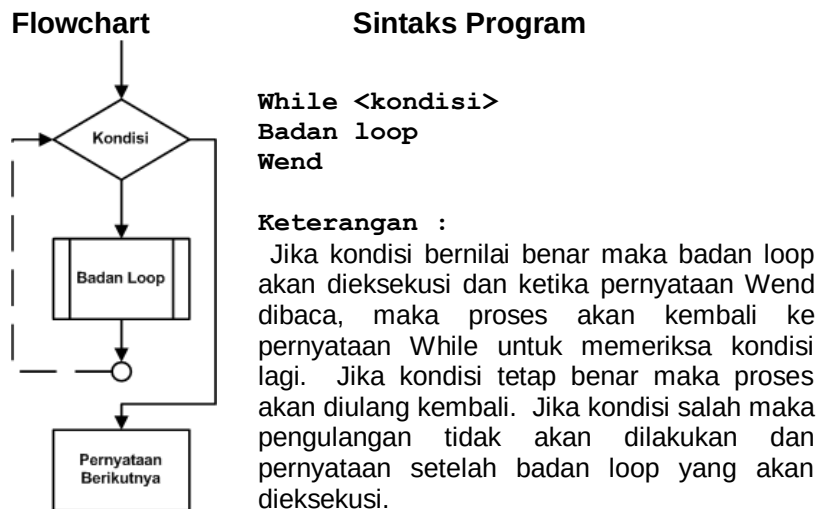


Gambar 9.10. Flowchart penyelesaian untuk contoh 9.13.

9.6.2. Struktur pengulangan dengan While, Loop, dan Repeat

Pada For ... Next banyaknya pengulangan diketahui dengan pasti karena nilai awal (start) dan nilai akhir (end) sudah ditentukan diawal pengulangan. Bagaimana jika kita tidak tahu pasti harus berapa kali mengulang? Sebagian besar bahasa pemrograman menyediakan pernyataan While ... Wend dan Do ... Loop untuk mengatasi hal ini. Pada bagian ini akan dijelaskan tentang penggunaan While ... Wend.

Struktur While ... Wend akan mengulang pernyataan pada badan loop sepanjang kondisi pada While bernilai benar. Syntax dan flowchart untuk struktur While ... Wend adalah sebagai berikut :



Gambar 9.11. Struktur pengulangan dengan While..Wend

Perhatikan potongan program berikut ini. Berapakah nilai variable Total pada akhir pengulangan dan berapa kalikah pengulangan dilakukan?

Contoh 9.14.

```
Dim number As Integer
Dim Total As Integer
number = 0
Total = 0
While number < 5
    Total = Total + number
    number = number + 1
Wend
Debug.Print Total
```

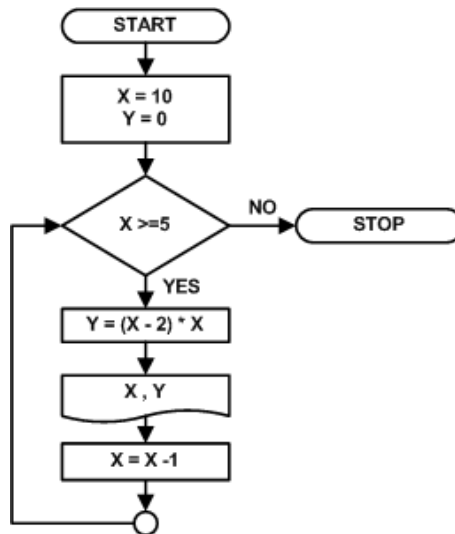
Proses pengulangan pada kode program di atas akan berlaku sebagai berikut :

1. Pernyataan While akan memeriksa apakah nilai variable number kurang dari 5. Jika salah maka keluar dari proses pengulangan (pernyataan Debug.Print Total yang akan dieksekusi, jika benar maka eksekusi pernyataan di bawah baris pernyataan While.
2. Eksekusi pernyataan $Total = Total + number$
3. Tambahkan nilai number dengan 1.
4. Ketika bertemu pernyataan Wend, maka program akan mengembalikan ke baris pernyataan While.
5. Ulangi kembali mulai langkah no 1.

Hasil dari proses pengulangan diatas, nilai variable Total pada akhir pengulangan adalah 10 dan pengulangan dilakukan sebanyak 5 kali. Coba anda uraikan lebih detil langkah di atas.

Contoh 9.15.

Perhatikan flowchart berikut ini. Kemudian tentukan outputnya.



Gambar 9.12. Contoh penggunaan While .. wend

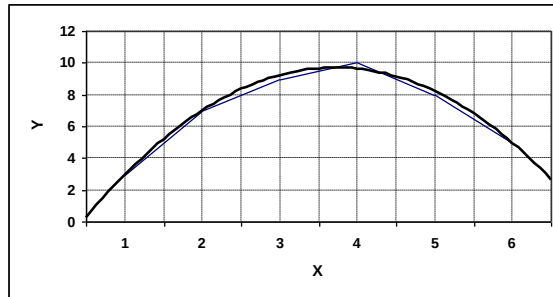
Penyelesaian:

Persoalan di atas adalah persoalan pengulangan dengan menggunakan while ... wend. Ada pengecekan pada notasi decision, apakah nilai x lebih besar atau sama dengan lima. Berikut output dari flowchart di atas.

X	Y
10	80
9	63
8	48
7	35
6	24
5	15

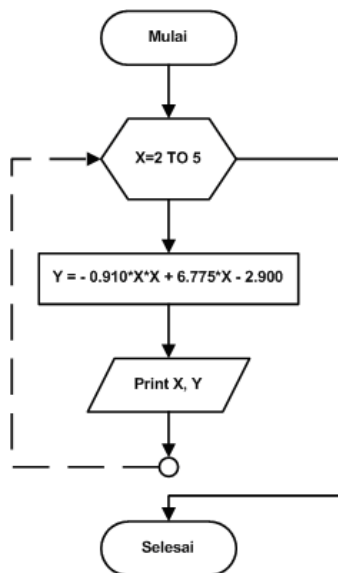
Contoh 9.16.

Gambar berikut ini mengikuti persamaan matematika $Y = -0,910x^2 + 6,775x - 2,900$.



- Tentukan semua variable dan tipe data yang digunakan.
- Buatlah flowchart untuk mencari nilai-nilai Y untuk x antara 2 sampai dengan 5.

Penyelesaian:



Gambar 9.13. Flowchart penyelesaian untuk contoh 9.16.

Algoritma Pemrograman Lanjutan

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Membuat algoritma prosedur.
 - Membuat algoritma array.
-

10.1. PROSEDUR

Setiap bahasa pemrograman selalu menyediakan fungsi-fungsi yang sudah didefinisikan oleh bahasa pemrograman tersebut (*built-in function*). Namun ada kalanya kita memerlukan suatu prosedur tertentu yang kita gunakan berulang kali dan tidak tersedia dalam *built-in function*.

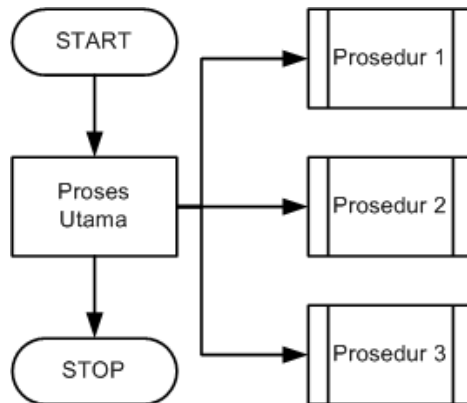
Prosedur :

Sekumpulan perintah yang merupakan bagian dari program yang lebih besar yang berfungsi mengerjakan suatu tugas tertentu.

Prosedur atau kadang disebut subrutin / subprogram biasanya relative independent terhadap bagian kode program yang lain. Atau sebenarnya prosedur dapat berdiri sendiri. Keuntungan menggunakan prosedur adalah :

- mengurangi duplikasi kode program.
- memberikan kemungkinan penggunaan kembali kode untuk program yang lain.
- memecah masalah yang rumit dalam masalah-masalah yang lebih kecil dan lebih mudah diselesaikan (lihat prinsip problem reduction pada Bab 2).
- membuat kode program lebih mudah dibaca.
- dapat digunakan untuk menyembunyikan detail program.

Pada flowchart untuk menuliskan prosedur digunakan notasi Predefined Process (lihat Bab 9 untuk notasi flowchart). Secara skematis penggunaan prosedur dapat dilihat pada Gambar 10.1.



Gambar 10.1. Skema penggunaan prosedur.

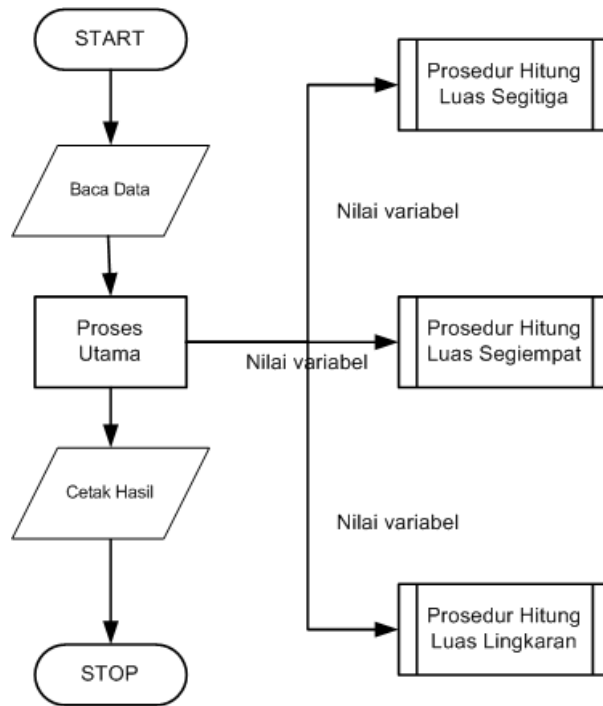
Gambar 10.1. menunjukkan ada proses utama yang terjadi dan ada prosedur yang sebenarnya merupakan bagian dari proses utama ini. Ketika proses utama membutuhkan suatu tugas tertentu maka proses utama akan memanggil prosedur tertentu menyelesaikan tugas tersebut. Perhatikan contoh berikut:

Contoh 10.1.

Buatlah algoritma menghitung luas segitiga, segiempat, dan lingkaran.

Penyelesaian:

Untuk membuat algoritma ini kita dapat memandang proses perhitungan luas segitiga, luas segiempat dan luas lingkaran sebagai bagian program yang berdiri sendiri. Kita dapat membuat prosedur untuk masing-masing proses. Dan kita akan memanggil prosedur tersebut dari proses utama (Gambar 10.2).



Gambar 10.2. Penyelesaian contoh 10.2.

Urutan proses pada Gambar 10.2 adalah sebagai berikut.

- Pembacaan data
- Pada proses utama akan terjadi pengecekan pada data yang dibaca,
- Apabila data yang dibaca adalah untuk segitiga, maka proses utama akan memanggil prosedur hitung luas segitiga dengan membawa nilai variabel yang diperlukan oleh prosedur luas hitung segitiga.
- Proses perhitungan luas segitiga hanya dilakukan pada prosedur tersebut.
- Setelah proses perhitungan maka hasil perhitungan akan dibawa kembali ke proses utama untuk dicetak hasilnya.

Urutan proses yang sama juga terjadi jika data yang dibaca adalah untuk segiempat atau lingkaran.

Prosedur yang baik memiliki ciri-ciri sebagai berikut :

- Hanya memiliki satu fungsi tujuan (*logical inherent*).

Sebuah prosedur sebaiknya hanya memiliki satu fungsi tujuan dan tidak bercampur dengan tujuan-tujuan lain. Hal ini untuk membuat prosedur lebih focus sehingga tujuan akan dapat tercapai dengan baik.

- Tidak tergantung pada prosedur lain (*independent*).

Sebuah prosedur harusnya bersifat mandiri, artinya prosedur ini dapat berjalan dan diuji tanpa menunggu bagian lainnya selesai. Selain itu variable-variabel yang digunakan dalam prosedur tersebut tidak mempengaruhi variable-variabel yang digunakan pada bagian lain di keseluruhan program.

- Berukuran kecil (*small size*).

Yang dimaksud ukuran disini adalah panjang algoritma atau panjang kode program pada suatu prosedur. Ukuran kecil akan mudah dibaca dan diperbaiki. Apabila sebuah modul sudah terlalu besar maka sebaiknya dipertimbangkan untuk dipecah-pecah menjadi beberapa modul yang lebih kecil.

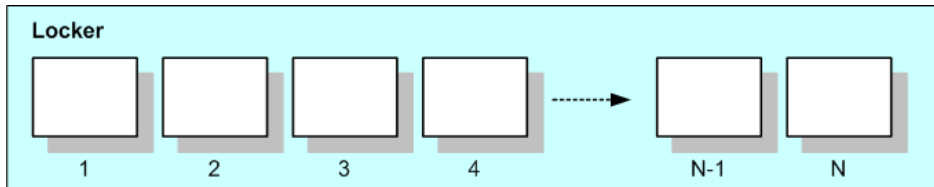
10.2. ARRAY

Variabel-variabel yang kita gunakan selama ini adalah variable biasa yang memiliki sifat bahwa nama variable hanya dapat menyatakan sebuah nilai numeric atau string pada suatu saat. Apabila kita ingin memberi nilai yang baru pada variable tersebut maka nilai lama akan hilang tergantikan oleh nilai yang baru. Bagaimana apabila kita ingin menyimpan beberapa nilai/data dalam sebuah variable dengan nama yang sama, tetapi semua nilai tetap tersimpan? Solusi yang dapat dilakukan adalah dengan menggunakan indeks pada nama variable tersebut. Pendekatan ini biasa disebut dengan array.

Array :

Struktur data yang menyimpan sekumpulan elemen yang bertipe sama, setiap elemen diakses langsung melalui indeksnya.

Array adalah struktur standar untuk menyimpan data pada sebagian besar bahasa pemrograman. Indeks array haruslah tipe data yang menyatakan keterurutan, misalnya integer atau string. Array dapat dianalogikan sebagai sebuah lemari atau locker yang memiliki sederetan kotak penyimpanan yang diberi nomor berurutan (lihat Gambar 10.3). Untuk menyimpan atau mengambil sesuatu dari kotak tertentu kita hanya cukup mengetahui nomor kotaknya saja.



Gambar 10.3. Lemari dengan banyak kotak laci di dalamnya

Array memiliki sekumpulan tempat yang masing-masing berisi sebuah nilai. Untuk menyimpan atau melihat nilai tempat tertentu dari array ini, kita hanya perlu mengetahui indeksinya saja. Sebagai contoh untuk menyimpan nilai tertentu pada sebuah array dapat dilakukan dengan kode sebagai berikut:

```
FOR I = 1 TO 3
    A(I) = I * I
    PRINT A(I)
NEXT I
```

Output dari kode di atas adalah $A(1) = 1$, $A(2) = 4$ dan $A(3) = 9$. Semua nilai akan disimpan dalam satu nama variable yang sama, yaitu A. Namun nilai hasil $I * I$ akan disimpan pada indeks (dalam kasus ini adalah I) yang ditunjuk. Dengan cara seperti ini akan lebih mudah dan menghemat penulisan kode dibandingkan harus membuat nama variable yang berbeda-beda untuk setiap nilai yang berbeda.

10.2.1. Array satu dimensi

Tidak seperti variable biasa, array harus didefinisikan dengan pernyataan Dim kemudian diikuti dengan nama variable array dan nilai indeks maksimum yang dapat disimpan seperti pada contoh berikut ini :

```
Dim Gaji(20) As Long
```

Contoh tersebut adalah sebuah array satu dimensi dengan nama variable I dan jumlah maksimum elemen adalah 21. Jumlah elemen tidak 20 tapi 21 karena secara default, variable array akan selalu dimulai dari indeks 0. $I(0)$ adalah nilai variable Gaji untuk orang yang pertama, $I(1)$ adalah variable Gaji untuk orang yang kedua, dan seterusnya. Kita dapat menambahkan dengan variable array lainnya misalnya sebagai berikut :

```
Dim Nama(20) As String
```

Kemudian kita bisa mengisi nilai pada masing-masing elemen dari kedua variable array diatas dengan cara berikut ini :

```
Nama(0) = "Joni"
Gaji(0) = 250000
Nama(1) = "Niyo"
```

```
Gaji(1) = 350000
. . .
Nama(20) = "Ojin"
Gaji(20) = 450000
```

Setiap elemen pada array seperti pada contoh harus mempunyai tipe data yang seragam. Sebagai contoh pada variable Gaji dideklarasikan tipe datanya adalah Long, maka seluruh nilai yang kita masukkan pada variable Gaji harus mempunyai tipe data Long. Namun apabila tipe data variable dideklarasikan sebagai variant maka kita boleh mengisi nilai variable sembarang tipe data.

Seperti dijelaskan di atas, secara default element pertama dari array selalu berindeks 0, namun kita dapat merubahnya dengan memberikan batas bawah (lower bound) dan batas atas (upper bound) indeks. Batas bawah indeks dapat berupa angka berapapun, namun yang harus diperhatikan adalah batas bawah indeks harus lebih kecil dari batas atas indeks. Contoh berikut ini dapat menjelaskan hal ini.

```
Dim Nama(1 To 20) As String
Dim Gaji(1 To 20) As Long
Dim NoTelpon (100 To 500) as String
```

10.2.2. Array multi-dimensi

Array satu dimensi seperti pada bagian II sangat baik untuk menyimpan data sejenis yang berurutan, namun bagaimana bila kita ingin menyimpan daftar kota dengan temperature rata-ratanya secara bersama-sama, atau menyimpan data nama Siswa dengan nilai ujiannya? Pada kasus seperti ini kita dapat menggunakan dua array satu dimensi, satu array untuk menyimpan nama dan satu array untuk menyimpan nilai. Namun ini bukanlah pilihan yang baik karena akan menyulitkan dan membuat kode program menjadi tidak efisien. Pilihan yang lebih baik adalah dengan menggunakan pendekatan Array Multidimensi. Kita dapat menyimpan dengan menggunakan array dua dimensi untuk kasus di atas. Perhatikan gambar berikut ini untuk melihat perbedaan **dua array satu dimensi** dengan **array dua dimensi**.

Nama(4)	Nilai(4)		NilaiSiswa (4,1)	
Joni	70	0	Joni	70
Nijo	80	1	Nijo	80
Ojin	45	2	Ojin	45
Jino	56	3	Jino	56
Noji	77	4	Noji	77

Dua array satu dimensi

Array dua dimensi

Dua array satu dimensi

Array dua dimensi

Gambar 10.4. Perbedaan array satu dimensi dan dua dimensi

Array dua dimensi mempunyai dua indeks. Indeks yang pertama menunjukkan baris sedangkan indeks yang kedua adalah kolom. Pada Gambar 10.2, variable array NilaiSiswa memiliki dua indeks yaitu indeks pertama 4 yang menyatakan nilai indeks maksimal untuk baris adalah 4 (atau ada 5 baris karena indeks baris pertama bernilai 0), sedangkan indeks kedua adalah 1 yang menunjukkan nilai indeks maksimal untuk kolom adalah 1 (atau ada 2 kolom karena indeks kolom pertama bernilai 0). Untuk mendeklarasikan array dua dimensi dapat digunakan cara sebagai berikut :

Dim NilaiSiswa(4,1)

Sedangkan untuk mengakses nilai pada array dua dimensi dapat digunakan pernyataan sebagai :

NilaiSiswa(3,0) 'untuk mengakses nilai "Jino" (indeks baris 3, indeks kolom 0)

NilaiSiswa(3,1) 'untuk mengakses nilai 56 (indeks baris 3, indeks kolom 1)

Keuntungan menggunakan array multidimensi adalah secara konseptual, array ini lebih mudah dikelola. Sebagai contoh kita ingin membuat program permainan dan kita ingin mencari posisi dari suatu tempat pada sebuah papan permainan. Setiap bujursangkar dapat diidentifikasi dengan menggunakan dua angka, yaitu koordinat horizontal dan vertikalnya (atau baris dan kolomnya). Struktur seperti ini adalah tipikal penggunaan array dua dimensi. Koordinat horizontal adalah indeks barisnya sedangkan koordinat vertical adalah indeks kolomnya. Bentuk array multidimensi ini dapat dikembangkan menjadi lebih dari dua dimensi. Pernyataan berikut ini akan membuat array multidimensi yang memiliki 1000 elemen (10x10x10)

Dim Matrix(9,9,9)

Kadangkala kita tidak tahu seberapa banyak elemen yang harus kita tetapkan. Untuk mengatasi hal ini kita tidak perlu membuat elemen semaksimal mungkin karena akan membutuhkan memori yang relative besar. Kita dapat

mendeklarasikan variable ini menjadi array dinamis. Ukuran dari array dinamis ini dapat beragam sepanjang program dijalankan atau tergantung dari pengguna ketika meng-inputkan ukuran array. Dengan menggunakan array dinamis kita dapat menghapus data yang tidak perlu sehingga lebih menghemat penggunaan sumber daya (memori). Untuk membuat array dinamis dapat dengan menggunakan pernyataan berikut :

```
Dim Matrix()
```

Pada pernyataan tersebut ukuran array tidak kita tentukan. Apabila kita ingin menggunakan variable tersebut kita dapat menggunakan pernyataan sebagai berikut :

```
ReDim Matrix(9,9,9)
```

Penerapan Algoritma Dalam Bahasa Pemrograman

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjalankan dan menggunakan perangkat lunak pemrograman berbasis teks (Python).
 - Memahami pernyataan input dan output
 - Menerapkan struktur algoritma pada bahasa pemrograman.
 - Membuat program sederhana
-

Bab ini secara khusus membahas penerapan struktu algoritma pada bahasa pemrograman. Sampai dengan Bab 10, sebenarnya kita belum sama sekali menggunakan bahasa pemrograman. Kita baru membahas bagaimana membuat algoritma penyelesaian suatu masalah dengan benar. Untuk dapat dieksekusi oleh komputer maka algoritma ini harus kita terjemahkan dalam kode-kode program yang dimengerti komputer. Bahasa pemrograman yang digunakan dalam bab ini adalah Python dan perangkat lunak pengembangnya adalah IDLE. Pemilihan pada bahasa pemrograman ini akan dijelaskan pada bagian pertama dari bab ini.

11.1. SEKILAS TENTANG BAHASA PEMROGRAMAN PYTHON

Ada beberapa alasan mengapa Python dipilih sebagai bahasa pemrograman pada buku ini. Yang pertama, banyak kajian dan penelitian yang menunjukkan bahwa Python merupakan bahasa pemrograman yang dirancang dengan baik untuk mudah dipelajari oleh pemula di bidang pemrograman. Hal ini juga merupakan tujuan awal dari pembuat bahasa Python yaitu Guido van Rossum. Alasan yang kedua adalah karena Python mendukung banyak paradigma pemrograman (lihat Bab 7). Dari pemrograman procedural, terstruktur sampai pemrograman berorientasi obyek. Selain itu tersedia library yang sangat banyak yang dapat digunakan untuk mendukung paradigma pemrograman yang dipilih. Alasan yang ketiga, Python tersedia pada hampir semua platform sistem operasi seperti, Windows, Linux dan keluarga Unix, Mac OS X dan lain-lain. Alasan terakhir

adalah karena Python bersifat open source. Hal ini berarti Python dapat bersifat terbuka dan dapat digunakan tanpa khawatir dengan biaya lisensi. Dengan begitu banyak keuntungan, maka Python merupakan pilihan bahasa pemrograman yang sangat baik.

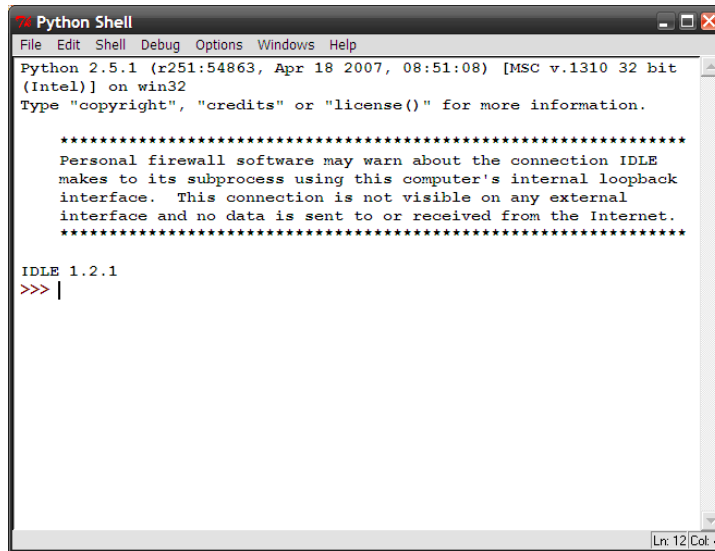
Python merupakan salah satu bahasa pemrograman yang perkembangannya sangat cepat. Meskipun bila dibandingkan dengan bahasa pemrograman seperti C++, BASIC, JAVA atau PASCAL, Python merupakan bahasa pemrograman yang relative lebih muda, namun popularitasnya hampir menyamai. Beberapa aplikasi yang menggunakan Python antara lain: aplikasi instalasi pada Redhat (salah satu distribusi utama Linux), berbagai servis internet pada Yahoo!, sistem control pada program penerbangan NASA, web service Zope dan masih banyak lagi.

11.1.1. Menjalankan Python

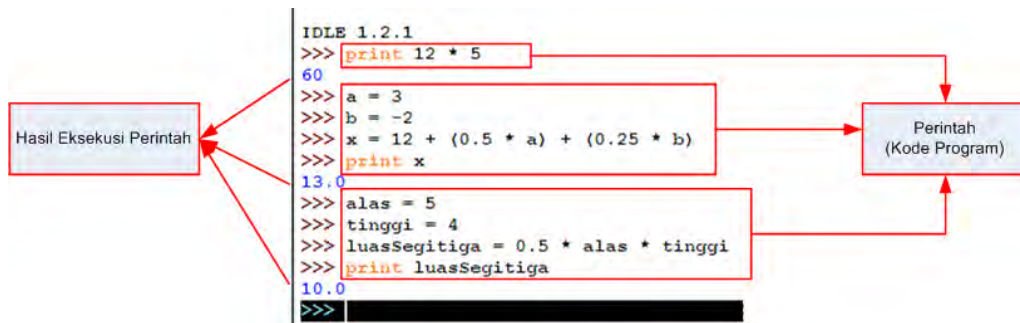
Untuk dapat menjalankan Python, maka yang harus dilakukan adalah menginstal lebih dulu program Python. File instalasi Python dapat di-download dari situs resmi Python (www.python.org). Apabila kita menggunakan system operasi Windows, kita dapat memilih file binary untuk Windows. Bila kita menggunakan Linux, biasanya Python telah terinstall secara default. Selain bahasa pemrograman Python, pada paket yang di-download biasanya juga telah termasuk Python Interpreter, IDLE (Python GUI), dan Manual. Pada bagian berikut dan seterusnya kita akan menggunakan IDLE yang merupakan perangkat lunak pengembang untuk membuat program dalam bahasa Python. Setelah download berhasil, Python dapat diinstal Python dengan cara yang sama seperti menginstal program-program lain.

Pada Microsoft Windows, untuk menjalankan Python dapat dilakukan dengan memilih Start → Programs → Python → IDLE. Jendela IDLE akan terbuka seperti tampak pada Gambar 11.1. Pada jendela ini kita dapat langsung menuliskan kode program dan langsung mengeksekusinya untuk mengetahui hasilnya. Untuk mengeksekusi kode program, Python menggunakan mesin penerjemah yang termasuk dalam kategori interpreter (untuk pengertian interpreter, lihat kembali Bab 7). Pada Gambar 11.1 terlihat ada tanda >>> (tanda lebih besar tiga buah). Tanda ini menunjukkan prompt utama dari Python. Kode atau perintah kita ketikkan setelah tanda tersebut.

Python menyediakan dua mode yang berbeda untuk bekerja dalam lingkungan Python, yaitu mode interaktif dan mode skrip. Pada mode interaktif, kita dapat bekerja seperti ketika bekerja dengan CLI system operasi. Setiap perintah yang kita ketikkan dapat langsung dieksekusi dan diperoleh hasilnya. Pada Gambar 11.2, terlihat bagaimana mode interaktif digunakan.



Gambar 11.1. Jendela IDLE atau Python Shell.

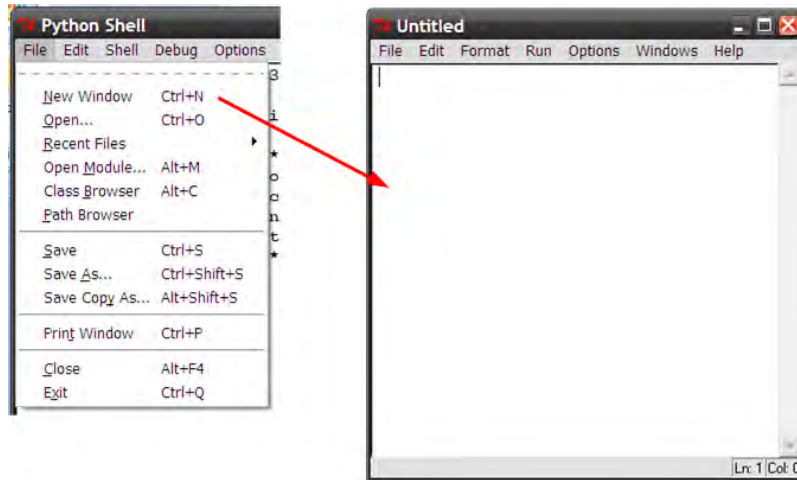


Gambar 11.2. Mode interaktif.

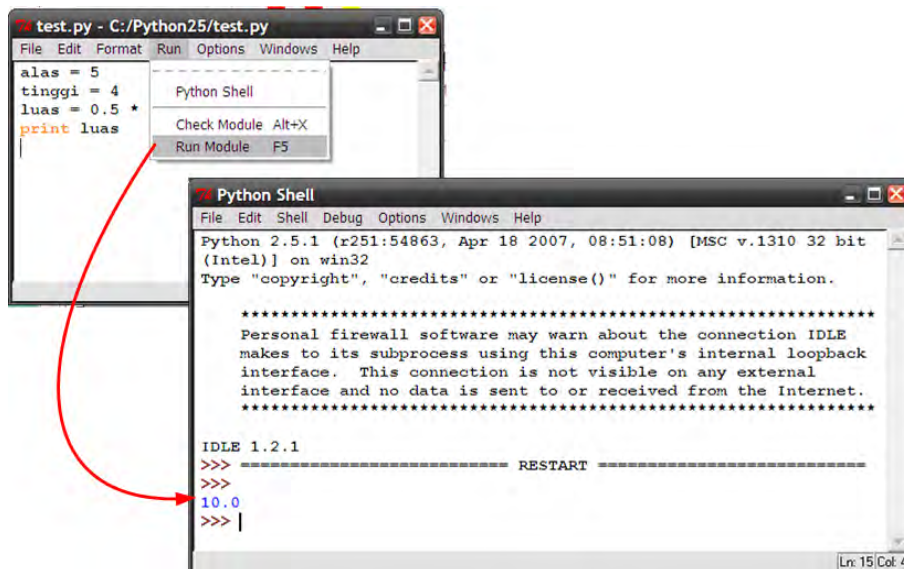
Gambar 11.2 menunjukkan bagaimana mudahnya membuat program sederhana dengan interpreter Python. Pada baris pertama kita dapat menggunakan Python seperti sebuah kalkulator tanpa harus dipusingkan dengan variable atau konstanta. Pada kelompok kode program kedua, kita menggunakan beberapa variable seperti *a*, *b* dan *x*. Kelompok baris ketiga adalah baris-baris kode untuk menyelesaikan perhitungan luas segitiga. Variable yang digunakan adalah *alas*, *tinggi* dan *luasSegitiga*. Untuk mencetak hasil, digunakan perintah **print**.

Berbeda dengan mode interaktif, pada mode skrip, kita buat dulu kode-kode programnya kemudian baru dieksekusi pada interpreter Python. Kode-kode program ini harus disimpan dalam bentuk file dengan ekstensi *.py*. Pada IDLE untuk membuat file kode program dapat dilakukan dengan memilih

menu File kemudian click New Window (Gambar 11.3). Jendela baru akan terbuka dan kita dapat menuliskan kode program yang kita inginkan. Setelah kita mengetikkan kode program, kita dapat menyimpan file dengan memilih menu File → Save. File yang telah kita simpan dapat kita eksekusi dengan memilih menu Run → Run Module. Hasil akan ditampilkan pada jendela Python Shell (lihat Gambar 11.4).



Gambar 11.3. Jendela editor baru pada IDLE.



Gambar 11.4. Proses eksekusi kode program pada mode skrip.

11.2. VARIABEL, KONSTANTA, TIPE DATA DAN OPERATOR

Variabel, konstanta, dan operator yang digunakan dalam Python, tidak jauh berbeda dengan apa yang terdapat pada Bab 8. Namun untuk tipe data ada beberapa perbedaan dan kekhususan yang dimiliki oleh Python.

- Variable

Mendefinisikan variabel dalam Python sangat mudah. Kita hanya perlu mengisi nilai pada sebuah variabel dengan tipe data yang inginkan. Untuk mengisi data pada variable digunakan tanda sama-dengan (=). Hampir sama dengan BASIC, Visual Basic, PHP atau bahasa interpreter lainnya, Python tidak mewajibkan (tidak memerlukan) deklarasi variable sebelum program dieksekusi. Python juga membolehkan merubah tipe data untuk suatu variable ketika program dieksekusi. Python juga bersifat case sensitive (huruf besar dan huruf kecil dianggap berbeda). C akan berbeda dengan c. Perhatikan contoh berikut ini.

Contoh 11.1. Penggunaan variable

```
IDLE 1.2.1
>>> A = 4
>>> B = 12.6
>>> print A
4
>>> print B
12.6
>>> C = 'Phyton itu mudah'
>>> C = A + B
>>> print C
16.6
>>>
```

Pada contoh ini terlihat bahwa kita mendefinisikan A sebagai integer dan B sebagai real (floating). Sedangkan C, awalnya kita definisikan sebagai string kemudian kita ubah menjadi real. Ketika kita cetak maka tipe data C mengikuti tipe data paling akhir yang digunakan.

- Tipe data

Pada Python juga dikenal tipe data numerik, string dan array. Namun Python juga menambahkan tipe data dictionary dan tuple. Tipe data numeric, baik integer maupun real hamper sama dengan yang telah dijelaskan pada Bab 8. Penentuan tipe data numerik yang tepat akan memberikan hasil yang tepat dan penggunaan memori yang efisien. Perhatikan contoh berikut.

Contoh 11.2. Penggunaan tipe data numerik.

```
>>> A = 25
>>> B = 4
>>> C = A / B
>>> print C
6
>>> A = 25.0
>>> B = 4.0
>>> C = A / B
>>> print C
6.25
>>>
```

Pada contoh ini terlihat perbedaan hasil yang terjadi apabila tipe data yang digunakan berbeda. Pada bagian atas, kita mendefinisikan A dan B sebagai integer (tanpa angka desimal). Hasil yang diperoleh, yaitu C menjadi juga bernilai integer. Pada bagian kedua kita mendefinisikan A dan B sebagai real (dengan angka desimal). Hasil yang diperoleh, yaitu C menjadi juga bernilai real.

Tipe data string secara umum dapat ditulis dengan cara diapit tanda petik tunggal atau diapit tanda petik ganda. Kita juga dapat mengatur tampilan menjadi berpindah baris dengan menggunakan tanda `\n`. Perhatikan contoh berikut.

Contoh 11.3. Penggunaan tipe data string.

```
>>> kalimat = 'Saya berteriak, "Awat, ada Python !!!"'
>>> print kalimat
Saya berteriak, "Awat, ada Python !!!"
>>> kalimat_panjang = 'Baris ini akan diatur menjadi \n 2 baris yang s
ama panjang'
>>> print kalimat_panjang
Baris ini akan diatur menjadi
 2 baris yang sama panjang
```

Kita dapat menggabungkan isi variable string dengan menggunakan operator concatenation yaitu tanda `+`. Kita juga dapat mengulang suatu string dengan menggunakan tanda `*`. Berikut contoh menggunakan operator concatenation.

Contoh 11.4. Penggunaan operator `+` dan `*` pada string.

```
>>> kalimat_1 = "Python "
>>> kalimat_2 = "ular kesayanganku"
>>> print kalimat_1 + kalimat_2
Python ular kesayanganku

>>> kata = 'Python'
>>> print kata * 3
PythonPythonPython
>>>
```

Tipe data array dalam Python disebut sebagai List. List adalah jenis data campuran yang bisa memiliki komponen penyusun yang berbeda-beda. Sebuah list dapat dibuat dengan menggunakan tanda kurung

siku, []. Anggota list didaftar dalam kurung siku tersebut dan masing-masing dipisahkan oleh tanda koma. Perhatikan contoh berikut.

Contoh 11.5. Penggunaan list.

```
>>> Himpunan = [2,4,6,8]
>>> print Himpunan
[2, 4, 6, 8]
>>> print Himpunan[0]
2
>>> print Himpunan[2],Himpunan[3]
6 8
>>> Himpunan[1] = Himpunan[1] + 10
>>> print Himpunan
[2, 14, 6, 8]
>>> Himpunan[2] = Himpunan[0] + Himpunan[1]
>>> print Himpunan
[2, 14, 16, 8]
>>>
```

Pada contoh ini kita mendefinisikan Himpunan sebagai List dengan empat anggota yaitu [2, 4, 6, 8] (lihat baris 1 sampai dengan 3). Untuk memanggil anggota kita cukup menyebutkan indexnya saja. Index dalam List selalu dimulai dengan angka 0. Pada baris ke empat, perintah `print Himpunan[0]` akan menghasilkan output 2. Karena 2 adalah anggota pertama (indexnya 0). Pada baris-baris berikutnya merupakan contoh bagaimana merubah isi data dari List.

11.3. PERINTAH DASAR INPUT - OUTPUT

INPUT dan OUTPUT merupakan awal dan akhir pengolahan suatu data. Tanpa proses INPUT, kita tidak dapat memperoleh data untuk diolah dan data-data yang telah selesai diolah tidak akan berarti tanpa proses OUTPUT.

Perintah (fungsi) yang dapat digunakan untuk memperoleh input dari pengguna adalah `input`. Perhatikan contoh berikut ini.

Contoh 11.5. Penggunaan fungsi `input`.

```
print 'Program Menghitung Luas Segitiga'
alas = input('Nilai alas = ')
tinggi = input('Nilai tinggi = ')
luas = 0.5 * (alas * tinggi)
print 'Luas segitiga = ', luas
```

Pada contoh ini kita menggunakan mode skrip. Perintah `input` digunakan untuk mendapat masukan dari pengguna berupa nilai alas dan tinggi. Jika kita eksekusi maka kita akan memperoleh tampilan sebagai berikut.

```

>>> ===== RESTART :
>>>
Program Menghitung Luas Segitiga
Nilai alas = 3.0
Nilai tinggi = 4.5
Luas segitiga = 6.75
>>>

```

Perintah (fungsi) utama untuk menampilkan hasil (output) adalah `print`. Perintah ini akan menampilkan output pada standard output yaitu terminal. Contoh-contoh penggunaan `print` sebenarnya telah disampaikan pada contoh-contoh sebelumnya (Contoh 11.1 sampai dengan 11.5).

11.4. PENERAPAN STRUKTUR ALGORITMA

Seperti telah dibahas pada Bab 9, ada tiga struktur umum algoritma, yaitu berurutan, pemilihan dan pengulangan. Kita akan membahas satu-persatu penerapan ketiga struktur ini pada bahasa pemrograman Python. Kasus-kasus yang akan kita gunakan adalah kasus yang sama yang ada pada Bab 9.

11.4.1. Struktur Berurutan

Seperti dijelaskan pada Bab 9, struktur berurutan berisi perintah-perintah yang dijalankan berurutan satu persatu. Kita akan coba menerjemahkan algoritma yang ada pada Contoh 9.4 pada Bab 9. Algoritma pertama yaitu menghitung volume balok dan algoritma kedua menghitung luas lingkaran. Dari Gambar 9.3 (Bab 9) kita sebenarnya dapat dengan mudah menerjemahkan ke bahasa pemrograman Python. Yang menjadi perhatian mungkin adalah bagaimana menggunakan perintah input – output pada kode program. Perhatikan contoh-contoh kode program berikut.

Contoh 11.6. Kode program untuk menghitung volume balok.

```

print 'Program Menghitung Volume Balok'
panjang = input('Masukkan nilai panjang : ')
lebar = input('Masukkan nilai lebar : ')
tinggi = input('Masukkan nilai tinggi : ')
volume = panjang * lebar * tinggi
print 'Volume balok adalah : ', volume

```

Ada empat variable yang kita definisikan pada program ini. Variabel `panjang`, `lebar` dan `tinggi` adalah variable input, sedangkan `volume` adalah variable output. Baris-baris kode program di atas akan dieksekusi baris per baris. Contoh output dari eksekusi program adalah sebagai berikut :

```
>>> ===== RESTART :
>>>
Program Menghitung Volume Balok
Masukkan nilai panjang : 5
Masukkan nilai lebar : 5
Masukkan nilai tinggi : 3
Volume balok adalah : 75
>>>
```

Contoh eksekusi dengan input data yang lain :

```
>>> ===== RESTART :
>>>
Program Menghitung Volume Balok
Masukkan nilai panjang : 7
Masukkan nilai lebar : 3.7
Masukkan nilai tinggi : 5.6
Volume balok adalah : 145.04
>>>
```

Contoh 11.7. Kode program untuk menghitung luas lingkaran.

```
print 'Program Menghitung Luas Lingkaran'
nilai_pi = 3.14
radius = input('Masukkan nilai jari-jari lingkaran : ')
luas = nilai_pi * radius * radius
print 'Luas lingkaran adalah : ', luas
```

Ada dua variable yang kita definisikan pada program ini. Variabel radius adalah variable input, sedangkan luas adalah variable output. Nilai_pi pada kode program di atas adalah konstanta. Hal ini karena nilai_pi tidak akan berubah nilai datanya. Contoh output dari eksekusi program adalah sebagai berikut :

```
>>> ===== RESTART ==
>>>
Program Menghitung Luas Lingkaran
Masukkan nilai jari-jari lingkaran : 4.5
Luas lingkaran adalah : 63.585
>>>
```

Contoh eksekusi dengan input data yang lain :

```
>>> ===== RESTART ==
>>>
Program Menghitung Luas Lingkaran
Masukkan nilai jari-jari lingkaran : 3
Luas lingkaran adalah : 28.26
>>>
```

Pada kode program di atas, kita mendefinisikan sendiri nilai konstanta pi. Sebenarnya kita dapat menggunakan nilai pi yang telah didefinisikan dalam fungsi-fungsi built-in Python. Perhatikan contoh kode program berikut:

```
import math
print 'Program Menghitung Luas Lingkaran'
radius = input('Masukkan nilai jari-jari lingkaran : ')
luas = math.pi * radius * radius
print 'Luas lingkaran adalah : ', luas
```

Baris pertama yaitu `import math`, merupakan perintah untuk memuat module `math` (module yang berisi fungsi-fungsi matematika) ke dalam kode program. Kemudian untuk memanggil nilai pi kita menggunakan format perintah `math.pi` (baris ke 5). Kalau kita eksekusi kode program tersebut maka hasilnya adalah sebagai berikut.

```
>>> ===== RESTART =====
>>>
Program Menghitung Luas Lingkaran
Masukkan nilai jari-jari lingkaran : 4.5
Luas lingkaran adalah : 63.6172512352
>>>
```

11.4.2. Struktur Pemilihan

Dengan perintah struktur pemilihan seorang programmer dapat menentukan jalur-jalur proses yang harus dikerjakan oleh computer berdasarkan logika tertentu. Pada Python, perintah untuk struktur pemilihan adalah `if`. Format umum untuk struktur pemilihan dengan `if` adalah sebagai berikut:

```
if kondisi:
    perintah_jika_kondisi_benar
elif kondisi_lain:
    perintah_jika_kondisi_lain_benar
else:
    perintah_jika_tak_ada_kondisi_yang_benar
```

Seperti halnya struktur berurutan, berikut ini kita akan menggunakan contoh-contoh kasus pada Bab 9 untuk kita terapkan pada bahasa pemrograman Python.

Contoh 11.8. Kode program untuk algoritma contoh 9.7.

Pada Bab 9 contoh 9.7, kalau kita terjemahkan dalam kode program, maka akan tampak sebagai berikut :

```
print 'Struktur if sedernana'
umur = input('Masukkan umur anda : ')
if umur > 17:
    print 'Anda berumur', umur, 'tahun, silahkan nonton film ini'
else:
    print 'Anda berumur', umur, 'tahun, tidak boleh nonton film ini'
```

Pada kode program di atas, karena hanya ada dua pilihan maka kita langsung menggunakan `else`, tanpa menggunakan `elif`. Apabila kita eksekusi maka hasilnya akan sebagai berikut:

```
>>> ===== RESTART =====
>>>
Struktur if sedernana
Masukkan umur anda : 16
Anda berumur 16 tahun, tidak boleh nonton film ini
>>>
```

Output di atas adalah jika kita masukkan umur < 17 tahun. Jika kita masukkan umur > 17 tahun, maka akan tampak seperti berikut:

```
>>> ===== RESTART =====
>>>
Struktur if sedernana
Masukkan umur anda : 21
Anda berumur 21 tahun, silahkan nonton film ini
>>>
```

Contoh 11.9. Kode program untuk algoritma contoh 9.8.

Pada Bab 9 contoh 9.8, algoritma sedikit lebih rumit karena melibatkan perhitungan. Kode program untuk algoritma tersebut adalah sebagai berikut:

```
print 'Struktur if untuk menghitung nilai Q'
X = input('Masukkan nilai X : ')
Y = input('Masukkan nilai Y : ')
P = X + Y
print 'Nilai P = ', P
if P >= 0:
    Q = X * Y
else:
    Q = X / Y
print 'Nilai Q = ', Q
```

Bila kita eksekusi kode program di atas dengan input X = 4.0 dan Y = 6.0, maka hasilnya akan seperti berikut:

```
>>> ===== RESTART :
>>>
Struktur if untuk menghitung nilai Q
Masukkan nilai X : 4.0
Masukkan nilai Y : 6.0
Nilai P = 10.0
Nilai Q = 24.0
>>>
```

Sedangkan bila nilai X = 4.0 dan Y = -6.0, maka hasilnya akan sebagai berikut:

```

>>> ===== RESTART :
>>>
Struktur if untuk menghitung nilai Q
Masukkan nilai X : 4.0
Masukkan nilai Y : -6.0
Nilai P = -2.0
Nilai Q = -0.666666666667
>>>

```

Pada dua contoh di atas (contoh 11.8 dan 11.9) kita hanya menggunakan struktur if dalam bentuk yang sederhana. Contoh berikut akan menunjukkan bagaimana struktur if bersarang.

Contoh 11.10. Kode program untuk algoritma contoh 9.9.

Pada Bab 9 contoh 9.9, algoritma pemilihan menggunakan model bersarang, karena ada struktur pemilihan bertingkat. Kode program untuk algoritma tersebut adalah sebagai berikut:

```

print 'Struktur if bersarang'
status = raw_input('Masukkan status (P=Pelanggan, N=Non Pelanggan) : ')
JLF = input('Masukkan jumlah lembar fotokopi : ')
if status == 'P':
    HPP = 75.0
elif status == 'N':
    if JLF < 100:
        HPP = 100.0
    else:
        HPP = 85.0
TH = JLF * HPP
print 'Total harga fotokopi = Rp.', TH

```

Pada baris ke-2 kode program di atas, kita menggunakan perintah `raw_input` karena kita membutuhkan input dengan tipe data string. Berbeda dengan tipe data numeric yang cukup menggunakan perintah `input`. Selain itu kita juga melihat ada operator `==` (tanda sama dengan 2 buah) pada baris ke-4 dan ke-6. Tanda `==` adalah tanda untuk melakukan perbandingan. Biasanya digunakan dalam struktur if untuk menguji apakah suatu nilai variable sama dengan nilai tertentu. Kebalikan dari tanda `==` adalah `!=` (tidak sama dengan).

Perhatikan pada baris ke-6 sampai dengan ke-10 (`elif status ..` dan seterusnya). Pada baris-baris kode tersebut kita menggunakan struktur pemilihan bersarang. If yang pertama adalah `elif status == 'N'` dan if berikutnya adalah `if JLF < 100`.

Apabila kita eksekusi maka contoh hasilnya adalah sebagai berikut:


```
>>> ===== RESTART =====
>>>
Struktur if bersarang
Masukkan status (P=Pelanggan, N=Non Pelanggan) : N
Masukkan jumlah lembar fotokopi : 145
Total harga fotokopi = Rp. 12325.0
>>>
```

Contoh hasil eksekusi yang lain adalah :

```
>>> ===== RESTART =====
>>>
Struktur if bersarang
Masukkan status (P=Pelanggan, N=Non Pelanggan) : P
Masukkan jumlah lembar fotokopi : 145
Total harga fotokopi = Rp. 10875.0
>>>
```

Sebenarnya ada kelemahan yang cukup mendasar pada kode program di atas. Bagaimana jika pengguna program memasukkan input status bukan 'P' atau 'N'. Perhatikan output program berikut ini.

```
>>> ===== RESTART =====
>>>
Struktur if bersarang
Masukkan status (P=Pelanggan, N=Non Pelanggan) : Y
Masukkan jumlah lembar fotokopi : 145

Traceback (most recent call last):
  File "E:\draft RPL\script\python\sc06.py", line 11, in <module>
    TH = JLF * HPP
NameError: name 'HPP' is not defined
>>>
```

Pada output program ini kita memasukkan status pelanggan dengan huruf 'Y'. Kita lihat ada petunjuk kesalahan bahwa pada baris ke-11, variable HPP tidak didefinisikan. Sebenarnya kesalahan bukan pada baris ke-11, namun pada baris ke-2 karena kita tidak mengantisipasi kemungkinan pengguna memasukkan huruf yang lain selain 'P' dan 'N'. Bagaimana memperbaiki kode program di atas? Silahkan mencoba-coba dengan menggunakan struktur **if** kemudian digabung dengan operator logika **or**.

Contoh 11.10. Kode program untuk algoritma contoh 9.10.

Pada Bab 9 contoh 9.10, algoritma pemilihan juga menggunakan model bersarang, namun sedikit lebih dikembangkan. Kode program untuk algoritma tersebut adalah sebagai berikut:

```

print 'Nilai Ujian Mata Pelajaran Pemrograman Web'

NIM = raw_input('Nomor Induk Siswa : ')
nama = raw_input('Nama Siswa : ')
NUTS = input('Nilai Ujian Tengah Semester : ')
if NUTS > 70:
    NA = NUTS
    status = 'Lulus'
else:
    NUAS = input('Nilai Ujian Akhir Semester : ')
    NA = (NUTS * 0.4) + (NUAS * 0.6)
    if NA >= 60:
        status = 'Lulus'
    else:
        status = 'Tidak Lulus'

# format output
print ''
print ''
print 'Hasil Akhir :'
print '%-8s%-20s%-5s%-12s' % ('NIM', 'Nama', 'NA', 'Status')

length = len('%-8s%-20s%-5s%-15s' % ('NIM', 'Nama', 'NA', 'Status'))
print '=' * length
print '%-8s%-20s%-5s%-12s' % (NIM, nama, NA, status)

```

Pada kode program di atas kita melihat ada struktur if bersarang. Selain itu pada kode di atas juga diperkenalkan bagaimana memformat keluaran pada terminal agar tampak lebih enak dibaca. Perhatikan pada bagian-bagian akhir kode program (dimulai dari baris # format output dan seterusnya).

Hasil eksekusi program akan tampak sebagai berikut:

```

>>> ===== RESTART =====
>>>
Nilai Ujian Mata Pelajaran Pemrograman Web
Nomor Induk Siswa : S0198
Nama Siswa : Theo Walcott
Nilai Ujian Tengah Semester : 45
Nilai Ujian Akhir Semester : 34

Hasil Akhir :
NIM      Nama              NA      Status
=====
S0198    Theo Walcott        38.4    Tidak Lulus
>>>

```

Output di atas menunjukkan jika Nilai Ujian Tengah Semester kurang dari 70, maka program akan meminta input Nilai Ujian Akhir Semester. Bagaimana jika Nilai Ujian Tengah Semester lebih dari 70. Perhatikan output berikut ini.

```

>>> ===== RESTART =====
>>>
Nilai Ujian Mata Pelajaran Pemrograman Web
Nomor Induk Siswa : S0178
Nama Siswa : Denilson
Nilai Ujian Tengah Semester : 83

Hasil Akhir :
NIM      Nama      NA      Status
=====
S0178    Denilson      83      Lulus
>>>

```

Python tidak menyediakan fasilitas untuk melakukan pilihan dengan model select .. case seperti pada BASIC, Visual Basic atau Delphi. Sehingga jika ada banyak pilihan maka tetap pola if ... elif ... else yang dipakai. Perhatikan contoh berikut ini.

Contoh 11.11. Kode program untuk struktur if dengan banyak pilihan

```

print 'Struktur If dengan banyak pilihan'

Hari = input("Masukkan Nomor Hari (1-7) ")

if Hari == 1:
    print "Minggu"
elif Hari == 2:
    print "Senin"
elif Hari == 3:
    print "Selasa"
elif Hari == 4:
    print "Rabu"
elif Hari == 5:
    print "Kamis"
elif Hari == 6:
    print "Jumat"
elif Hari == 7:
    print "Sabtu"
else:
    print "Tidak ada hari dengan nomor = ", Hari

```

Pada contoh kode ini tampak ada banyak pilihan. Pola if ... elif ... else tetap harus digunakan untuk menyelesaikan kasus di atas. Output dari kode program di atas adalah sebagai berikut.

```

>>> ===== RESTART =====
>>>
Struktur If dengan banyak pilihan
Masukkan Nomor Hari (1-7) 5
Kamis
>>>
>>>
>>> ===== RESTART =====
>>>
Struktur If dengan banyak pilihan
Masukkan Nomor Hari (1-7) 9
Tidak ada hari dengan nomor = 9
>>>

```

11.4.3. Struktur Pengulangan

Salah satu kelebihan komputer daripada manusia adalah ia tidak pernah bosan terhadap tugas-tugas rutin yang berulang-ulang. Inilah yang menyebabkan komputerisasi selalu diterapkan terhadap tugas-tugas yang rutin. Dan disinilah struktur pengulangan menjadi penting perannya. Secara umum Python memberikan dua model dalam struktur pengulangan, yaitu model pengulangan dengan for dan model pengulangan dengan while. Seperti telah dijelaskan pada Bab 9, model for akan cocok jika kita tahu lebih dulu berapa kali proses pengulangan dilakukan, sedangkan model while bila kita tidak tahu berapa kali proses pengulangan dilakukan, namun kita tahu kondisi yang menyebabkan pengulangan berhenti.

Format umum untuk model for adalah sebagai berikut :

```

for <target> in <object>:
    badan loop
else:
    perintah lain

```

Sedangkan format umum untuk model while adalah sebagai berikut:

```

while <kondisi>:
    badan loop yang dieksekusi jika <kondisi> benar
else:
    perintah lain

```

Kita akan mencoba dua model di atas dengan beberapa contoh sederhana kemudian kita coba dengan kasus pada Bab 9.

Contoh 11.12. Kode program untuk struktur pengulangan for ke-1.

```

print 'Struktur for sederhana'

for I in range(1,5):
    print 'Looping itu mudah'

```

Kode program ini adalah kode program untuk mencetak kalimat 'Looping itu mudah' berulang kali. Kita tidak menggunakan else seperti pada format umum, karena else bersifat optional. Berapa kali pengulangan dilakukan? Kita akan lihat hasilnya dengan mengeksekusi kode program di atas.

```
>>> ===== RESTART
>>>
Struktur for sederhana
Looping itu mudah
Looping itu mudah
Looping itu mudah
Looping itu mudah
>>>
```

Hasil dari eksekusi menunjukkan hanya ada 4 kali pengulangan. Hal ini karena dalam Python, angka batas atas pada range (pada contoh di atas adalah 5) tidak termasuk dalam anggota range.

Contoh 11.13. Kode program untuk struktur pengulangan for ke-2.

Pada contoh 11.12, kita menggunakan range sebagai object. Kita bisa juga menggunakan list sebagai object, seperti contoh berikut ini.

```
print 'Struktur for ke-2'

total = 0
for x in [1, 2, 3, 4]:
    total = total + x
    print 'total ke -', x, '=', total
```

Pada kode program ini kita menggunakan list yang berisi [1, 2, 3, 4]. Pengulangan akan dilakukan sebanyak 4 kali karena ada 4 anggota dalam list. Pada kode program di atas kita juga mengenal apa yang disebut inisialisasi, yaitu pemberian nilai awal pada suatu variable. Variable total diberi nilai awal 0. Hasil eksekusi dari kode program tersebut adalah sebagai berikut:

```
>>> ===== RESTART =
>>>
Struktur for ke-2
total ke - 1 = 1
total ke - 2 = 3
total ke - 3 = 6
total ke - 4 = 10
>>>
```

Output tersebut dapat menunjukkan bagaimana proses pengulangan terjadi. Pada pengulangan pertama, x akan berisi nilai pertama dari list yaitu satu. Variable total awal akan berisi nilai 0. Sehingga pernyataan total = total + x akan menghasilkan nilai 1. Pada pengulangan ke-2, x berisi nilai ke-2 dari list, yaitu 2. Sedangkan total sudah berisi nilai 1. Sehingga pengulangan ke-2 adalah 1 + 2 = 3. Demikian seterusnya.

Contoh 11.14. Kode program untuk struktur pengulangan for ke-3.

Misalkan kita diminta untuk menampilkan anggota suatu himpunan seperti berikut (2, 4, 6, 8, 10, 12, ... , 30). Bagaimanakah kode programnya? Kita dapat menggunakan for untuk kasus ini. Pola anggota himpunan dapat kita ketahui dengan mudah yaitu kenaikan nilai sebanyak 2. Kode program adalah sebagai berikut:

```
print 'Anggota himpunan'

for anggota in range (2, 31, 2):
    print anggota,
```

Kita dapat dengan mudah membuat pola bilangan di atas dengan menggunakan range. Sebelumnya kita telah menggunakan range, namun hanya menetapkan batas bawah dan atas. Range mempunyai satu parameter lain, yaitu step. Sehingga range (2, 31, 2) berarti batas bawah 2, batas atas 31 dan step (atau kenaikan) adalah 2. Jika kita eksekusi maka hasilnya tampak seperti berikut:

```
>>> ===== RESTART =====
>>>
Anggota himpunan
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
>>>
```

Contoh 11.15. Kode program untuk perpotongan dua himpunan.

Pada contoh ini kita akan menggunakan algoritma yang sudah kita selesaikan pada contoh 9.13 pada Bab 9. Yang akan kita buat ini juga merupakan pengembangan dari Contoh 11.14. Kode program untuk algoritma ini adalah sebagai berikut:

```
print 'Himpunan A = (1, 3, 5, ..., 19)'
print 'Himpunan B = (2, 5, 8, ..., 29)'
print 'Perpotongan Himpunan A dan B adalah :',
for A in range(1,20,2):
    for B in range(2,30,3):
        if A == B:
            print A,
```

Untuk menyelesaikan kasus ini, pada kode program di atas, kita menggunakan nested looping (pengulangan bersarang). For yang pertama adalah untuk himpunan A dan for yang kedua adalah untuk himpunan B. Perpotongan himpunan A dengan himpunan B terjadi jika ada anggota himpunan A yang juga anggota himpunan B. Atau pada

kode program di atas dinyatakan dengan `if A == B`. Hasil eksekusi program adalah sebagai berikut:

```
>>> ===== RESTART =====
>>>
Himpunan A = (1, 3, 5, ..., 19)
Himpunan B = (2, 5, 8, ..., 29)
Perpotongan Himpunan A dan B adalah : 5 11 17
>>>
```

Contoh 11.16. Kode program untuk mencari fungsi Y..

Pada contoh ini kita akan menggunakan algoritma pada contoh 9.16 pada Bab 9. Kita akan mencari nilai-nilai Y berdasarkan berapa nilai X nya. Kode programnya adalah sebagai berikut:

```
print 'Nilai Y untuk fungsi Y = -0.910x^2 + 6.775x - 2.900'
print 'untuk nilai X antara 2 sampai dengan 5'
print ''
print '%-4s%-5s' % ('X','Y')
length = len('%-4s%-5s' % ('X','Y'))
print '=' * length
for X in range(2,6):
    Y = -0.910*X**2 + 6.775*X - 2.900
    print '%-4s%-5s' % (X,Y)
```

Hasil eksekusinya seperti berikut ini.

```
>>> ===== RESTART =====
>>>
Nilai Y untuk fungsi Y = -0.910x^2 + 6.775x - 2.900
untuk nilai X antara 2 sampai dengan 5

X    Y
=====
2    7.01
3    9.235
4    9.64
5    8.225
>>>
```

Contoh 11.12 sampai dengan 11.15 menunjukkan bagaimana kita menggunakan model `for` untuk mengulang suatu perintah. Ada banyak variasi yang dapat dikembangkan dengan model `for`. Bagi pembaca dipersilahkan untuk mencoba dan mencoba dengan berbagai macam kasus untuk meningkatkan kemampuan.

Model pengulangan dengan `while` sebenarnya lebih fleksibel daripada model `for`. Karena kita tidak perlu menetapkan jumlah pengulangan, cukup syarat pengulangan saja. Contoh-contoh berikut menunjukkan bagaimana kita menggunakan model `while`.

Contoh 11.17. Kode program model while untuk contoh 11.12..

Pada contoh ini kita akan menggunakan while untuk kasus yang sama seperti Contoh 11.12. Kode programnya adalah sebagai berikut:

```
print 'Pengulangan dengan while ke - 1'
print ''
I = 1
while I < 5:
    print 'Looping itu mudah'
    I = I + 1
```

Pada kode program di atas, $I < 5$ adalah kondisi. Sebelumnya kita inisialisasi $I = 1$. Pernyataan $I = I + 1$, merupakan pernyataan untuk menaikkan nilai I setiap kali pengulangan. Pernyataan ini biasa disebut sebagai counter (pencacah). While akan memeriksa apakah nilai I masih kurang dari 5, jika benar maka pernyataan di bawah baris while akan dieksekusi. Hasil eksekusi kode program di atas adalah sebagai berikut:

```
>>> ===== RESTART ==
>>>
Pengulangan dengan while ke - 1

Looping itu mudah
Looping itu mudah
Looping itu mudah
Looping itu mudah
>>>
```

Kita lihat bahwa hasil eksekusi program di atas, sama persis dengan hasil eksekusi pada Contoh 11.12.

Contoh 11.18. Kode program model while untuk contoh 11.15..

Pada contoh ini kita akan langsung mencoba menggunakan while untuk kasus sedikit rumit yaitu kasus yang sama seperti Contoh 11.15. Kode programnya adalah sebagai berikut:

```
print 'Perpotongan dua himpunan dengan menggunakan while'
print ''
print 'Himpunan A = (1, 3, 5, ..., 19)'
print 'Himpunan B = (2, 5, 8, ..., 29)'
print 'Perpotongan himpunan A dan B adalah : '
A = 1
while A < 20:
    B = 2
    while B < 30:
        if A == B:
            print A,
            B = B + 3
        A = A + 2
```

Pada kode program ini kita juga menggunakan nested looping dengan while. Inisialisasi untuk A kita letakkan sebelum pernyataan while untuk A sedangkan inisialisasi untuk B kita letakkan di atas pernyataan while

untuk B. Apabila kita salah meletakkan inisialisasi hasilnya akan berbeda. Counter untuk A adalah $A = A + 2$ dan B adalah $B = B + 3$. Angka 2 dan 3 tersebut sama dengan step yang ada pada range (lihat kode program pada contoh 11.15). Output dari kode program ini adalah sebagai berikut:

```
>>> ===== RESTART =====
>>>
Perpotongan dua himpunan dengan menggunakan while

Himpunan A = (1, 3, 5, ..., 19)
Himpunan B = (2, 5, 8, ..., 29)
Perpotongan himpunan A dan B adalah :
5 11 17
>>>
```

Contoh 11.19. Kode program model while untuk contoh 9.15..

Contoh ini akan menggunakan kasus yang sama seperti Contoh 9.15 pada Bab 9. Dari algoritma yang sudah dibuat pada Contoh 9.15, kita dapat menerjemahkan menjadi kode program sebagai berikut:

```
print 'Nilai Y untuk fungsi Y = (X - 2) * X'
print ''
print '%-4s%-5s' % ('X', 'Y')
length = len ('%-4s%-5s' % ('X', 'Y'))
print '=' * length
X = 10
Y = 0
while X >= 5:
    Y = (X - 2) * X
    print '%-4s%-5s' % (X, Y)
    X = X - 1
```

Kode program di atas menunjukkan bagaimana kita membuat hitungan mundur untuk counter. X kita inisialisasi dengan nilai 10. Sedangkan pada pernyataan while kita tentukan syaratnya bahwa X harus lebih besar atau sama dengan 5. Pada counter-nya, kita tentukan $X = X - 1$. Artinya akan terjadi pengurangan counter sebesar satu setiap kali terjadi pengulangan. Pengulangan akan berhenti sampai $X = 5$. Perhatikan hasil eksekusi program berikut ini.

```
>>> ===== RESTART =====
>>>
Nilai Y untuk fungsi Y = (X - 2) * X

X      Y
=====
10     80
9       63
8       48
7       35
6       24
5       15
>>>
```

Nilai awal X adalah 10 (hasil dari inisialisasi) kemudian akan terus berkurang sampai X = 5.

11.5. FUNGSI

Fungsi atau pada Bab 8 disebut sebagai prosedur adalah fasilitas yang disediakan untuk membantu programmer dalam mengembangkan program. Dengan adanya fungsi programmer tidak perlu menuliskan kode program berulang kali (lihat pembahasan pada Bab 8).

11.5.1. Fungsi Built-in

Python telah menyediakan fungsi-fungsi built-in dalam standard library yang dapat digunakan untuk berbagai keperluan. Sebenarnya kita telah menggunakan beberapa fungsi ini pada contoh-contoh sebelumnya. Pernyataan-pernyataan seperti, len, input, raw_input, range dan beberapa pernyataan lain, termasuk dalam fungsi. Masih banyak lagi fungsi-fungsi built-in yang dapat digunakan. Untuk memeriksa fungsi-fungsi built-in apa saja yang tersedia dapat digunakan perintah dir seperti berikut ini.

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'ReferenceError', 'RuntimeError', 'RuntimeWarning', 'StandardError', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '_', '__debug__', '__doc__', '__import__', '__name__', 'abs', 'all', 'any', 'apply', 'basestring', 'bool', 'buffer', 'callable', 'chr', 'classmethod', 'cmp', 'coerce', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'execfile', 'exit', 'file', 'filter', 'float', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'intern', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'long', 'map', 'max', 'min', 'object', 'oct', 'open', 'ord', 'pow', 'property', 'quit', 'range', 'raw_input', 'reduce', 'reload', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'unichr', 'unicode', 'vars', 'xrange', 'zip']
>>>
```

Masing-masing fungsi tersebut mempunyai cara tersendiri untuk digunakan. Kita telah melihat bagaimana syntax untuk menggunakan fungsi len, range, input, dan raw_input pada contoh-contoh sebelumnya. Buku ini tidak akan membahas penggunaan semua fungsi-fungsi built-in tersebut. Pembaca disarankan untuk membaca dokumentasi lengkap Python ada pada paket instalasi Python.

11.5.2. Membuat Fungsi

Menggunakan fungsi-fungsi built-in memang sudah sangat membantu, namun pada pemrograman tertentu, kita membutuhkan fungsi yang mungkin tidak tersedia. Python memberikan fasilitas untuk membuat fungsi sendiri. Pada Python untuk mendefinisikan fungsi, formatnya adalah sebagai berikut:

```
def <nama_fungsi> (argument_1, argument_2, ... argument_n):  
    <pernyataan>  
    return
```

Perintah `def` akan mendefinisikan sebuah fungsi yang kita beri nama tertentu. Argument adalah tempat di mana nilai-nilai akan dilewatkan. Sedangkan perintah `return` akan mengembalikan hasil dari fungsi tersebut. Jika `return` tidak ditulis, maka hasil akhir dari fungsi dapat dipakai untuk dalam baris kode setelah fungsi. Untuk lebih jelas perhatikan contoh berikut.

Contoh 11.20. Fungsi perkalian.

```
>>> def kali (x, y):  
        hasil = x * y  
        return hasil
```

Kode program di atas adalah sebuah fungsi sederhana untuk menghitung hasil kali dua bilangan. Fungsi kita beri nama `kali` dengan dua argument yaitu `x` dan `y`. Fungsi akan mengembalikan (`return`) hasil kali dua bilangan tersebut. Untuk menjalankan fungsi tersebut, kita dapat melakukan seperti berikut ini.

```
>>> kali (4, 6)  
24  
>>>  
>>> x = 4  
>>> y = 6  
>>> kali (x, y)  
24  
>>>  
>>> a = 10  
>>> b = 5  
>>> kali (a, b)  
50  
>>>
```

Cara yang pertama menggunakan fungsi adalah dengan memasukkan nilai secara langsung pada argument. Cara yang kedua adalah dengan mendefinisikan dulu variabelnya. Pada contoh di atas terlihat bahwa nama variabel tidak harus sama dengan nama argumentnya, asal ada dua argument sesuai yang ditetapkan pada fungsi.

Contoh 11.21. Fungsi perpotongan dua himpunan.

```
>>> def intersect (seq1, seq2):  
    res = []  
    for x in seq1:  
        if x in seq2:  
            res.append(x)  
    return res
```

Fungsi di atas bernama `intersect`, yang dapat digunakan untuk mencari nilai perpotongan dari dua buah himpunan. Ada dua argument yaitu `seq1` dan `seq2`. Pada baris ke-2 kita buat sebuah variable `res` yang kita definisikan bertipe data list dan dengan anggota list yang masih kosong. Hasil dari fungsi ini akan berupa list. Kalau kita perhatikan dengan seksama algoritma yang digunakan sama persis dengan algoritma pada contoh 11.15. Namun dalam penggunaannya jauh lebih fleksibel. Perhatikan bagaimana cara menggunakan fungsi tersebut.

```
>>> A = range(1,20,2)  
>>> B = range(2,30,3)  
>>> intersect(A, B)  
[5, 11, 17]  
>>>
```

Kita lihat bahwa dengan menggunakan memasukkan argument `A` dan `B` seperti di atas, kita memperoleh hasil yang sama persis dengan hasil dari Contoh 11.15. Fungsi `intersect` ini juga dapat kita gunakan untuk tipe data string, seperti tampak berikut ini.

```
>>> kata1 = 'PYTHON'  
>>> kata2 = 'PITON'  
>>> intersect(kata1, kata2)  
['P', 'T', 'O', 'N']  
>>>
```

Hasil eksekusi menunjukkan sebuah list yang berisi huruf-huruf yang ada baik pada variable `kata1` maupun variable `kata2`.

Dengan mempelajari dua contoh di atas, kita dapat melihat kegunaan fungsi. Fungsi sangat membantu kita, karena kita tidak perlu mengetik ulang kode program yang sama. Pada contoh kedua, kita dapat menggunakan suatu fungsi untuk berbagai tipe data tanpa mengetik ulang. Hanya dengan mengganti isi argument, kita sudah mendapatkan hasil yang kita inginkan.

11.6. MODUL

Modul adalah file yang berisi kode-kode program Python. Modul ini dapat kita muat (load) ke dalam program yang kita buat dan kita gunakan sesuai kebutuhan. Python telah menyediakan banyak sekali modul yang termasuk dalam standard library. Namun bila diperlukan kita dapat membuat sendiri. Modul dapat berisi fungsi-fungsi yang telah kita definisikan. Untuk

memuat suatu modul, kita dapat menggunakan perintah import kemudian diikuti dengan nama file dari modul.

11.6.1. Modul Dalam Standard Library

Python telah menyediakan standard library yang berisi banyak sekali modul jadi yang dapat kita gunakan. Modul-modul ini tersedia untuk berbagai aplikasi, mulai dari matematika, basis data, internet sampai multimedia. Inilah salah satu keuntungan menggunakan Python, yaitu tersedianya modul-modul yang sangat luas penggunaannya. Kita dapat membangun berbagai jenis aplikasi dengan cepat dengan bantuan modul-modul yang tersedia. Secara lebih lengkap, modul-modul yang tersedia pada Python dapat dilihat pada dokumentasi (help) paket Python.

Cara menggunakan modul pada standar library ini sebenarnya secara tidak langsung telah kita coba, yaitu pada Contoh 11.7. Pada contoh tersebut kita menggunakan modul math untuk mengakses konstanta pi. Secara umum cara menggunakan modul sama seperti pada contoh tersebut. Namun untuk lebih memperjelas penggunaan modul pada standard library, kita akan mencobanya pada beberapa contoh.

Contoh 11.22. Penggunaan modul untuk mencari nilai akar bilangan.

```
>>> import math
>>> A = math.sqrt(9)
>>> print A
3.0
>>>
```

Pada contoh ini kita menggunakan modul math dan menggunakan fungsi sqrt atau akar. Modul math dipanggil dengan perintah import. Perhatikan cara memanggil fungsi sqrt pada baris ke 2. Angka Sembilan merupakan argument yang kita tempatkan pada fungsi sqrt.

Modul math menyediakan banyak sekali fungsi matematika. Untuk mengetahui fungsi apa saja yang ada dalam suatu modul, kita dapat menggunakan perintah dir. Sebagai contoh, bila modul math yang ingin kita lihat fungsi-fungsinya cukup kita import modulnya kemudian ketikkan dir(math). Perhatikan contoh-contoh berikut ini.

```

>>> import math
>>> dir(math)
['_doc_', '_name_', 'acos', 'asin', 'atan', 'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log', 'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh']
>>>
>>> import datetime
>>> dir(datetime)
['MAXYEAR', 'MINYEAR', '_doc_', '_name_', 'date', 'datetime', 'datetime_CAPI', 'time', 'timedelta', 'tzinfo']
>>>
>>> import email
>>> dir(email)
['Charset', 'Encoders', 'Errors', 'FeedParser', 'Generator', 'Header', 'Iterators', 'LazyImporter', 'MIMEAudio', 'MIMEBase', 'MIMEImage', 'MIMEMessage', 'MIMEMultipart', 'MIMENonMultipart', 'MIMEText', 'Message', 'Parser', 'Utils', '_LOWERNAMES', '_MIMENAMES', '_all_', '_builtins_', '_doc_', '_file_', '_name_', '_path_', '_version_', '_name', 'base64MIME', 'email', 'importer', 'message_from_file', 'message_from_string', 'mime', 'quopriMIME', 'sys']
>>>

```

Contoh lain bagaimana menggunakan modul adalah seperti berikut:

Contoh 11.23. Penggunaan modul untuk mencetak kalender bulan tertentu.

```

>>> import calendar
>>> print calendar.month(1970, 11)
November 1970
Mo Tu We Th Fr Sa Su
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30

```

Pada contoh ini kita ingin mencetak kalender untuk bulan November tahun 1970. Modul calendar lebih dulu harus kita panggil, baru kemudian kita gunakan fungsi yang termasuk dalam modul kalender (lihat baris kedua).

Contoh 11.24. Penggunaan modul untuk merubah huruf.

```

>>> import string
>>> kata = 'python'
>>> KATA = string.upper(kata)
>>> print KATA
PYTHON
>>>
>>> kata = string.lower(KATA)
>>> print kata
python
>>>

```

Contoh di atas menunjukkan bagaimana mudahnya merubah huruf kecil ke besar dan sebaliknya. Modul yang kita pakai adalah string. Modul ini menyediakan banyak fungsi. Dua yang kita pakai adalah upper (untuk

merubah huruf kecil ke kapital) dan lower (untuk merubah huruf kapital ke huruf kecil).

11.6.2. Membuat Modul.

Seperti halnya fungsi, Python juga memberikan fasilitas untuk mendefinisikan modul sendiri. Tahapan untuk membuat modul adalah sebagai berikut:

- Buat file modul terlebih dahulu dengan menggunakan teks editor.
- Simpan file modul tersebut dengan ekstensi .py. File modul harus dengan ekstensi .py, kalau tidak maka tidak dapat digunakan.
- Gunakan modul tersebut dengan cara yang sama seperti modul-modul dalam standard library.

Untuk memudahkan pemahaman, kita akan mencoba dengan menggunakan contoh berikut:

Contoh 11.24. Membuat dan menggunakan modul.

Langkah pertama yang kita lakukan adalah membuka teks editor atau kita dapat menggunakan editor yang ada pada IDLE. Kemudian kita ketikkan isi modul tersebut. Untuk contoh ini, isi modul adalah sebagai berikut:

```
def kali (x, y):  
    hasil = x * y  
    return hasil  
  
def bagi (x, y):  
    hasil = x / y  
    return hasil  
  
def tambah (x, y):  
    hasil = x + y  
    return hasil  
  
def kurang (x, y):  
    hasil = x - y  
    return hasil
```

Langkah ke-dua adalah menyimpan file tersebut dengan menggunakan ekstensi .py. Pada contoh ini kita akan menyimpan dengan nama mymodul.py.

Langkah terakhir adalah menggunakan modul. Untuk contoh ini kita akan menggunakan modul yang telah kita buat dalam suatu program (dengan mode skrip). Kode programnya tampak di bawah ini. File kode program ini harus kita simpan di direktori yang sama dengan modul yang telah dibuat.

```

print 'Contoh memanggil modul'
print
import mymodul

X = 12.3
Y = 6.5
print 'Nilai X = ', X
print 'Nilai Y = ', Y

print 'X * Y =', mymodul.kali(X, Y)
print 'X / Y =', mymodul.bagi(X, Y)
print 'X + Y =', mymodul.tambah(X, Y)
print 'X - Y =', mymodul.kurang(X, Y)

```

Pada baris ketiga kode program, terlihat kita mengimpor modul yang telah kita buat sebelumnya. Sedangkan pada empat baris terakhir menunjukkan bagaimana kita mengakses fungsi-fungsi yang ada pada modul. Jika kita jalankan program di atas, maka hasilnya akan tampak sebagai berikut:

```

>>> ===== RESTART :
>>>
Contoh memanggil modul

Nilai X = 12.3
Nilai Y = 6.5
X * Y = 79.95
X / Y = 1.89230769231
X + Y = 18.8
X - Y = 5.8
>>>

```

Pemrograman Berbasis GUI

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Mengenali bagian-bagian dari IDE perangkat lunak pemrograman berbasis GUI dan fungsinya.
 - Menjalankan perangkat pemrograman berbasis GUI
 - Memahami prinsip-prinsip utama dalam pemrograman berbasis GUI.
 - Menerapkan algoritma dalam pemrograman berbasis GUI
-

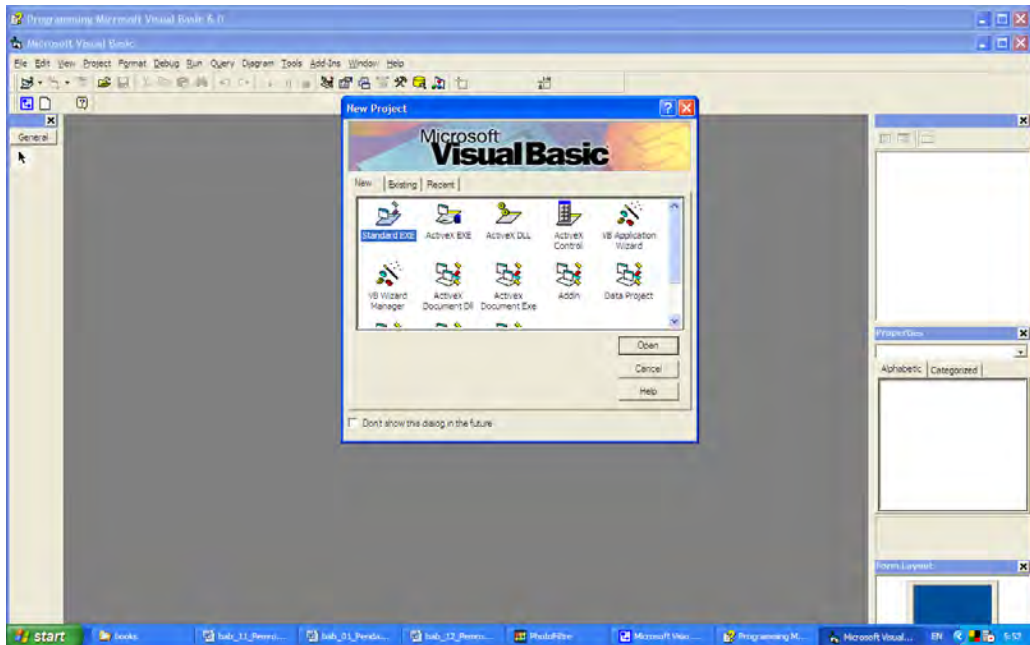
Pada bab ini kita akan membicarakan salah satu pemrograman yang populer yaitu pemrograman berbasis GUI atau kadang orang menyebutnya sebagai desktop programming atau windows programming. Perangkat lunak yang akan dipakai disini adalah Microsoft Visual Basic. Pemilihan perangkat lunak ini dikarenakan Microsoft Visual Basic menggunakan bahasa Basic yang sudah kita kenal pada Bab-Bab sebelumnya.

12.1. PENGENALAN PADA VISUAL BASIC

Visual Basic (VB) adalah salah satu bahasa pemrograman komputer. Bahasa pemrograman VB, yang dikembangkan oleh Microsoft sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu bahasa pemrograman BASIC (*Beginner's All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. VB adalah salah suatu *development tools* 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 dialek bahasa Basic yang cenderung mudah dipelajari. Visual Basic telah menjadi *tools* yang terkenal bagi para pemula maupun para developer. Namun ada kelemahan dari VB yaitu pada kinerja yang relative terasa lebih lambat dibanding dengan bahasa pemrograman lain. Namun dengan perkembangan *processor* dan *main memory* yang semakin cepat permasalahan ini menjadi tidak begitu penting.

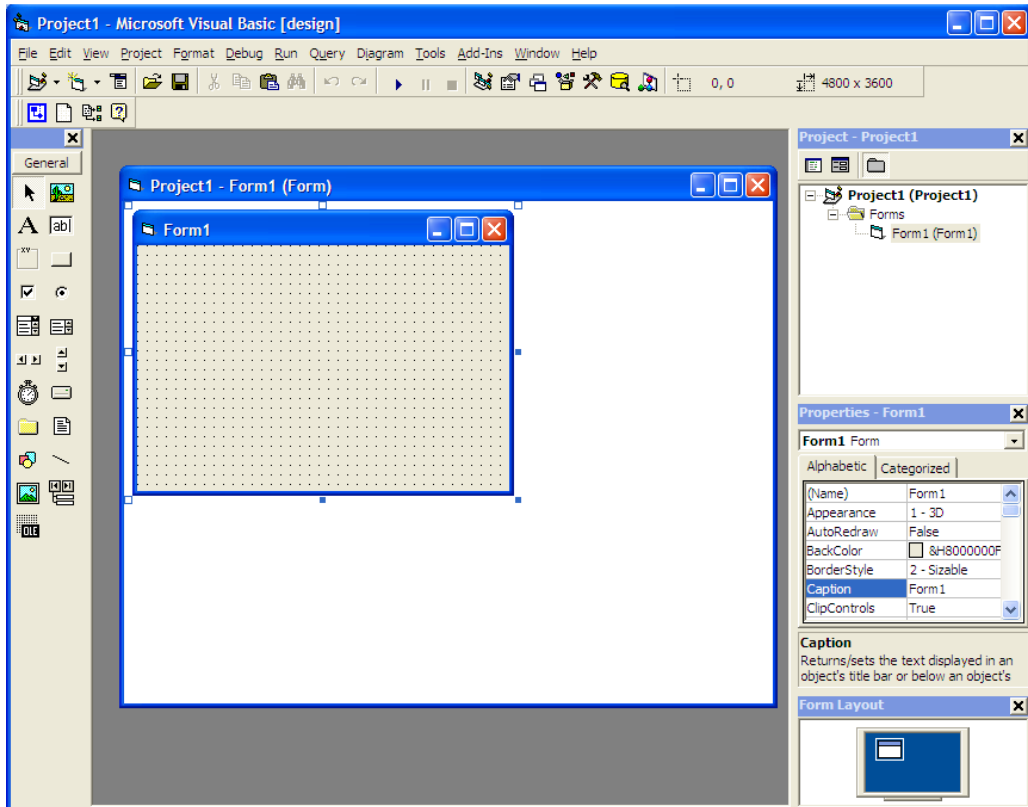
12.1.1. Memulai Visual Basic

Visual Basic adalah perangkat lunak yang berjalan di atas platform system operasi Microsoft Windows. Untuk memulai Visual Basic dapat dilakukan dengan mengklik **Start -> Programs -> Microsoft Visual Studio 6 -> Microsoft Visual Basic**. Tampilan awal Visual Basic akan tampak seperti Gambar 12.1.



Gambar 12.1. Tampilan awal Visual Basic.

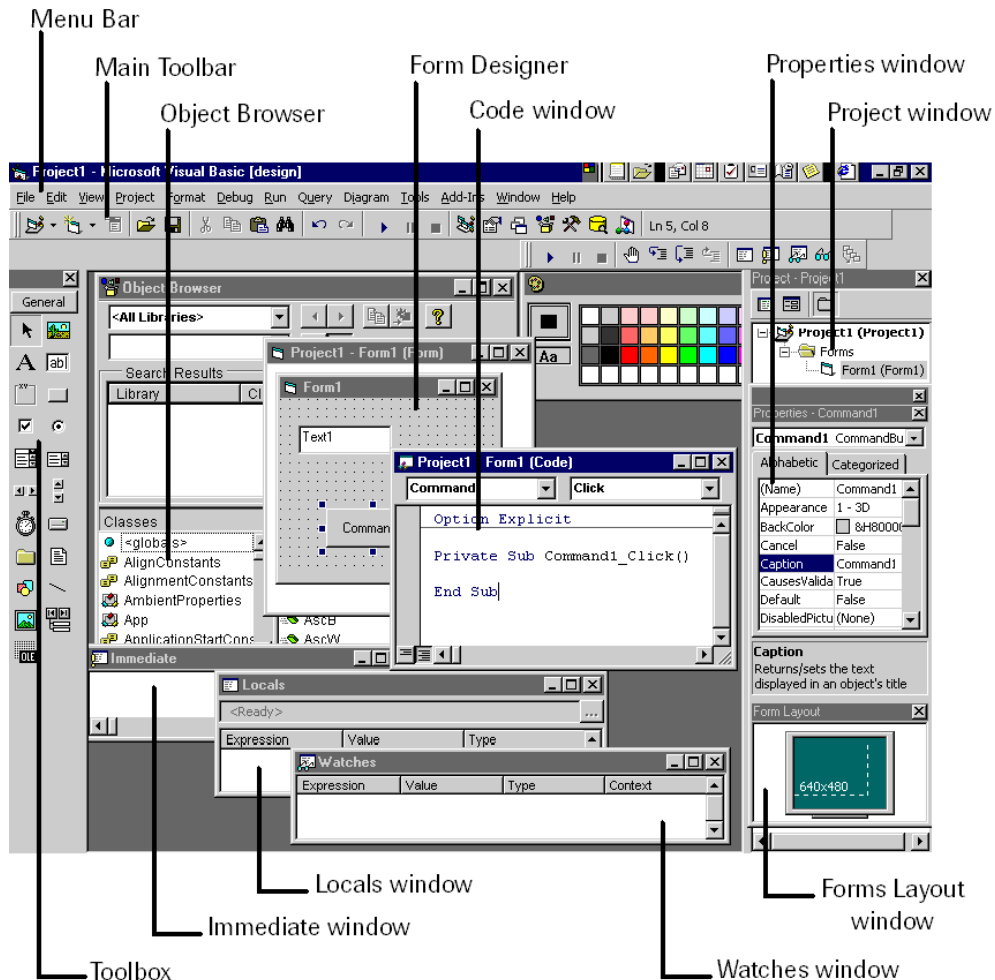
Pada Gambar 12.2 kita diminta untuk memilih jenis proyek yang akan kita buat. Untuk tahap awal proyek Standar.EXE merupakan pilihan yang biasa dilakukan. Setelah kita memilih Standar.EXE maka kita akan dibawa pada tampilan berikut.



Gambar 12.2. Tampilan awal untuk pilihan Standard.EXE.

12.1.2. IDE Visual Basic

Langkah awal dari belajar Visual Basic adalah mengenal IDE (Integrated Development Environment) Visual Basic yang merupakan Lingkungan Pengembangan Terpadu bagi programmer dalam mengembangkan aplikasinya. Dengan menggunakan IDE programmer dapat membuat user interface, melakukan coding, melakukan testing dan debugging serta menkompilasi program menjadi executable. Penguasaan yang baik akan IDE akan sangat membantu programmer dalam mengefektifkan tugas-tugasnya sehingga dapat bekerja dengan efisien. Tampilan IDE VB dapat dilihat pada Gambar 12.3.



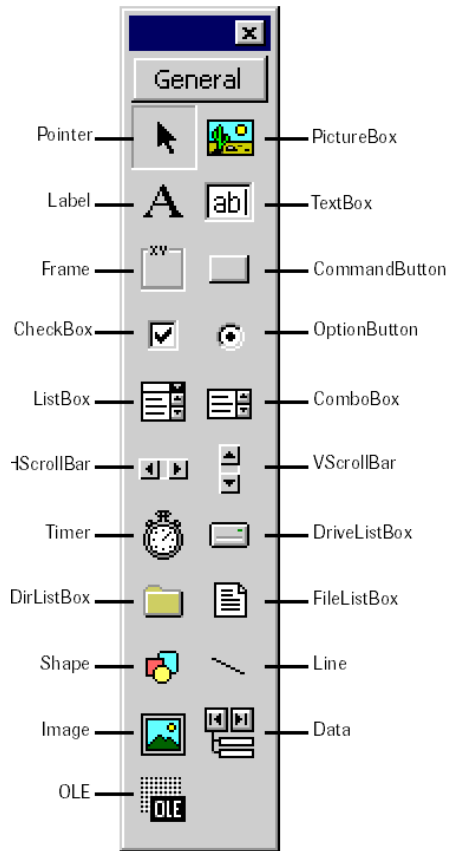
Gambar 12.3. IDE Visual Basic

- *Menu Bar*, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan project, membuka project, dll
- *Main Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.
- *Jendela Project*, jendela ini berisi gambaran dari semua modul yang terdapat dalam aplikasi anda. Anda dapat menggunakan icon Toggle Folders untuk menampilkan modul-modul dalam jendela tersebut secara di group atau berurut berdasarkan nama. Anda dapat menggunakan Ctrl+R untuk menampilkan jendela project, ataupun menggunakan icon Project Explorer.

- *Jendela Form Designer*, jendela ini merupakan tempat anda untuk merancang user interface dari aplikasi anda. Jadi jendela ini menyerupai kanvas bagi seorang pelukis.
- *Jendela Toolbox*, jendela ini berisi komponen-komponen yang dapat anda gunakan untuk mengembangkan user interface.
- *Jendela Code*, merupakan tempat bagi anda untuk menulis koding. Anda dapat menampilkan jendela ini dengan menggunakan kombinasi Shift-F7.
- *Jendela Properties*, merupakan daftar properti-properti object yang sedang terpilih. Sebagai contohnya anda dapat mengubah warna tulisan (foreground) dan warna latarbelakang (background). Anda dapat menggunakan F4 untuk menampilkan jendela properti.
- *Jendela Color Palette*, adalah fasilitas cepat untuk mengubah warna suatu object.
- *Jendela Form Layout*, akan menunjukkan bagaimana form bersangkutan ditampilkan ketika runtime.

12.1.3. Toolbox

Jendela Toolbox merupakan jendela yang sangat penting bagi anda. Dari jendela ini anda dapat mengambil komponen-komponen (object) yang akan ditanamkan pada form untuk membentuk user interface.



Gambar 12.4. Toolbox VB 6.

- *Pointer* bukan merupakan suatu kontrol; gunakan icon ini ketika anda ingin memilih kontrol yang sudah berada pada form.
- *PictureBox* adalah kontrol yang digunakan untuk menampilkan image dengan format: BMP, DIB (bitmap), ICO (icon), CUR (cursor), WMF (metafile), EMF (enhanced metafile), GIF, dan JPEG.
- *Label* adalah kontrol yang digunakan untuk menampilkan teks yang tidak dapat diperbaiki oleh pemakai.
- *TextBox* adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.
- *Frame* adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.

- *CommandButton* merupakan kontrol hampir ditemukan pada setiap form, dan digunakan untuk membangkitkan event proses tertentu ketika pemakai melakukan klik padanya.
- *CheckBox* digunakan untuk pilihan yang isinya bernilai yes/no, true/false.
- *OptionButton* sering digunakan lebih dari satu sebagai pilihan terhadap beberapa option yang hanya dapat dipilih satu.
- *ListBox* mengandung sejumlah item, dan user dapat memilih lebih dari satu (bergantung pada property *MultiSelect*).
- *ComboBox* merupakan kombinasi dari *TextBox* dan suatu *ListBox* dimana pemasukkan data dapat dilakukan dengan pengetikkan maupun pemilihan.
- *HScrollBar* dan *VScrollBar* digunakan untuk membentuk scrollbar berdiri sendiri.

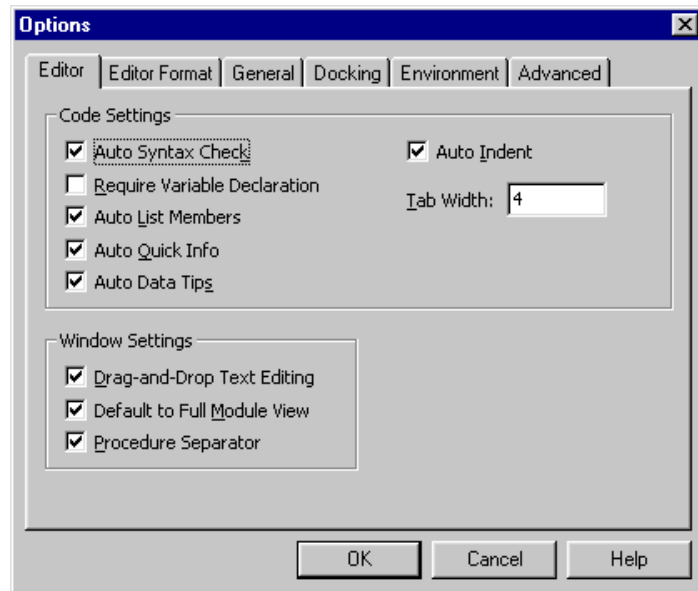
- *Timer* digunakan untuk proses background yang diaktifkan berdasarkan interval waktu tertentu. Merupakan kontrol non-visual.
- *DriveListBox*, *DirListBox*, dan *FileListBox* sering digunakan untuk membentuk dialog box yang berkaitan dengan file.
- *Shape* dan *Line* digunakan untuk menampilkan bentuk seperti garis, persegi, bulatan, oval.
- *Image* berfungsi menyerupai image box, tetapi tidak dapat digunakan sebagai kontainer bagi kontrol lainnya. Sesuatu yang perlu diketahui bahwa kontrol image menggunakan resource yang lebih kecil dibandingkan dengan PictureBox
- *Data* digunakan untuk *data binding*
- *OLE* dapat digunakan sebagai tempat bagi program eksternal seperti Microsoft Excel, Word, dll.

12.1.4. Mengatur Lingkungan Kerja Visual Basic

Pengaturan IDE Visual Basic dapat dilakukan dengan menu Tools, Option, pengaturan dapat dilakukan sesuai dengan selera dan kebiasaan programmer sehingga dapat bekerja dengan baik dan efektif. Mulai Visual Basic 5.0, IDE Visual Basic memperkenalkan MDI Development Environment, dan beberapa hal dapat diatur dengan menggunakan menu Tools -> Option adalah sebagai berikut :

- **Mengatur Editor**

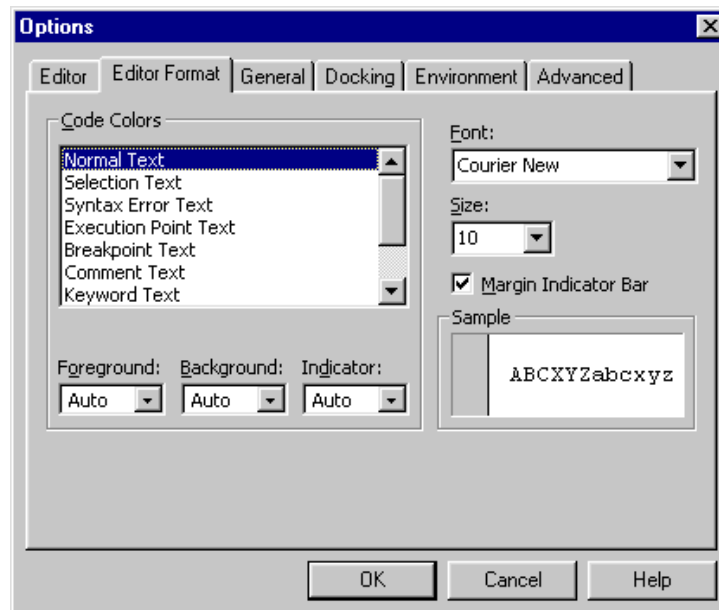
Bagian ini digunakan untuk mengatur tampilan jendela Code (Editor). Pada opsi ini kita dapat mengatur tentang indentasi kode, bantuan untuk auto syntax, auto list dan lain-lain (Gambar 12.4)



Gambar 12.5. Jendela pengaturan editor.

- **Mengatur Format Editor**

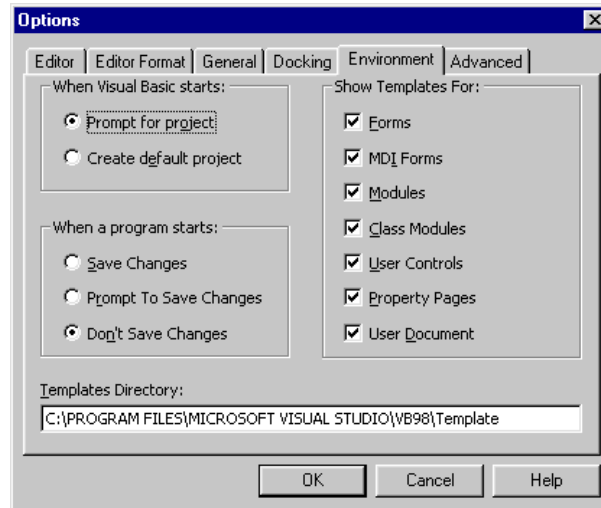
Bagian ini mengatur bagaimana format editor termasuk jenis font yang dipakai dan ukurannya, perbedaan warna font untuk kode-kode tertentu dan lain-lain.



Gambar 12.6. Jendela pengaturan format editor.

- **Mengatur Environment**

Bagian ini berhubungan dengan pengaturan IDE secara keseluruhan, seperti bagaimana ketika awal Visual Basic dipanggil, template, dan lain-lain.



Gambar 12.7. Jendela pengaturan environment.

12.2. PRINSIP POKOK PEMROGRAMAN BERBASIS GUI

Secara prinsip ada dua bagian pokok dalam pengembangan aplikasi dengan menggunakan VB, yaitu: visual design dan event-driven programming.

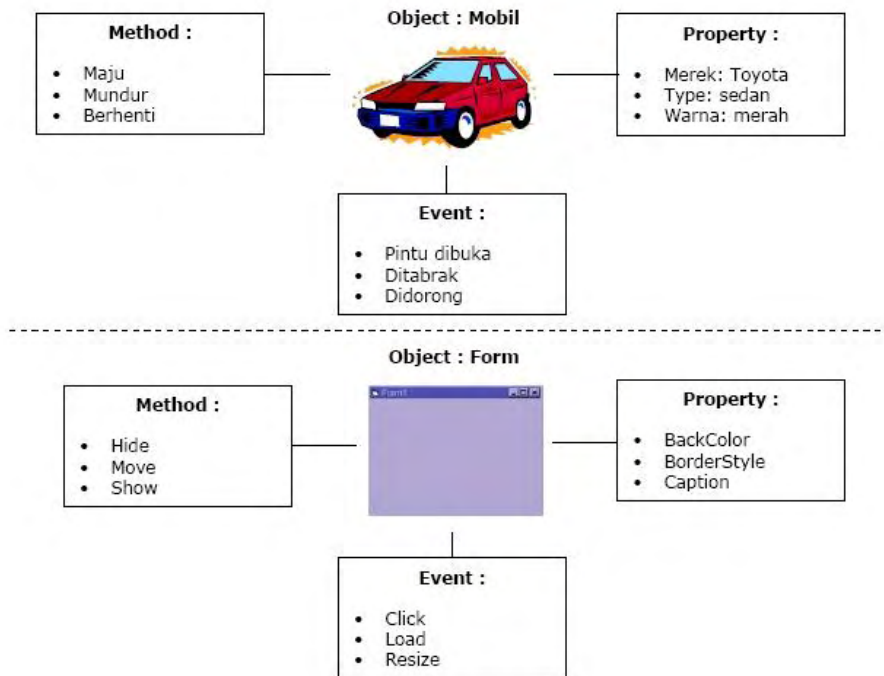
12.2.1. Visual Design

Dalam lingkungan Windows, *User-interface* sangat memegang peranan penting, karena dalam pemakaian aplikasi yang kita 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 selalu dimulai dengan pembentukan *user interface*. Untuk mendisain user interface, pengetahuan yang paling dibutuhkan hanyalah pemahaman dasar tentang jenis dan kegunaan dari control dan dasar-dasar bagaimana menggambar sebuah object.

Form dan control merupakan elemen-elemen dasar dalam user interface pada aplikasi-aplikasi berbasis Windows. Dalam VB, elemen-

elemen ini disebut obyek karena dapat dimanipulasi seperti sebuah obyek. Obyek merupakan suatu kombinasi dari kode dan data yang dapat diperlakukan sebagai satu kesatuan. Sebuah obyek memiliki sejumlah property dan sejumlah method, dan akan bereaksi terhadap sejumlah event eksternal seperti halnya obyek fisik.

Sebagai ilustrasi (gambar 12.7), sebuah mobil adalah sebuah obyek fisik yang memiliki property, method dan event. Salah satu property adalah warna. Biasanya property warna dari mobil ditetapkan ketika sebuah mobil dibuat. Tetapi kalau kita tidak suka dengan warna mobil yang kita beli, kita masih dapat merubahnya, misalnya dengan mengecat ulang. Demikian juga dalam VB, property dari sebuah control biasanya ditentukan ketika object dibuat (pada saat ditempatkan pada sebuah form), tetapi kita dapat merubah property ini dengan memberikan nilai baru. Kita dapat merubah nilai property pada saat disain (dengan menggunakan jendela Properties) atau pada saat *runtime* (melalui kode program). Beberapa property hanya tersedia pada saat disain, dan beberapa property hanya tersedia pada saat runtime.



Gambar 12.8. Obyek, Property, Method dan Event

12.2.2. Event-Driven Programming

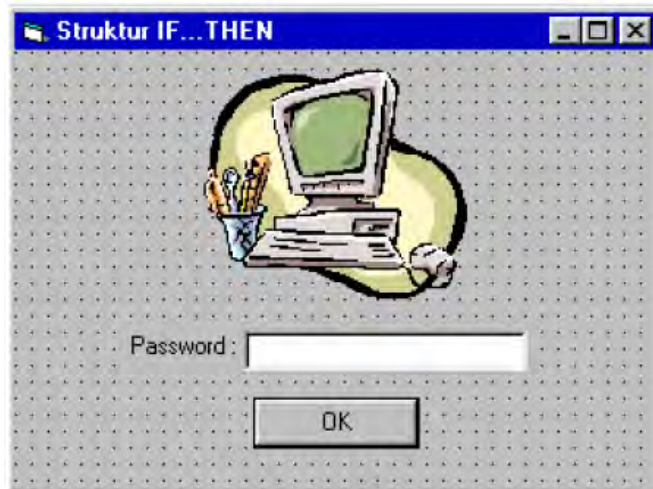
Pemrograman suatu aplikasi bukanlah sesuatu yang mudah, namun ada sebuah metodologi yang tidak boleh kita tinggalkan. Aplikasi yang dibuat dengan VB bukanlah sebuah program yang monolithic (hanya ada satu urutan jalannya program aplikasi). Ketika kita membuat program dengan VB, pertama kita harus menentukan bagaimana aplikasi akan berinteraksi dengan pengguna. Atau dengan kata lain, kita harus menentukan bagaimana setiap control bereaksi terhadap aksi yang dilakukan pengguna (misalnya: click mouse, double-click mouse, penekanan salah satu tombol pada keyboard, dan lain-lain). Konsep ini biasa disebut sebagai Event-Driven Programming, karena bukan aplikasi yang menentukan alur namun kejadian (event) yang disebabkan oleh pengguna yang menentukan alur dari aplikasi. Program aplikasi yang kita buat bereaksi terhadap kondisi eksternal (event), dan aksi dari pengguna yang menentukan bagaimana alur dari aplikasi.

12.3. STRUKTUR KENDALI

Seperti yang telah tercantum pada bab-bab sebelumnya tentang struktur kendali, pada Visual Basic hal inipun menjadi bagian penting dalam pemrograman. Beberapa aplikasi berikut ini menunjukkan bagaimana prinsip struktur kendali diterapkan.

Contoh 12.1. Program dengan Struktur IF...THEN

Aktifkan VB 6 kemudian buatlah form sebagai berikut :



Atur property untuk masing-masing obyek sebagai berikut :

Object	Properties	Value
Form5	Caption StartPosition	Struktur IF...THEN 2 – CenterScreen
Image1	Stretch Picture Visible	True Komputer.wmf False
Label1	Caption	Password :
Text1	PasswordChar Text	* <kosong>
Command1	Caption Default	OK True

Buka Jendela Code dan pada bagian Code Editor ketikkan kode programnya sebagai berikut :

```
Private Sub Command1_Click()
    If Text1.Text = "nusantara" Then Image1.Visible = True
End Sub
```

Klik menu **Project > Project1 Properties** lalu klik tab General. Gantilah Startup Object-nya menjadi Form5. Coba jalankan Project1 :

- Ketikkan sembarang teks pada TextBox lalu klik tombol OK atau tekan Enter, tidak terjadi apaapa.
- Ketikkan “nusantara” pada TextBox lalu klik tombol OK atau tekan Enter, gambar computer akan muncul.

Penjelasan kode program :

If Text1.Text = "nusantara" Then Image1.Visible = True

kondisi
kode program yang dikerjakan
bila kondisi TRUE

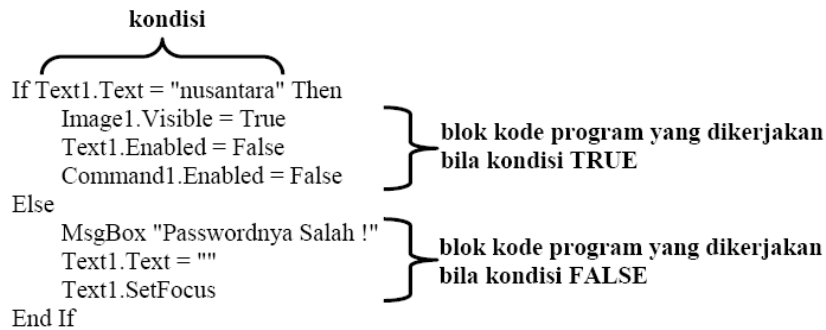
Modifikasi kode programnya menjadi sebagai berikut :

```
Private Sub Command1_Click()
    If Text1.Text = "nusantara" Then
        Image1.Visible = True
        Text1.Enabled = False
        Command1.Enabled = False
    Else
        MsgBox "Passwordnya Salah !"
        Text1.Text = ""
        Text1.SetFocus
    End If
End Sub
```

Coba jalankan Project1 :

- Ketikkan sembarang teks pada TextBox lalu klik tombol OK atau tekan Enter, muncul kotak pesan “Passwordnya Salah !”. Klik tombol OK pada kotak pesan tersebut.
- Ketikkan “nusantara” pada TextBox lalu klik tombol OK atau tekan Enter, gambar computer akan muncul. TextBox dan tombol OK menjadi *disable* (tidak bisa digunakan).

Penjelasan kode program :



Catatan tambahan :

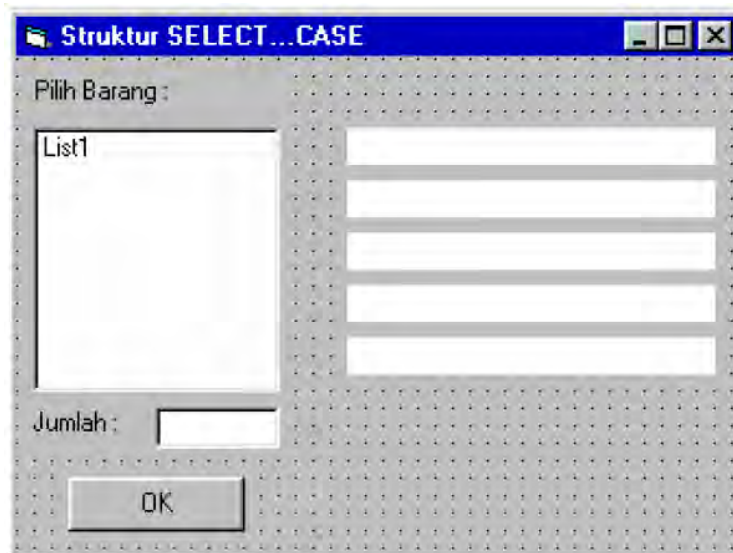
- Teks “nusantara” harus diketik huruf kecil semua. Ingat : data string bersifat *case sensitive* !
- Agar password-nya tidak bersifat *case sensitive*, modifikasi pernyataan kondisinya menjadi :

```
If LCase(Text1.Text) = "nusantara" Then
```

Fungsi LCase adalah untuk mengkonversi semua string yang diinput ke Text1.Text menjadi huruf kecil, walaupun user menginputnya dengan huruf kapital.

Contoh 12.2. Program dengan menggunakan Struktur SELECT...CASE

Aktifkan VB 6 melalui tombol Start, kemudian buat form sebagai berikut :



Atur property untuk masing-masing obyek sebagai berikut :

Object	Properties	Value
Form6	Caption StartPosition	Struktur SELECT...CASE 2 – CenterScreen
Label1	Caption	Pilih Barang :
List1	-	-
Label2	Caption	Jumlah :
Text1	Text	<kosong>
Command1	Caption	OK
Label3-7	Name BackColor Caption	lblBarang, lblHarga, lblJumlah, lblDiskon, lblTotal Palette : <putih> <kosong>

Buka Jendela Code dan pada bagian Code Editor ketikkan kode programnya sebagai berikut :

```

Private Sub Form_Load()
    List1.AddItem "Disket"
    List1.AddItem "Buku"
    List1.AddItem "Kertas"
    List1.AddItem "Pulpen"
End Sub

Private Sub Command1_Click()
    Dim harga As Currency, total As Currency
    Dim jumlah As Integer
    Dim diskon As Single
    Dim satuan As String
    If List1.Text = "" Then
        MsgBox "Anda belum memilih barang !!"
        List1.ListIndex = 0
    
```

```

        Exit Sub
    End If
    If Text1.Text = "" Then
        MsgBox "Anda belum mengisi jumlah barang !!"
        Text1.SetFocus
        Exit Sub
    End If
    Select Case List1.Text
        Case "Disket"
            harga = 35000
            satuan = "Box"
        Case "Buku"
            harga = 20000
            satuan = "Lusin"
        Case "Kertas"
            harga = 25000
            satuan = "Rim"
        Case "Pulpen"
            harga = 10000
            satuan = "Pak"
    End Select

    lblBarang.Caption = "Barang : " & List1.Text
    lblHarga.Caption = "Harga : " & Format(harga,
    "Currency") & "/" & satuan
    lblJumlah.Caption = "Jumlah : " & Text1.Text & " " &
    satuan
    jumlah = Text1.Text
    Select Case jumlah
        Case Is < 10
            diskon = 0
        Case 10 To 20
            diskon = 0.15
        Case Else
            diskon = 0.2
    End Select
    total = jumlah * (harga * (1 - diskon))
    lblDiskon.Caption = "Diskon : " & Format(diskon, "0 %")
    lblTotal.Caption = "Total Bayar : " & Format(total,
    "Currency")
End Sub

```

Klik menu Project > Project1 Properties lalu klik tab General. Gantilah Startup Object-nya menjadi Form6.

Coba jalankan Project1 :

- List1 akan terisi nama-nama barang.

- Klik tombol OK, akan muncul kotak pesan "Anda belum memilih barang !!".
- Klik OK, nama barang pertama akan terpilih secara otomatis. Anda boleh memilih nama barang yang lainnya.
- Klik tombol OK, akan muncul kotak pesan "Anda belum mengisi jumlah barang !!".
- Klik OK, text1 akan menjadi focus. Isi jumlah barangnya, misalnya : 10.
- Klik tombol OKm akan tampil : nama barang, harga barang (persatuannya), jumlah barang (dengan satuannya), diskon dan total bayar.
- Coba ganti pilihan nama barang (pada List1) dan jumlah barang (pada Text1) lalu klik lagi tombol OK.

Penjelasan kode program :

```

Select Case List1.Text      cek barang yang dipilih :
Case "Disket"
    harga = 35000
    satuan = "Box" }      bila barang = Disket
Case "Buku"
    harga = 20000
    satuan = "Lusin" }    bila barang = Buku
Case "Kertas"
    harga = 25000
    satuan = "Rim" }      bila barang = Kertas
Case "Pulpen"
    harga = 10000
    satuan = "Pak" }      bila barang = Pulpen
End Select

lblBarang.Caption = "Barang : " & List1.Text
lblHarga.Caption = "Harga : " & Format(harga, "Currency") & "/" & satuan
lblJumlah.Caption = "Jumlah : " & Text1.Text & " " & satuan }      tampilkan has

jumlah = Text1.Text

Select Case jumlah      cek jumlah barang :
Case Is < 10
    diskon = 0          bila jumlah < 10 → diskon = 0%
Case 10 To 20
    diskon = 0.15       bila jumlah 10-20 → diskon = 15%
Case Else
    diskon = 0.2        bila jumlah > 20 → diskon = 20%
End Select

total = jumlah * (harga * (1 - diskon))

lblDiskon.Caption = "Diskon : " & Format(diskon, "0 %")
lblTotal.Caption = "Total Bayar : " & Format(total, "Currency") }      Hitung total bayar dan
                                                                    tampilkan hasilnya

```

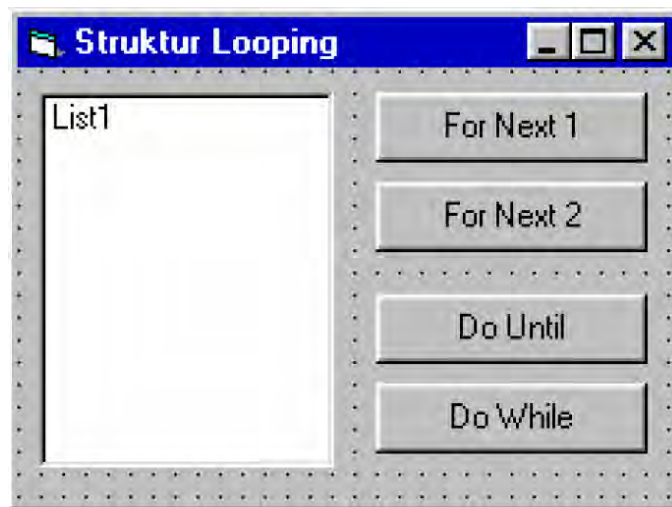

Catatan :

- Bila jumlah barang diisi dengan selain angka akan muncul pesan error.
- Untuk mengecek isi Text1 angka atau bukan, tambahkan kode berikut :

```
If Not IsNumeric(Text1.Text) Then
    MsgBox "Isi jumlah barang harus angka !!"
    Text1.SetFocus
Exit Sub
End If
```

Contoh 12.3. Penggunaan pengulangan dengan For ... Next.

Buka VB dan buatlah form sebagai berikut :



Atur property seperti tabel berikut :

Object	Properties	Value
Form7	Caption StartPosition	Struktur Looping 2 – CenterScreen
List1	-	-
Command1-4	Caption	For Next 1 For Next 2 Do Until Do While

Buka Jendela Code dan pada bagian Code Editor ketikkan kode programnya sebagai berikut :

```

Dim i As Integer

Private Sub Command1_Click()
List1.Clear
For i = 1 To 100
    List1.AddItem "Angka " & i
Next i
End Sub

Private Sub Command2_Click()
List1.Clear
For i = 100 To 1 Step -2
    List1.AddItem "Angka " & i
Next i
End Sub

Private Sub Command3_Click()
List1.Clear
i = Asc("A")
Do Until i > Asc("Z")
    List1.AddItem "Huruf " & Chr(i)
    i = i + 1
Loop
End Sub

Private Sub Command4_Click()
List1.Clear
i = Asc("Z")
Do While i >= Asc("A")
    List1.AddItem "Huruf " & Chr(i)
    i = i - 1
Loop
End Sub

```

12.4. Mengenal Procedure dan Fungsi

Procedure digunakan untuk memadatkan tugas-tugas berulang ataupun proses yang digunakan bersama, seperti perhitungan yang sering dilakukan, text dan manipulasi kontrol, serta operasi database.

Secara umum ada dua keuntungan dengan pemakaian procedure dalam program :

- Procedure memungkinkan anda untuk memecahkan program anda pada unit logika yang lebih kecil, sehingga anda dapat dengan mudah melakukan proses debug dari pada jika keseluruhan program tanpa menggunakan procedure.

- Procedure yang digunakan dalam suatu program dapat bertindak sebagai suatu blok yang dibangun untuk program lain dengan sedikit perubahan atau tanpa perubahan sama sekali.

Ada beberapa jenis procedure yang digunakan dalam Visual Basic :

- Sub procedure yang tidak mengembalikan nilai
- Function procedure yang mengembalikan nilai
- Property procedure yang dapat mengembalikan nilai dan diisi nilai yang mengacu pada suatu objek.

12.4.1. Sub Procedure

Syntax penulisan Sub procedure:

```
[Private|Public] [Static] Sub namaprosedur (argumen-argumen)  
    pernyataan-pernyataan  
End Sub
```

Setiap kali procedure dipanggil, maka pernyataan-pernyataan yang berada di antara Sub dan End Sub akan dijalankan. Argumen pada procedure adalah nilai yang akan dilewatkan saat pemanggilan procedure.

Di Visual Basic Sub Procedure dapat dibagi atas dua yaitu :

- **General Procedure**, procedure yang diaktifkan oleh aplikasi
- **Event Procedure**, Procedure yang diaktifkan oleh system sebagai respon terhadap event.

Contoh, sub Tengah yang dapat digunakan untuk menampilkan form ketengah Layar, dimana x adalah parameter yang merupakan form yang akan dibuat ketengah layar.

```
Sub Tengah(x As Form)  
    x.Top = (Screen.Height - x.Height) \ 2  
    x.Left = (Screen.Width - x.Width) \ 2  
End Sub
```

```
Private Sub Form_Load()  
    Call Tengah(Me)  
End Sub
```

12.4.2. Function Procedure

Pada Visual Basic telah tersedia berbagai fungsi bawaan seperti Sqr, Cos, dan Chr, tetapi fungsi-fungsi yang tersedia tersebut bersifat umum dan kadang-kadang tidak memenuhi kebutuhan programmer, untuk keperluan

tersebut anda dapat menciptakan fungsi-fungsi sendiri yang dikenal dengan Function procedure.

Adapun syntax penulisan function procedure :

```
[Private|Public] [Static] Function    namaprocedure    (argumen-  
argumen) [As type]  
    statements  
End Function
```

Ada tiga perbedaan antara function dan procedure :

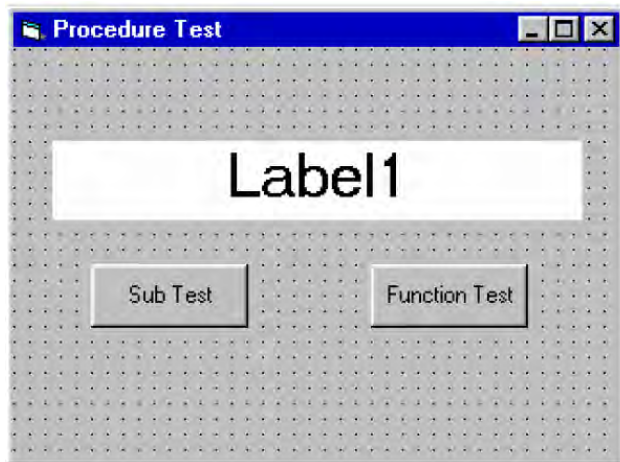
- Umumnya anda dapat memanggil suatu function dengan mengikutkan nama function sisi kanan dari statement atau ekspresi. (returnvalue = function()).
- Function memiliki type data seperti suatu variabel. Ini menentukan type yang dari nilai yang dikembalikan.
- Nilai kembali dimasukkan ke namafunction itu sendirinya, dan suatu function dapat menjadi bagian dari suatu ekspresi yang panjang.

Contoh Fungsi ciptaan untuk mengembalikan nama bulan dari suatu tanggal dalam bahasa Indonesia.

```
Function Bulan(x As Date)  
Dim sRet As String  
  
Select Case Month(x)  
    Case 1: sRet = "Januari"  
    Case 2: sRet = "Februari"  
    Case 3: sRet = "Maret"  
    Case 4: sRet = "April"  
    Case 5: sRet = "Mei"  
    Case 6: sRet = "Juni"  
    Case 7: sRet = "Juli"  
    Case 8: sRet = "Agustus"  
    Case 9: sRet = "September"  
    Case 10: sRet = "Oktober"  
    Case 11: sRet = "Nopember"  
    Case 12: sRet = "Desember"  
    Case Else  
        sRet = "tidak sah"  
    End Select  
  
Bulan = sRet  
  
End Function
```

Contoh 12.4. Penggunaan Prosedur

Aktifkan VB melalui tombol Start, kemudian form sebagai berikut.



Atur Property-nya sebagai berikut :

Object	Properties	Value
Form1	Caption StartPosition	Procedure Test 2 – Center Screen
Label1	Alignment BackColor Font ToolTipText	2 – Center Palette : <putih> Size : 24 Dobel-Klik di Sini
Command1	Caption ToolTipText	Sub Test Klik di Sini
Command2	Caption	Function Test

Buka Jendela Code, lalu pada bagian Code Editor ketikkan kode programnya sebagai berikut :

```

Private Sub TulisTeks(teks As String, warna As
ColorConstants)
    With Label1
        .Caption = teks
        .ForeColor = warna
    End With
End Sub

Private Function JumlahAngka() As String
    Dim angka1 As String, angka2 As String
    Dim hasil As Single
    angka1 = InputBox("Tulis angka 1 :", "Jumlah Angka")
    angka2 = InputBox("Tulis angka 2 :", "Jumlah Angka")
    If angka1 <> "" And angka2 <> "" Then
        hasil = CSng(angka1) + CSng(angka2)
        JumlahAngka = Cstr(hasil)
    End If
End Function

```

```

Private Sub Label1_DblClick()
    Call TulisTeks("Hai", vbBlue)
End Sub

Private Sub Command1_Click()
    Call TulisTeks("Hallo", vbRed)
End Sub

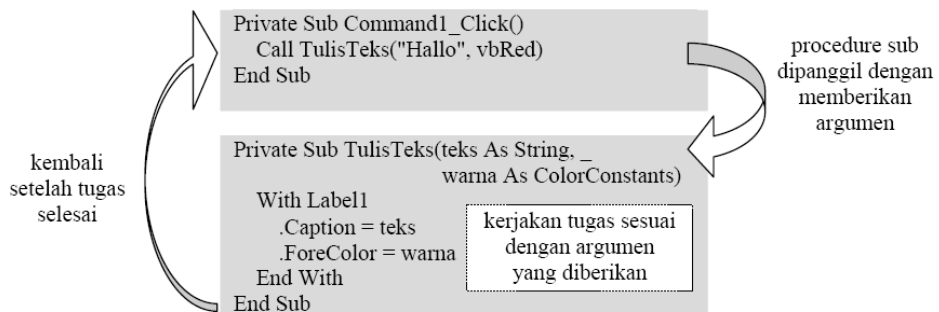
Private Sub Command2_Click()
    Label1.Caption = "Jumlah = " & JumlahAngka()
End Sub

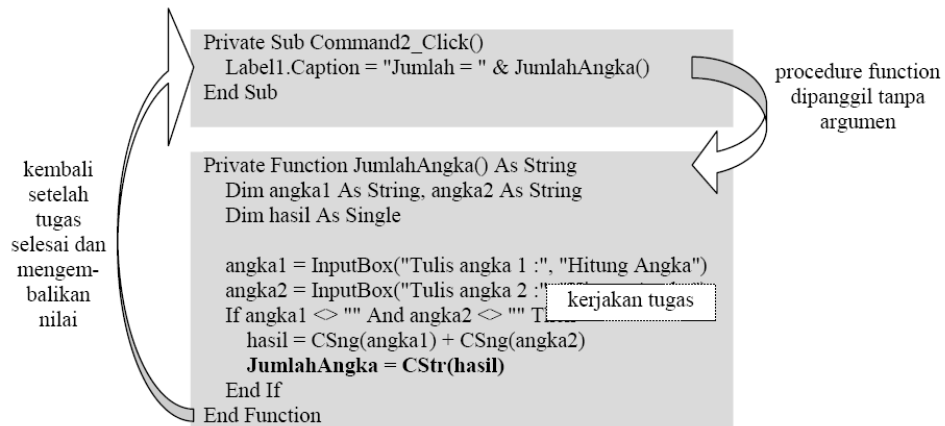
```

Klik menu Project > Project1 Properties lalu klik tab General. Gantilah Startup Object-nya menjadi Form11. Coba jalankan Project-nya :

- . Dobel-klik Label1, akan muncul tulisan “Hai” dengan warna biru.
- . Klik Command1, akan muncul tulisan “Hallo” dengan warna merah.
- . Klik Command2, akan muncul kotak input.
- . Ketikkan angka 5, lalu klik tombol OK.
- . Ketikkan angka 10, lalu klik tombol OK.
- . Akan muncul tulisan “Jumlah = 15”.

Proses ketika procedure di-‘panggil’:





Dapat kita lihat bahwa prosedur Tulisteks dipanggil, dan dimasukkan 2 variabel ke dalamnya yaitu teks dan warna. Selanjutnya kedua argumen tersebut diolah oleh prosedur Tulisteks. Akhirnya ditampilkan tulisan dan warnanya yang merupakan hasil dari properti Caption dan Forecolor dari obyek label.

12.4.3. Fungsi built-in

Seperti halnya pada Python, pada VB juga telah disediakan banyak fungsi built-in yang dapat digunakan untuk berbagai tujuan. Pada bab ini tidak akan dibahas detail fungsi-fungsi built-in tersebut. Namun pada lampiran, dapat dilihat beberapa fungsi built-in yang sering digunakan.

Pencarian Kesalahan Dan Pengujian

TUJUAN

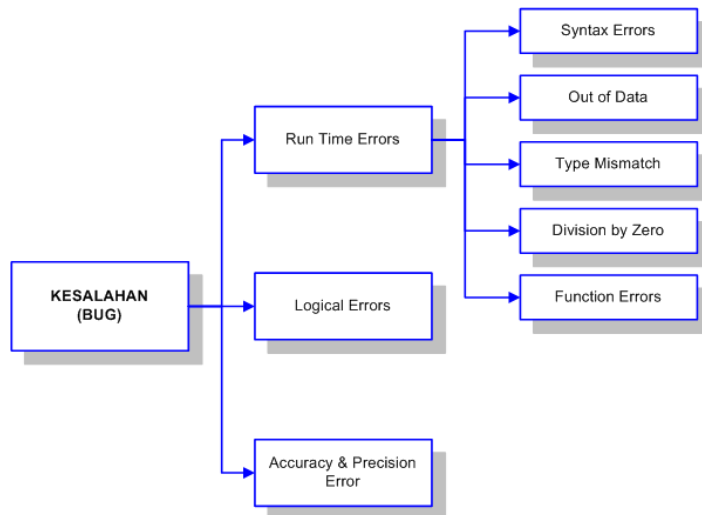
Setelah anda mempelajari ini, diharapkan anda akan mampu :

- Memahami pengertian kesalahan.
 - Mengidentifikasi jenis-jenis kesalahan dalam pemrograman.
 - Memperbaiki kesalahan berdasarkan jenis kesalahannya
 - Mengetahui tipe-tipe pengujian perangkat lunak.
 - Mengetahui dokument apa yang harus ada pada perangkat lunak
-

Pembuatan program computer apalagi jika program yang dibuat sangat panjang dan logika prosesnya sangat rumit, seringkali kita akan menjumpai kesalahan. Kesalahan atau bug adalah *ketidaktepatan, kekurangan atau ketidakcocokan dari program yang dibuat yang menyebabkan program tidak dapat bekerja seperti yang diinginkan*. Bug ini harus dicari dengan teliti dan harus dibenarkan karena jika tidak dibenarkan program komputer tidak akan berjalan sesuai yang diinginkan. Proses pencarian kesalahan pada program komputer disebut sebagai *debugging*. Pada tahapan pengkodean program sebenarnya aktifitas terbesar adalah pencari kesalahan bahkan ada yang menyebutkan 10% menuliskan kode program dan 90% pencarian kesalahan.

13.1. JENIS-JENIS KESALAHAN

Bug dapat muncul dimana saja, namun dalam pemrograman komputer dikenal klasifikasi bug seperti terdapat pada Gambar 13.1.



Gambar 13.1. Jenis-jenis kesalahan.

13.1.1. Run Time Errors

Run Time Errors adalah kesalahan yang terdeteksi ketika program dijalankan. Kesalahan ini merupakan tipe kesalahan yang paling mudah diperbaiki. Beberapa perangkat lunak pemrograman terutama yang berbasis GUI memberikan petunjuk dimana letak kesalahan dan memberikan saran perbaikan.

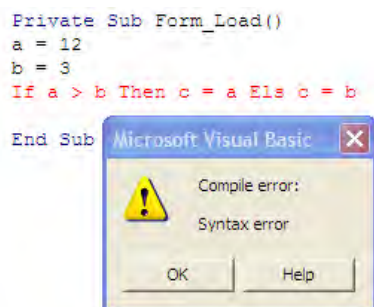
a) Syntax Errors

Kesalahan ini terjadi karena beberapa hal seperti:

- Kesalahan dalam penulisan pernyataan (reserve word) dari bahasa pemrograman.

Contoh 13.1.

Perhatikan Gambar 13.2 berikut ini:



Gambar 13.2. Kesalahan penulisan reserve word.

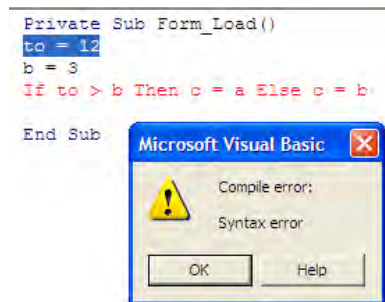
Pada Gambar 13.2 tampak adanya kesalahan karena penulisan pernyataan Els. Seharusnya adalah Else. Kesalahan tipe ini mungkin kesalahan yang paling sering kita lakukan ketika mengetikkan kode program.

Perangkat pengembang pemrograman seperti Microsoft Visual Basic, Borland Delphi, atau yang lainnya, akan memberikan peringatan kesalahan dan tempat dimana ada kesalahan. Dengan cara seperti ini programmer akan dengan mudah memperbaiki kesalahannya.

- Penggunaan pernyataan (reserve word) sebagai nama variable

Contoh 13.2.

Perhatikan Gambar 13.3 berikut ini:



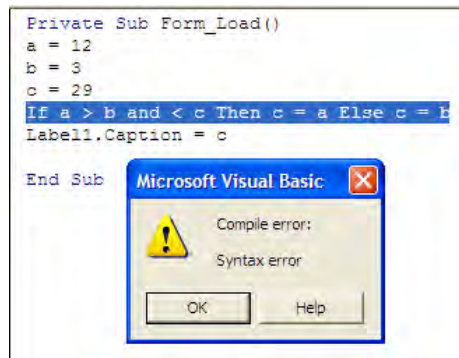
Gambar 13.3. Kesalahan penggunaan nama variable.

Pada Gambar 13.3 di atas terlihat bahwa ada penggunaan nama variable yang tidak dibenarkan, yaitu penggunaan 'to' yang diisi nilai 12. 'to' sebenarnya adalah reserve word yang dimiliki oleh bahasa pemrograman Microsoft Visual Basic sehingga tidak boleh digunakan sebagai nama variable.

- Aturan penulisan pernyataan (reserve word) yang tidak benar.

Contoh 13.3.

Perhatikan Gambar 13.4 berikut ini:



Gambar 13.4. Cara penulisan pernyataan yang salah.

Pada gambar di atas terjadi kesalahan dalam pada penulisan persamaan yaitu pada:

If a > b and < c Then c = a Else c = b

Seharusnya ditulis sebagai berikut:

If a > b and b < c Then c = a Else c = b

b) Out of Data

Kesalahan jenis ini terjadi jika kita mencoba membaca lebih dari banyaknya data yang disediakan pada file data, record pada table basis data, atau pernyataan DATA.

Contoh 13.4.

Perhatikan Gambar 13.5 berikut ini:



Gambar 13.5. Kesalahan out of data.

Pada Gambar 13.5. terlihat sebuah kode program untuk membaca (READ) sejumlah data dan disimpan dalam variable A. Data yang tersedia pada akhir baris kode hanya 3 yaitu 2,3 dan 6, sedangkan

proses pembacaan dilakukan sebanyak 5 kali (lihat bagian `FOR i = 1 TO 5`). hal ini yang disebut sebagai out of data. Bagaimana bila jumlah data yang tersedia lebih banyak dari jumlah ulangan pembacaan? Perhatikan Gambar 13.6 berikut ini.



Gambar 13.6. Pembacaan data jika jumlah data lebih banyak

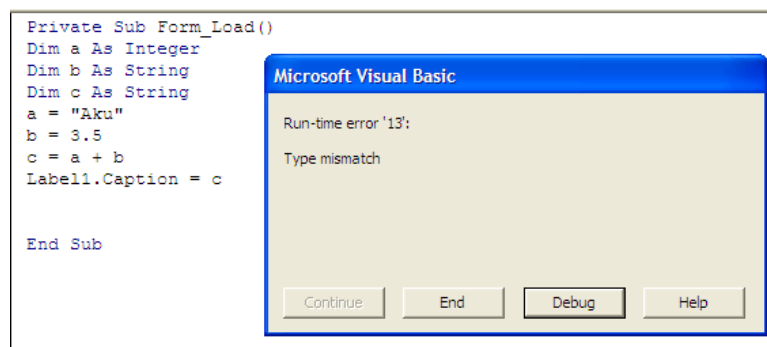
Jika jumlah data lebih banyak daripada ulangan pembacaan maka data yang dibaca hanya sejumlah ulangan pembacaan. Pada Gambar 13.6. jumlah data ada 6 yaitu 2,3,6,8,10,13 tapi yang akan ditampilkan hanya 5 data 2,3,6,8,10 karena ulangan pembacaan hanya 5 kali.

c) Type Mismatch

Kesalahan ini terjadi jika kita menempatkan tipe data yang tidak tepat. Misalnya kita ingin suatu variable berisi nilai-nilai integer, tetapi kita salah menempatkan menjadi tipe data string.

Contoh 13.5.

Perhatikan Gambar 13.7 berikut ini:



Gambar 13.7. Kesalahan penggunaan tipe data

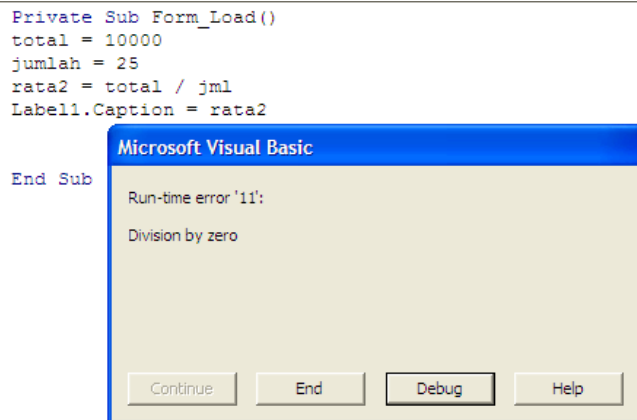
Pada Gambar 13.7 terlihat ada ketidakcocokan penggunaan tipe data dan nilai yang dimasukkan. Variable a didefinisikan dengan tipe data Integer tetapi isi dari variable a adalah string (lihat baris a = "Aku". Hal inilah yang disebut sebagai *type mismatch*.

d) Division by Zero

Kesalahan ini terjadi jika dalam suatu perhitungan terdapat pembagian dengan nilai penyebut sama dengan 0.

Contoh 13.6.

Perhatikan Gambar 13.8 berikut ini:



Gambar 13.8. kesalahan karena division by zero.

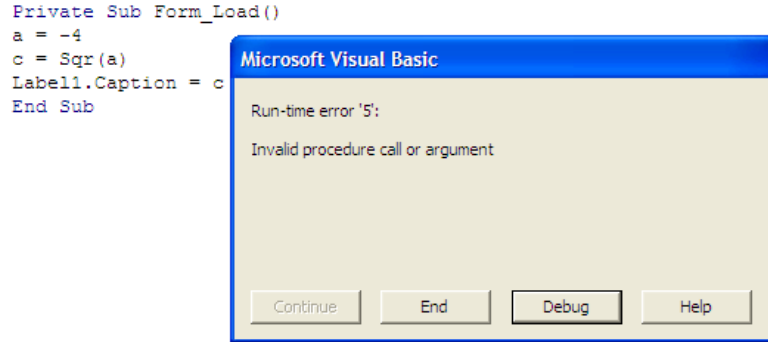
Pada Gambar 13.8 terjadi kesalahan karena variable total yang sudah mempunyai nilai yaitu 10000 dibagi dengan variable jml yang belum didefinisikan nilainya. Dalam banyak bahasa pemrograman apabila kita tidak memberikan nilai awal untuk sebuah variable, maka dianggap variable itu bernilai 0.

e) Function Errors

Function errors adalah kesalahan ketika kita menggunakan argument suatu fungsi di luar batas/ketentuan yang diperbolehkan.

Contoh 13.7.

Perhatikan Gambar 13.9 berikut ini:



Gambar 13.9. Kesalahan penggunaan argument.

Gambar 13.9 menunjukkan adanya kesalahan menggunakan argument dari fungsi Sqr. Fungsi Sqr atau akar hanya boleh diisi dengan bilangan real positif, tidak boleh negative. Sedangkan pada contoh di atas nilai argument yang digunakan yaitu a bernilai -4 (negative).

13.1.2. Logical Errors

Logical errors atau kesalahan logika, terjadi karena kesalahan dalam pembuatan formula atau kesalahan algoritma program. Kesalahan ini sulit untuk dideteksi, karena ketika kesalahan terjadi, program tetap dapat berjalan (tidak terganggu). Kesalahan ini dapat diketahui dengan melihat dan menganalisa keluaran yang salah atau tidak sesuai dengan yang diperkirakan.

Contoh 13.8.

Misalkan kita ingin menukar nilai dari variable a dan b. Variable a bernilai awal 3 dan variable b bernilai awal 5. kita ingin memperoleh hasil akhir variable a menjadi bernilai 5 dan variable b bernilai 3. Perhatikan potongan kode program berikut:

```
a = 3
b = 5
a = b
b = a
Debug.Print a, b
```

Jika program di atas kita eksekusi maka tidak akan ada peringatan kesalahan apapun. Namun dari tampilan hasil kita akan dapatkan keluaran dari program ini tidak benar. Baik variable a maupun b akan sama-sama bernilai 5. Mengapa?

Pada baris ketiga nilai variable a akan berganti menjadi 5 (sama dengan b). Sehingga ketika digunakan pada baris berikutnya maka variable b akan diisi nilai variable a yang sudah menjadi 5. Output program akan menjadi a = 5 dan b = 5. Apa yang kita harapkan, yaitu terjadinya pertukaran nilai a dengan b tidak terjadi. Hal ini karena ada kesalahan logika. Kita dapat memperbaiki dengan cara berikut:

```
a = 3
b = 5
c = 0
c = a    'variabel c akan bernilai sama dengan a, yaitu 3
a = b    'variabel a akan bernilai sama dengan b, yaitu 5
b = c    'variabel b akan bernilai sama dengan c, yaitu 3
```

Pada kode yang telah diperbaiki ini, kita menggunakan 1 variabel bantu yaitu c sebagai tempat penyimpanan sementara. Keluaran dari kode program di atas akan tepat seperti yang kita inginkan yaitu a = 5 dan b = 3

13.1.3. Precision Errors

Kesalahan ini berhubungan dengan ketelitian hasil. Kesalahan dapat terjadi karena keterbatasan ketelitian pada variable-variabel yang digunakan. Penggunaan tipe data yang tidak tepat seringkali berpengaruh pada ketelitian hasil.

Contoh 13.9.

Perhatikan potongan kode program berikut ini:

```
Dim a As Single
Dim b As Single
Dim c As Single
a = 25000205
b = 2 / 1000 * a
c = a - b
Debug.Print a, b, c
```

Jika program ini dijalankan maka hasilnya adalah:

```
2,50002E+07      50000,41      2,49502E+07
```

Bandingkan jika kita rubah tipe data masing-masing variable menjadi double, seperti potongan program berikut ini:

```
Dim a As Double
Dim b As Double
Dim c As Double
a = 25000205
b = 2 / 1000 * a
c = a - b
```

```
Debug.Print a, b, c
```

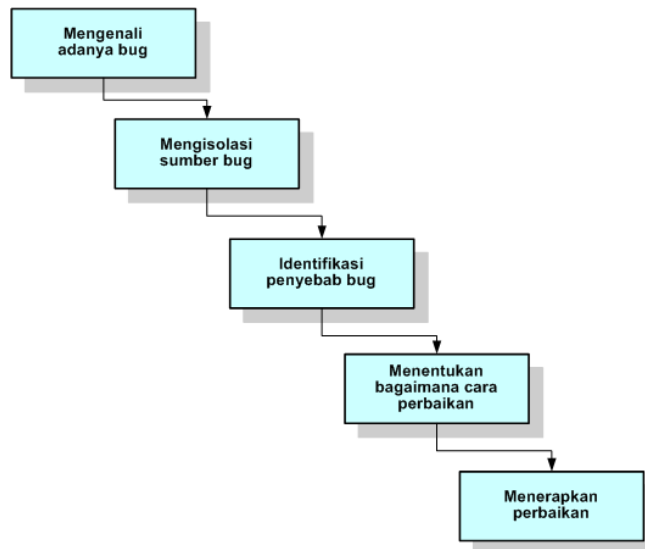
Output yang diberikan akan menjadi seperti berikut:

```
25000205      50000,41      24950204,59
```

Penggunaan tipe data double membuat keluaran menjadi lebih teliti. Kesalahan dalam ketelitian perhitungan ini biasanya sangat diperhatikan ketika kita berhubungan dengan masalah-masalah ilmu pengetahuan alam, seperti fisika, biologi, kimia dan lain-lain.

13.2. TEKNIK PENCARIAN KESALAHAN

Meskipun proses pencarian kesalahan (debugging) sebagian besar ditentukan oleh pengalaman dalam pemrograman (jam terbang) namun ada prinsip-prinsip utama yang dapat diterapkan dalam prose pencarian kesalahan. Gambar 13.10 menunjukkan tahapan-tahapan dalam pencarian kesalahan.



Gambar 13.10. tahapan-tahapan dalam pencarian kesalahan

- Mengenalinya adanya bug

Tahapan ini bertujuan untuk mengenali gejala terjadinya bug. Mempelajari gejala dengan baik kemudian menentukan problem yang memang sebenarnya terjadi akan sangat membantu mengenali bug dengan benar.

- Mengisolasi sumber bug

Tahapan ini mungkin tahapan yang paling sulit, karena kita harus memperkirakan di bagian mana dari kode program yang kita buat yang

menyebabkan adanya bug. Pada kode program yang relative pendek, lebih mudah menandai bagian mana yang menyebabkan bug. Tapi pada kode program yang sangat panjang mungkin membutuhkan waktu yang cukup lama dan berulang-ulang dalam penelusurannya. Setelah kita memperkirakan pada beberapa tempat di kode program, maka kita dapat memisahkan dan kemudian mencermati bagian-bagian yang terpisah ini lebih detil.

- Identifikasi penyebab bug

Setelah dapat menentukan lokasi dimana terjadi bug, maka kita dapat menentukan apa sebenarnya penyebab dari bug. Seperti telah dijelaskan di bagian sebelumnya, pada kesalahan-kesalahan yang bertipe run time errors, peringatan terjadinya kesalahan dapat dilakukan oleh perangkat pengembang pemrograman. Selain itu lokasi dimana terjadi bug juga ditunjukkan dan penyebab terjadinya bug juga disampaikan. Hal ini akan sangat membantu programmer untuk bekerja lebih cepat dan tepat. Sedangkan kesalahan logika dan kesalahan ketelitian penyebab bug harus dilakukan dengan hati-hati dan melihat keluaran dari program.

- Menentukan bagaimana cara perbaikan

Jika penyebab kesalahan sudah diidentifikasi, maka perbaikan harus segera dilakukan. Tahapan ini memerlukan kecermatan karena jangan sampai perbaikan justru menyebabkan terjadinya bug yang baru. Pada tipe kesalahan run time errors, perbaikan dapat dengan mudah dilakukan karena ada petunjuk atau manualnya.

- Menerapkan perbaikan

Apabila cara perbaikan dirasa sesuai, maka perbaikan dapat diterapkan dan diuji. Pengujian pada perbaikan mengandung dua hal penting, yaitu apakah perbaikan telah mampu menghilangkan bug dan apakah perbaikan tidak membuat terjadinya bug yang baru.

13.3. PENGUJIAN

Pengujian sistem diperlukan terhadap situs yang dibuat dengan cara *outsourcing* maupun *in-house site building*. Pengujian sistem melibatkan semua kelompok pengguna yang telah direncanakan pada tahap penyusunan strategi bisnis. Pengujian tingkat penerimaan terhadap sistem berakhir ketika semua entitas bisnis internal secara nyata menggunakan system informasi yang telah di-*install*. Uji penerimaan terhadap sistem diperlukan untuk membuktikan bahwa strategi bisnis telah bekerja.

Ada beberapa tipe pengujian yang bias dilakukan, yaitu :

- ◆ Inspection

Tipe ini melibatkan partisipan untuk memeriksa kode program untuk memprediksi kesalahan-kesalahan yang mungkin muncul, terutama dari sisi bahasa.

◆ Walkthrough

Pada tipe ini sekelompok group ahli menilai setiap produk yang dihasilkan selama proses pengembangan system.

◆ Desk Checking

Tipe ini kode-kode program dieksekusi secara manual dan berurutan oleh tim penilai.

◆ Unit Testing

Tipe ini menguji setiap modul secara tersendiri untuk mencari kesalahan pada kode-kode program.

◆ Integration Testing

Tipe ini merupakan uji lanjutan dari unit testing. Seluruh modul diuji secara bersama-sama sebagai suatu kesatuan system untuk menguji bagaimana modul terintegrasi. Modul biasanya terintegrasi secara top-down.

◆ System Testing

Tipe ini merupakan pengujian pada seluruh system untuk melihat unjuk kerja system apakah sudah memenuhi apa yang telah ditargetkan.

13.4. DOKUMENTASI

Dokumentasi dalam pengembangan system informasi merupakan bagian yang sangat penting. Dokumentasi yang baik dilakukan sepanjang siklus pengembangan system. Ada dua kelompok dokumen yang harus tersedia dalam pengembangan system informasi, yaitu dokumen pengguna dan dokumen sistem. Dokumen pengguna biasanya berisi tentang informasi umum tentang aplikasi, bagaimana system bekerja dan bagaimana menggunakan system tersebut. Sedangkan dokumen system biasanya meliputi :

- Dokumen yang berisi informasi detil tentang spesifikasi disain, fungsi-fungsi internal dan eksternal.
- Dokumen internal, yaitu dokumen yang berisi kode-kode program yang merupakan hak milik stakeholder system.
- Dokumen eksternal termasuk diagram-diagram terstruktural dari system seperti DFD dan ER Diagram.

Dasar-Dasar Sistem Basis Data

TUJUAN

Setelah anda membaca Bab ini, diharapkan anda akan mampu :

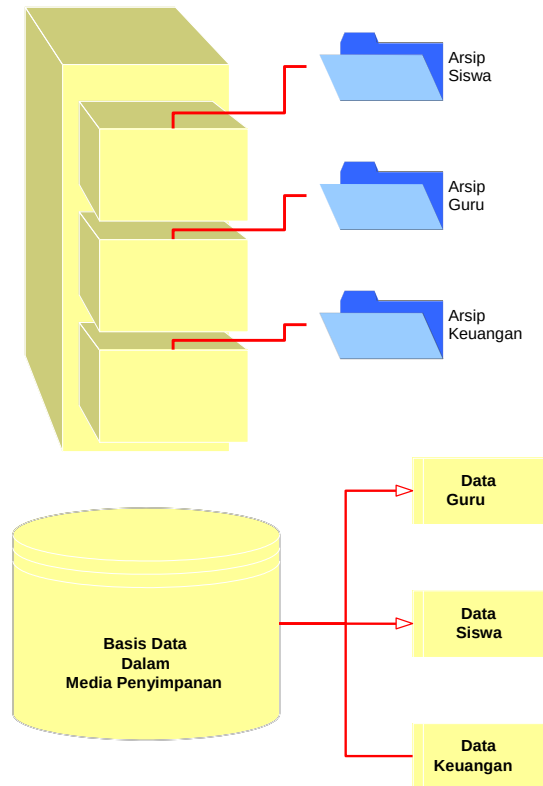
- Menjelaskan pengertian basis data.
 - Menjelaskan operasi-operasi dasar basis data.
 - Memahami system basis data dan komponen-komponennya.
 - Mengetahui tahapan-tahapan umum pengembangan basis data.
 - Menyebutkan beberapa jenis perangkat lunak DBMS
-

14.1. Pengertian Basis Data

Basis data (database) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Basis data merupakan salah satu komponen utama dalam sistem informasi, karena merupakan basis dalam penyediaan informasi bagi para pemakai (Fathansyah, 1999; Post, 1999).

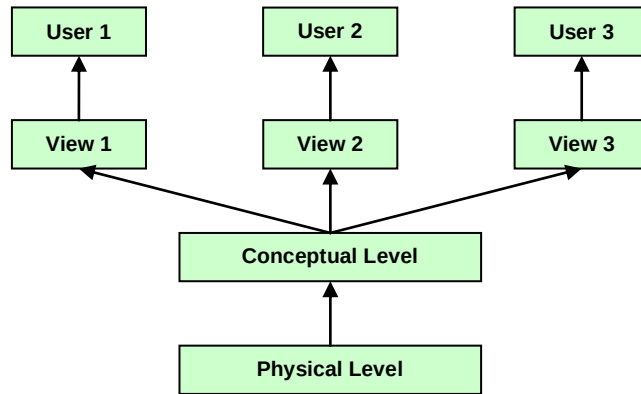
Jika dibayangkan, basis data mirip dengan lemari di ruang administrasi sekolah yang menyimpan berbagai arsip. Masing-masing jenis arsip dikelompokkan, diatur dan disimpan pada tempat yang telah ditentukan. Sehingga akan ada kelompok arsip siswa, arsip guru, arsip mata pelajaran, arsip keuangan, dan lain-lain. Perbedaannya hanya pada media penyimpanannya. Kalau lemari arsip menggunakan lemari dari kayu, besi atau plastik, sedangkan basis data menggunakan media penyimpan elektronik seperti disk (hard disc, CD, atau tape). Gambar 14.1 memberikan ilustrasi tentang kesamaan lemari arsip dan basis data.

Satu hal penting yang harus diperhatikan, basis data bukan hanya sekedar penyimpanan data secara elektronik. Tidak semua penyimpanan data elektronik bisa disebut basis data. Apabila penyimpanan itu tidak menggunakan prinsip pengaturan, pemisahan atau pengorganisasian maka kita tidak dapat menyebut penyimpanan data tersebut sebagai basis data. Pada Gambar 14.1 terlihat penerapan prinsip pengaturan, pengorganisasian atau pemisahan, baik pada lemari arsip atau pada basis data.



Gambar 14.1. Lemari arsip dan basis data.

Prinsip utama dalam basis data adalah konsep independensi data yaitu pemisahan data dari program aplikasinya (Lewis et al., 2002; Post, 1999). Sedangkan tujuan utama dalam basis data adalah membantu pengguna dalam abstraksi suatu system. Ada tiga level abstraksi yang biasanya digunakan yaitu *physical level*, *conceptual level* dan *view level* (Gambar 14.2). *Physical level* menunjukkan bagaimana data akan disimpan. *Conceptual level* berkaitan dengan data apa yang akan disimpan dan bagaimana hubungan antar data tersebut. *View level* merupakan level tertinggi yang menjelaskan bagian-bagian basis data pada pengguna tertentu (Ramakrishnan and Gehrke, 2000).



Gambar 14.2. Tingkatan dalam abstaksi data (Lewis et al., 2002).

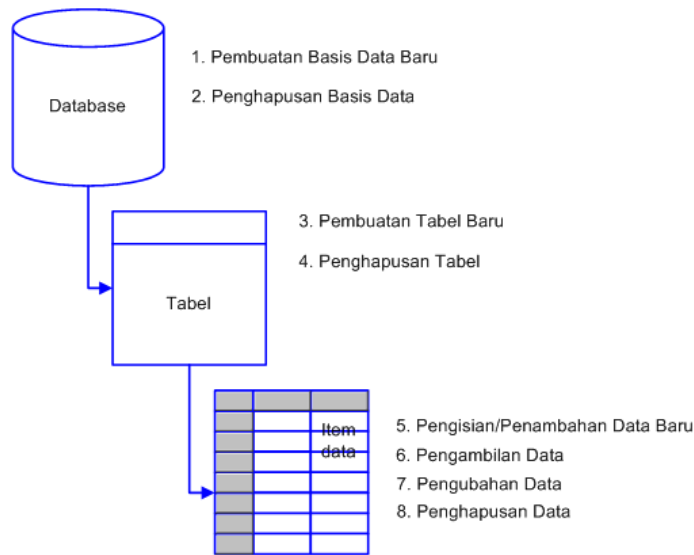
Basis data mempunyai beberapa kriteria penting, yaitu :

1. Bersifat data oriented dan bukan program oriented.
2. Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3. Dapat dikembangkan dengan mudah, baik volume maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah
5. Dapat digunakan dengan cara-cara yang berbeda.

14.1.1. Operasi dasar basis data

Secara bertingkat, operasi dasar basis data dapat digambarkan dalam skema pada Gambar 14.3. Operasi-operasi tersebut meliputi:

- Pembuatan basis data baru (create database). Operasi ini sama dengan pembuatan atau pembelian lemari arsip yang baru.
- Penghapusan basis data (drop database). Operasi ini sama dengan pengrusakan atau penghancuran lemari arsip.
- Pembuatan tabel baru (create table). Operasi ini sama dengan penambahan kelompok arsip baru. Operasi ini baru bisa dijalankan jika basis data telah dibuat.
- Penghapusan tabel (drop table). Operasi ini sama dengan pengrusakan kelompok arsip lama. Operasi ini baru bias dijalankan jika tabel telah ada pada suatu basis data.



Gambar 14. 3. Operasi-operasi dasar pada basis data.

- Pengisian atau penambahan data baru (insert data) pada suatu tabel. Operasi ini mirip dengan penambahan lembaran arsip baru pada kelompok arsip. Operasi ini baru bias dijalankan jika tabel telah dibuat.
- Pengambilan data dari suatu tabel (retrieve data). Operasi ini mirip dengan pencarian lembaran arsip yang tersimpan dalam kelompok arsip.
- Pengubahan data dari suatu tabel (update data). Operasi ini mirip dengan perbaikan isi lembaran arsip dari suatu kelompok arsip
- Penghapusan data dari suatu tabel (delete). Operasi ini mirip dengan penghapusan sebuah lembaran arsip dari suatu kelompok arsip.

14.1.2. Tujuan basis data

Basis data dibangun untuk memenuhi tujuan dalam pengorganisasian data, yang antara lain sebagai berikut :

1. Efisiensi meliputi kecepatan (speed), ruang simpan (space) dan keakuratan (accuracy).
2. Menangani data dalam jumlah besar.
3. Kebersamaan pemakaian (Shareability).
4. Meniadakan duplikasi dan inkonsistensi data.

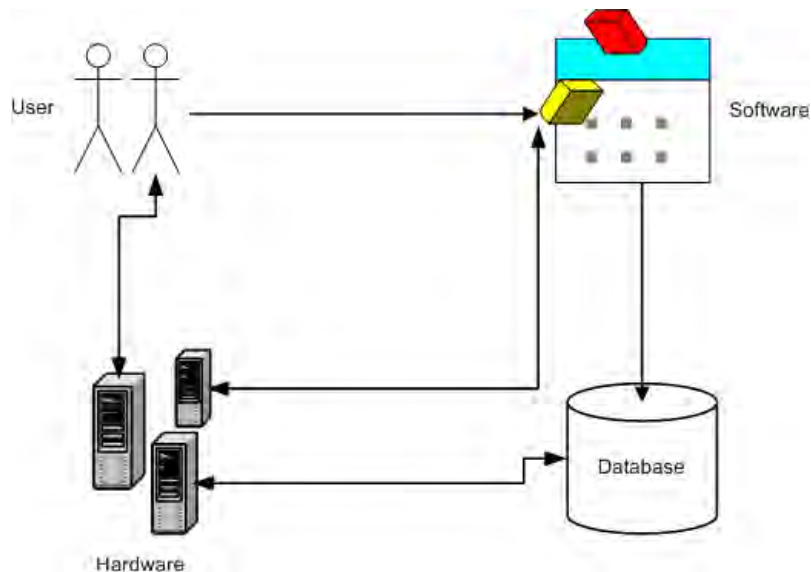
14.2. Sistem Basis Data

Seperti telah dijelaskan pada bab terdahulu, sistem merupakan kumpulan elemen yang saling berhubungan dan secara bersama-sama mencapai tujuan tertentu. Basis data hanyalah sebuah obyek yang pasif atau mati. Basis data ada karena ada yang membuat dan basis data akan berguna jika ada yang menggerakkan atau mengelolanya. Penggerak secara langsung adalah suatu program atau perangkat lunak dan yang menjalankan perangkat lunak adalah pengguna (manusia). Perangkat lunak hanya dapat berjalan jika ada perangkat keras yang bekerja. Oleh karena itu pengertian suatu sistem basis data adalah sebagai berikut :

Kumpulan elemen-elemen seperti basis data, perangkat lunak, perangkat keras, dan manusia yang saling berinteraksi untuk mencapai tujuan yaitu pengorganisasian data.

14.2.1. Komponen sistem basis data

Berdasarkan definisi definisi sistem basis data, ada 4 komponen utama sistem basis data yaitu, perangkat keras (hardware), perangkat lunak (software), basis data (database) dan pengguna (user). Keterkaitan antar komponen tersebut dapat dilihat pada gambar 14.4. Dari gambar tersebut tampak bahwa sebenarnya pengguna (user) tidak bisa secara langsung berinteraksi dengan basis data, tetapi harus dengan bantuan perangkat lunak.



Gambar 14. 4. Komponen-komponen sistem basis data dan keterkaitannya.

- Perangkat keras

Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah:

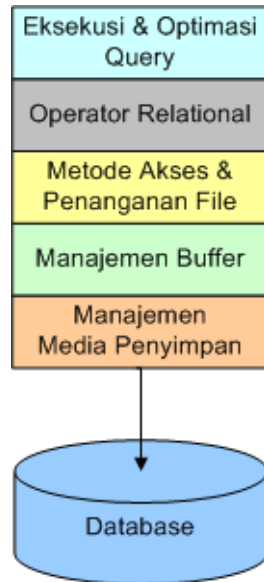
- a) Komputer (satu untuk sistem yang *stand-alone* atau lebih dari satu untuk sistem jaringan)
- b) Memori sekunder yang *on-line* (*Hardisk*)
- c) Memori sekunder yang *off-line* (*Tape* atau *Removable Disk*) untuk backup data.
- d) Media/perangkat komunikasi (untuk sistem jaringan).

- Perangkat lunak

Ada tiga kelompok perangkat lunak yang terkait dengan sistem basis data, yaitu Sistem Operasi, Database Management System (DBMS) dan Aplikasi-aplikasi tambahan.

Sistem operasi untuk menjalankan sistem basis data akan lebih baik jika memang secara khusus digunakan hanya untuk sistem basis data. Hal ini karena seiring dengan bertambahnya jumlah data maka kerja sistem menjadi berat, sehingga bila dicampur dengan penggunaan lain mungkin akan menurunkan kinerjanya. Program pengelola basis data hanya dapat aktif (*running*) jika Sistem Operasi yang dikehendaki sesuai. Jenis-jenis sistem operasi dapat dilihat pada Bab 5.

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah Perangkat Lunak yang khusus/spesifik. Perangkat lunak inilah disebut DBMS yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Ia juga menerapkan mekanisme pengaman data, pemakaian data secara bersama, pemaksaan keakuratan/konsistensi data, sebagainya. Secara ringkas struktur suatu DBMS dapat dilihat pada Gambar 14.5. Penjelasan singkat tentang beberapa jenis DBMS dapat dilihat pada bagian akhir dari bab ini.



Gambar 14.5. Struktur umum DBMS.

Aplikasi-aplikasi tambahan bersifat opsional (bisa ada dan bisa tidak) dan biasanya terdapat pada DBMS sebagai fungsi tambahan. Sebagai contoh, aplikasi pembuat report (laporan), aplikasi untuk mendisain form, aplikasi untuk membuat diagram atau chart, aplikasi untuk monitoring sistem, dan aplikasi-aplikasi lainnya.

- Basis data

Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data dapat berisi/memiliki sejumlah objek basis data (seperti file/tabel, indeks, dan lain lain). Disamping berisi/menyimpan data, setiap basis data juga mengandung/menyimpan definisi struktur(baik untuk basis data maupun objek objeknya secara detail).

- Pengguna

Jenis-jenis pengguna dalam sistem basis data dapat dilihat pada Gambar 14.6.

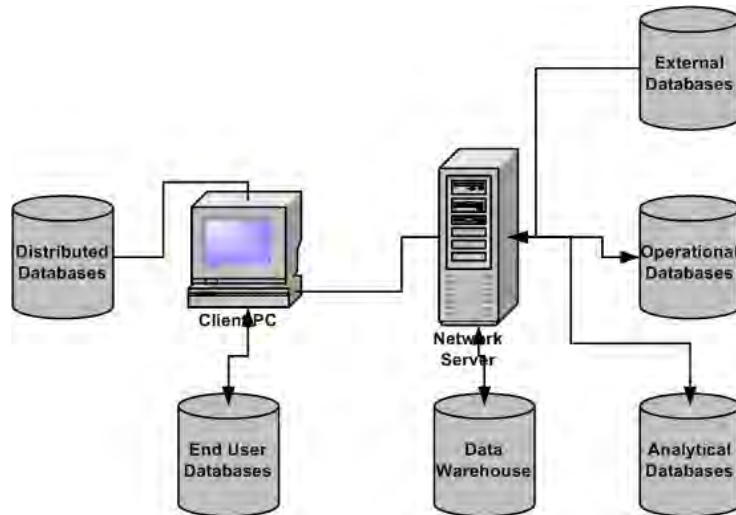


Gambar 14.6. Klasifikasi pengguna sistem basis data

14.3. Tipe Basis Data

Dalam perkembangan teknologi informasi dan penerapannya dalam aplikasi bisnis, basis data terbagi menjadi beberapa tipe. O'Brien (1999) menyebutkan ada beberapa tipe utama dari basis data, yaitu *Operational databases*, *Analytical Databases*, *Data Warehouse*, *Distributed Databases*, dan *End User Database* (Gambar 14.7).

- *Operational databases*. Tipe basis data ini menyimpan detail data yang dibutuhkan untuk mendukung operasi-operasi dalam keseluruhan organisasi. Tipe ini biasa disebut sebagai *Subject Area Databases* (SADB), *Transaction Databases*, dan *Production Databases*.
- *Analytical Databases*. Basis data tipe ini menyimpan data dan informasi yang merupakan hasil ekstraksi dari basis data tipe *Operational* dan *External*. Data dan informasi yang tersimpan merupakan ringkasan-ringkasan yang paling dibutuhkan oleh pihak manajer dan pengguna akhir.

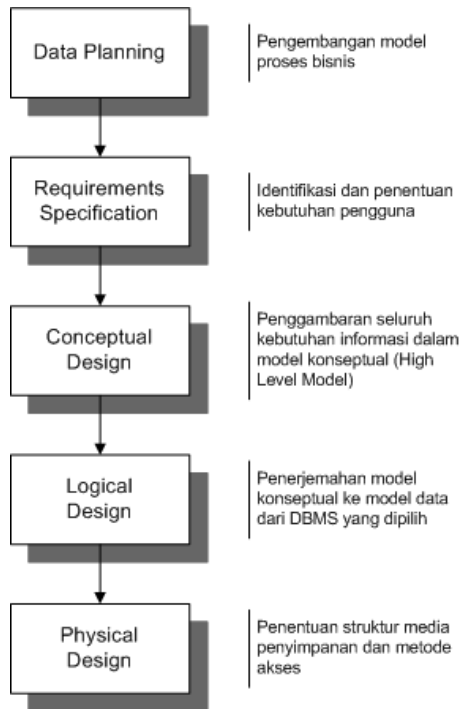


Gambar 14.7. Tipe-tipe basis data (O'Brien, 1999)

- *Data Warehouse*. Tipe ini menyimpan data sekarang dan masa lalu yang merupakan hasil ekstraksi dari berbagai macam operasi dari suatu organisasi. Data yang tersimpan merupakan data yang telah disaring, diedit, distandarisasi dan diintegrasikan sehingga dapat dimanfaatkan oleh manajer dan pengguna lain untuk berbagai macam bentuk analisis, riset dan pengambilan keputusan.
- *Distributed Databases*. Beberapa organisasi menggandakan dan mendistribusikan bagian-bagian dari basis datanya pada beberapa server yang terhubung dengan jaringan. Basis data yang didistribusikan dapat merupakan basis data operational atau analytical, atau tipe-tipe basis data lainnya.
- *End User Databases*. Basis data tipe ini berisi berbagai macam file data yang dikembangkan oleh pengguna akhir pada workstation-nya.

14.4. Tahapan Disain Sistem Basis Data

Ada beberapa model dalam disain basis data, namun secara umum disain sistem basis data mengikuti tahapan-tahapan seperti tampak pada gambar 14.8.



Gambar 14.8. Tahapan disain basis data (diadopsi dari O'Brien, 1999)

14.4.1. Data Planning

Tahapan paling awal dari pengembangan basis data ini bertujuan untuk membuat model organisasi yang mendefinisikan proses-proses bisnis utama dari organisasi tersebut. Database Administrator bersama-sama dengan pengguna akan mengembangkan enterprise model.

14.4.2. Requirements Specification

Tahapan ini bertujuan menentukan kebutuhan-kebutuhan informasi dan pengguna dalam proses bisnis. Selain itu persyaratan-persyaratan juga diidentifikasi.

14.4.3. Conceptual Design

Tahap ini merupakan tahap pengembangan *high-level model* dari hasil formulasi permasalahan dan identifikasi kebutuhan. Biasa juga disebut sebagai tahap pemodelan data. Model proses akan dapat menunjukkan hubungan-hubungan penting antara data/informasi, prosedur, dan komponen pengguna di dalam sistem. Model kemudian dikembangkan dalam bentuk *Entity-Relationship Diagrams* (ERD) yang memodelkan hubungan antar entity yang terlibat dalam sistem. Penjelasan tentang ERD secara lebih detail dapat dilihat pada Bab 15.

14.4.4. Logical Design

Pada tahap desain logis, E-R Diagram yang dihasilkan dari tahapan desain konseptual diterjemahkan ke dalam model basis data yang dipilih, yaitu Basis Data Relasional. Basis Data Relasional merupakan model yang banyak digunakan karena konsistensi dan kemudahan dalam implementasi serta banyak didukung oleh pembuat software DBMS. Pada model data ini basis data akan disebar (dipilah-pilah) ke dalam beberapa tabel 2 dimensi. Secara lebih jelas, pembahasan tentang Basis Data Relasional dapat dilihat pada Bab 16 dan 17.

14.4.5. Physical Design

Desain fisik basis data dilakukan dengan pertimbangan pada beban kerja umum yang diharapkan dapat didukung oleh basis data. Selain itu juga memperbaiki desain basis data dimasa datang untuk memastikan terpenuhinya kriteria performa yang diinginkan. Desain fisik ini sangat bergantung pada kemampuan dan fungsi-fungsi yang dimiliki oleh DBMS yang dipilih.

14.5. Perangkat Lunak DBMS

Ada puluhan bahkan mungkin ratusan perangkat lunak DBMS yang tersedia. Masing-masing dengan spesifikasinya sendiri-sendiri. Mulai dari yang sangat sederhana sampai yang paling kompleks. Pada bagian ini kita akan membahas 5 buah DBMS yang cukup familiar dikalangan pengguna DBMS, yaitu Microsoft Access, MySQL, Microsoft SQL Server, PostgreSQL, dan Oracle.

14.5.1. Microsoft Access

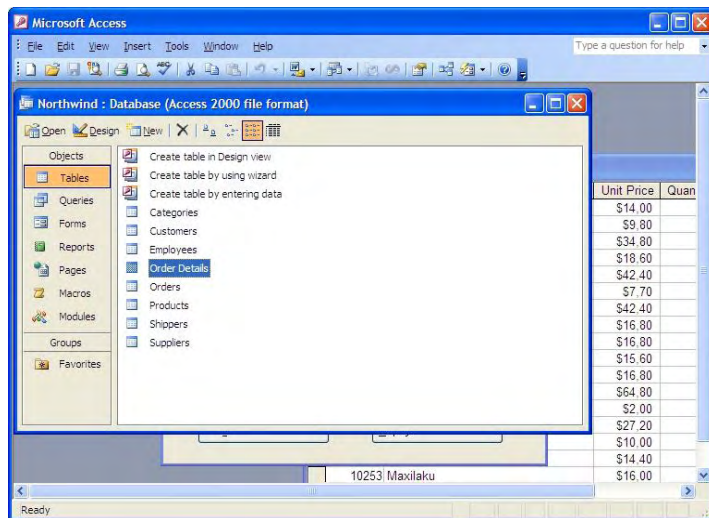


Gambar 14.9.
Logo Microsoft
Access

Microsoft Access atau kadang disebut juga Microsoft Office Access adalah DBMS relational keluaran dari Microsoft yang termasuk dalam paket Microsoft Office. Microsoft Access mengkombinasikan engine relational Microsoft Jet Database, **Graphical User Interface** (GUI) dan perangkat pengembang perangkat lunak. Microsoft Access dapat menggunakan data yang disimpan dalam Microsoft Jet Database, Microsoft SQL Server, Oracle atau tipe lain asal compatible dengan ODBC (*Open Database Connectivity*).

Microsoft Access seringkali digunakan dalam

pengembangan aplikasi secara cepat (**Rapid Application Development**), terutama untuk membangun prototipe dan aplikasi stand-alone. Microsoft Access juga dapat digunakan sebagai basis data untuk aplikasi berbasis web sederhana. Namun pada aplikasi yang lebih kompleks, baik berdasarkan web atau tidak, Microsoft Access bukanlah pilihan yang baik. Terutama karena kekurangannya dalam menangani penggunaan oleh banyak pengguna (**multi-user**). Hal ini karena sebenarnya Microsoft Access adalah sebuah basis data personal yang lebih ditujukan untuk **single-user**. Microsoft Access juga tidak dilengkapi dengan **database triggers** dan **stored procedurs**.



Gambar 14.10. Tampilan Microsoft Access.

Salah satu keuntungan dari Microsoft Access bagi programmer adalah kompatibilitasnya terhadap SQL (structured query language) relatif tinggi. Pada Microsoft Access kita dapat membuat query berbasis teks atau berbasis GUI kemudian dapat langsung dieksekusi dengan mudah untuk mendapatkan hasil. Penggunaan Microsoft Access dapat dilihat pada Bab 18 dan 19.

14.5.2. MySQL

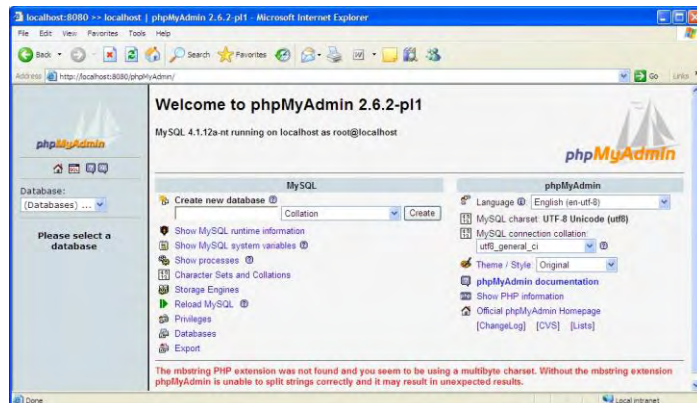
MySQL adalah SQL-DBMS yang bersifat multi-user, dan multithreaded. MySQL berjalan sebagai server yang melayani banyak pengguna untuk mengakses sejumlah basis data. DBMS ini sangat populer di dunia



Gambar 14.11.
Logo MySQL

aplikasi berbasis web sebagai komponen basis data. Selain karena tersedia dalam versi gratis, popularitas MySQL juga sangat dipengaruhi oleh populernya web server Apache dan bahasa pemrograman PHP. Istilah-istilah seperti LAMP (Linux-Apache-MySQL-PHP/Perl/Python), MAMP (Mac-Apache-MySQL-PHP/Perl/Python) dan WAMP (Windows-Apache-MySQL-PHP/Perl/Python) menjadi sangat terkenal. Banyak sekali aplikasi berbasis web yang dibangun dengan menggunakan kombinasi perangkat lunak tersebut. WordPress, Drupal, Mambo, Wikipedia, PHP-Nuke, merupakan beberapa contoh aplikasi berbasis web yang menggunakan kombinasi ini.

Tidak seperti Microsoft Access, default instalasi MySQL tidak menyediakan GUI bagi pengguna untuk berinteraksi dengan basis data. Pengguna dapat berinteraksi dengan client yang menggunakan perintah-perintah berbasis teks (penjelasan lebih lanjut penggunaan perintah berbasis teks pada MySQL dapat dibaca pada Bab 18). Namun saat ini telah banyak GUI yang dikembangkan untuk mempermudah interaksi dengan basis data, baik itu berupa aplikasi stand-alone (misalnya MySQL-Front, MySQL-GUI, dan lain-lain) atau yang berbasis web (misalnya, phpMyAdmin). Bahkan dengan menggunakan komponen MyODBC, MySQL dapat diakses dengan GUI dari Microsoft Access seperti halnya basis data yang kompatibel dengan ODBC lainnya.



Gambar 14.12. Tampilan awal phpMyAdmin.

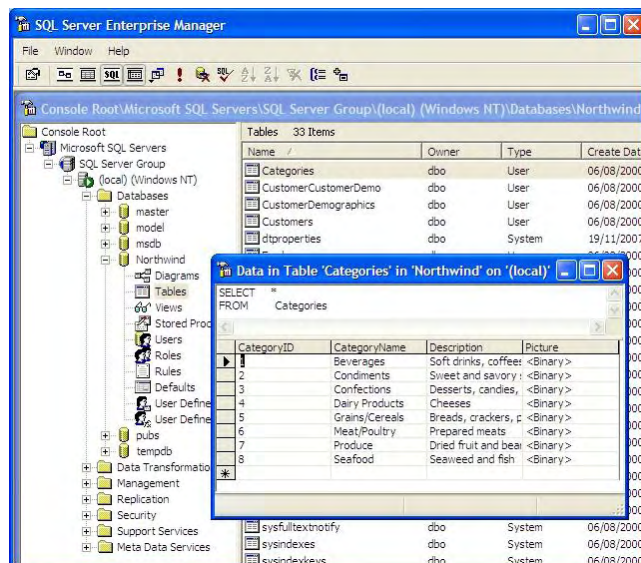
14.5.3. Microsoft SQL Server



Gambar 14.13.
Logo Microsoft
SQL Server

Microsoft SQL Server adalah DBMS relational keluaran dari Microsoft seperti halnya Microsoft Access. Bahasa query utama yang digunakan adalah varian dari ANSI SQL yang disebut sebagai T-SQL (Transact-SQL). Bahasa ini membolehkan pengguna untuk membuat stored procedure sehingga meningkatkan efisiensi akses ke basis data. DBMS ini juga dilengkapi dengan fasilitas clustering dan mirroring. Cluster adalah kumpulan dari server-server yang konfigurasinya identik sehingga memungkinkan pembagian kerja antar server. Sedangkan fasilitas mirroring membolehkan suatu DBMS untuk membuat tiruan (replika) dari isi basis data secara lengkap untuk digunakan pada server yang lain.

Microsoft SQL Server tersedia dalam beberapa versi distribusi. Pada Microsoft SQL Server yang didistribusikan bersama Microsoft Office atau Microsoft Visual Studio, biasa disebut MSDE (Microsoft SQL Server Database Engine), tidak dilengkapi dengan perangkat GUI. Pengguna dapat melakukan interaksi dengan menggunakan client berbasis perintah teks seperti halnya MySQL. Sedangkan pada versi yang lebih tinggi seperti versi personal atau profesional telah tersedia fasilitas GUI (Gambar 14.14).



Gambar 14.14. GUI pada Microsoft SQL Server

14.5.4. PostgreSQL



Gambar 14.15.
Logo
PostgreSQL

PostgreSQL atau sering disebut Postgres termasuk dalam kategori Object-Relational Database Management System (ORDBMS). ORDBMS adalah DBMS yang selain menggunakan prinsip-prinsip basis data relational juga menggunakan pendekatan berorientasi obyek dalam model basis datanya. Postgres dikembangkan sebagai free-software dan bersifat terbuka (open-source) sehingga tidak dikendalikan oleh satu atau dua perusahaan.

Kelebihan Posgres dibandingkan DBMS lainnya adalah, sifatnya yang free dan open-source, dukungan dokumentasinya yang luar biasa, fleksibilitasnya dan feature-featurenya yang tidak kalah dengan DBMS komersial. Selain mendukung model data object-relational, Postgres juga mendukung penggunaan basis data spasial (biasanya untuk penggunaan Sistem Informasi Geografis). Postgres juga mendukung operasi multi-user dan multithreaded, bahkan mungkin lebih bagus dari MySQL dari sisi keamanan.

Seperti halnya MySQL dan Microsoft SQL Server, kita dapat berinteraksi dengan basis data pada Postgres menggunakan perintah-perintah disisi klien dengan tool yang disebut psql. Antar muka yang bersifat GUI juga telah banyak dikembangkan, diantaranya phpPgAdmin, PgAdmin, dan lain-lain.

14.5.5. Oracle Database



Gambar 14.16.
Logo Oracle

Nama Oracle Database atau Oracle RDBMS adalah nama yang sangat diperhitungkan dalam dunia DBMS. Oracle dikembangkan oleh Oracle Corporation.

Oracle menyimpan data secara logika dalam bentuk tablespaces dan secara fisik dalam bentuk file-file data. Tablespaces dapat berisi berbagai macam bagian memori, misalnya bagian data, bagian index dan lain sebagainya. Bagian-bagian ini berisi satu atau lebih area. Area-area ini berisi kumpulan blok data yang berdekatan. Oracle dapat menyimpan dan store procedure dan fungsi secara mandiri.

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Mengidentifikasi Entitas dan Entitas Set.
- Mengidentifikasi Atribut dari suatu entitas
- Mengidentifikasi relasi antar entitas
- Menentukan kardinalitas suatu relasi
- Membuat Entity – Relationship Diagram Sederhana

Pemodelan konseptual basis data merupakan tahap pengembangan high-level model dari hasil formulasi permasalahan dan identifikasi kebutuhan yang dikembangkan pada tahap awal pengembangan basis data. Tahap ini dimulai dengan mengembangkan model awal berupa model proses. Harris (2003) menyebutkan *process model* adalah model yang memfokuskan pada seluruh proses di dalam system yang mentransformasikan data menjadi informasi. Model ini juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Dari model ini akan dapat ditunjukkan hubungan-hubungan penting antara data/informasi, prosedur, dan komponen pengguna di dalam system. Model kemudian dikembangkan dalam bentuk *Entity-Relationship Diagrams* (ERD) yang memodelkan hubungan antar entitas yang terlibat dalam sistem.

Tahapan dalam disain konseptual meliputi :

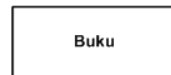
- Identifikasi dan penetapan seluruh himpunan entitas yang terlibat.
- Penentuan atribut-atribut key masing-masing himpunan entitas. Atribut-atribut key yang disertakan di masing-masing himpunan entitas merupakan atribut terpenting yang dapat digunakan untuk mengidentifikasikan (membedakan) setiap entitas yang ada di dalamnya
- Identifikasi dan penetapan seluruh himpunan relasi antar himpunan entitas. Langkah ini merupakan langkah terpenting dalam pembuatan E-R Diagram. Relasi-relasi yang ditetapkan harus dapat mengakomodasikan semua fakta yang ada dan menjamin semua kebutuhan penyajian data. Selain itu pada tahapan ini juga dilakukan penentuan foreign-key untuk masing-masing relasi.

- Penentuan derajat/kardinalitas relasi. Derajat atau kardinalitas menunjukkan banyaknya relasi maksimum yang terjadi antar himpunan entitas.
- Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif. Bagian akhir sebelum kita menggambarkan E-R Diagram adalah melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif. Atribut-atribut deskriptif ini diperoleh dari hasil analisis pada masing-masing entitas dan dibandingkan dengan kondisi dunia nyata hasil dari pengumpulan data.

15.1 DIAGRAM ENTITY-RELATIONSHIP

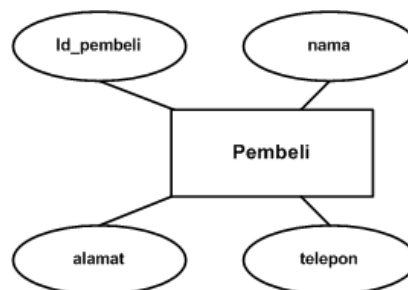
Model E-R biasa digambarkan dalam bentuk diagram yang disebut Entity-Relationship Diagram (ERD). ERD merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan. ERD digunakan untuk memodelkan struktur data dan hubungan antar data, karena hal ini relatif kompleks. Dengan ERD kita dapat menguji model dengan mengabaikan proses yang harus dilakukan. Dan dengan ERD kita mencoba menjawab pertanyaan seperti; data apa yang kita perlukan? bagaimana data yang satu berhubungan dengan yang lain? ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data, pada dasarnya ada 3 macam simbol yang digunakan yaitu :

1. **Entitas** : Entitas digambarkan dalam bentuk persegi empat.



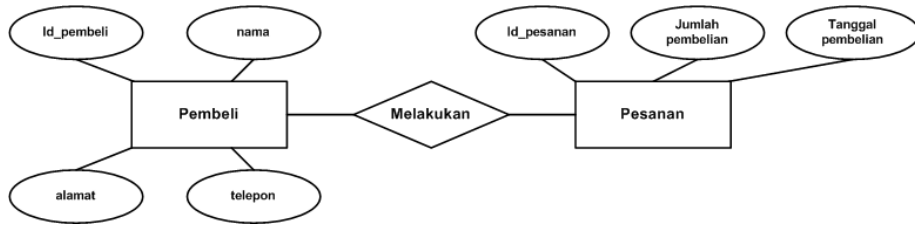
Gambar 15.1. Notasi entitas pada ER-Diagram

2. **Atribut** : Atribut digambarkan dalam bentuk ellips dan dihubungkan dengan entitas dimana atribut tersebut berada.



Gambar 15.2. Penggunaan notasi atribut pada ER-Diagram.

3. **Relationship:** Relationship digambarkan dalam bentuk intan/diamonds.



Gambar 15.3. Penggunaan notasi relationship pada ER-Diagram.

15.2 ENTITAS

Entitas atau *entity* :

individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain

Entitas dapat berupa suatu elemen dari suatu lingkungan, suatu sumber daya atau sebuah transaksi yang memiliki arti penting bagi suatu model yang akan dibangun.

Himpunan entitas atau *entity set*:

Sekumpulan entitas yang sama atau sejenis yang terdapat di dalam lingkup yang sama.

Contoh Entitas set :

- o Semua Guru atau Guru saja.

Himpunan ini memiliki anggota : Bapak Fahri, Ibu Fitri, Bapak Joko dan guru-guru yang lain.

- o Semua Siswa atau Siswa saja.

Himpunan ini memiliki anggota : Joni, Ridho, Fanny, Donny dan siswa-siswa yang lain.

- o Semua Mobil atau Mobil saja.

Himpunan ini memiliki anggota : mobil Daihatsu, mobil Toyota, mobil Suzuki, dan mobil-mobil yang lain.

Mengidentifikasi ada atau tidaknya entitas dalam suatu masalah memang tidak mudah. Tapi biasanya apabila kita menjumpai kata benda dalam permasalahan tersebut maka kata tersebut biasanya merupakan kandidat entitas. Sebagai contoh bila kita akan membangun basis data perpustakaan sekolah, maka kita akan menjumpai buku, siswa, guru, petugas perpustakaan sebagai kandidat kuat entitas.

15.3 ATRIBUT

Setelah kita dapat mengidentifikasi entitas maka kita juga harus memperhatikan atribut dari entitas tersebut. Setiap entitas mempunyai atribut.

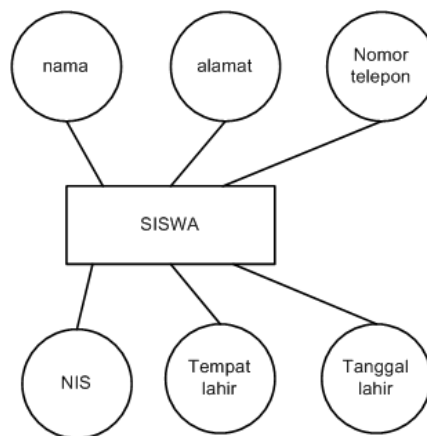
Atribut :

Karakteristik atau ciri yang membedakan antara entitas satu dengan entitas yang lainnya.

Contoh Atribut :

- o Entitas Siswa.

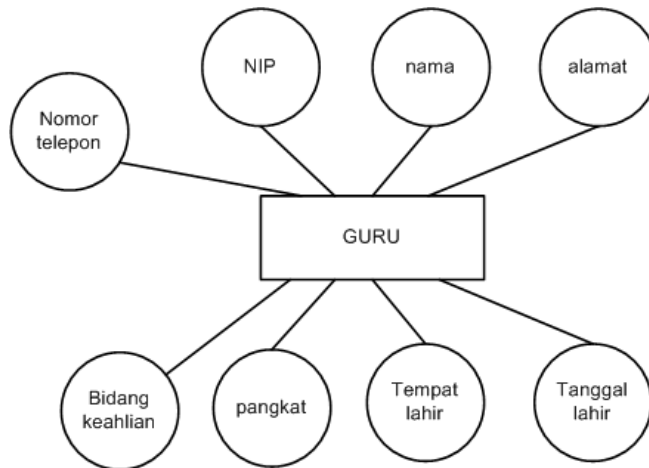
Memiliki atribut antara lain : nis (nomor induk siswa), nama, alamat, nomor telepon, tempat lahir, tanggal lahir dan lain-lain.



Gambar 15.4. Entitas siswa dan atributnya.

- o Entitas Guru.

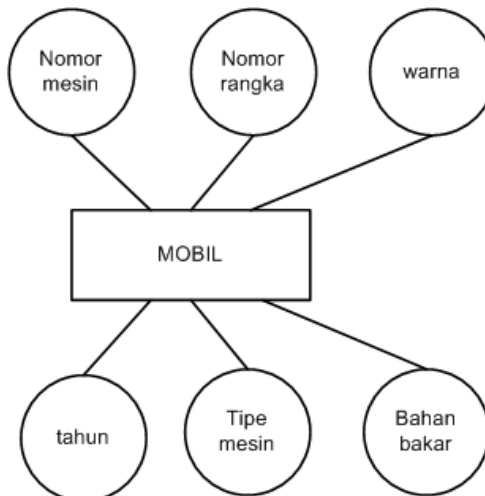
Memiliki atribut antara lain : NIP, nama, alamat, pangkat, nomor telepon, tempat lahir, tanggal lahir, bidang keahlian, dan lain-lain.



Gambar 15.5. Entitas guru dan atributnya.

- o Entitas Mobil.

Memiliki atribut antara lain : Nomor mesin, nomor rangka, warna, tahun keluar, tipe mesin, bahan baker, dan lain-lain.



Gambar 15.6. Entitas mobil dan atributnya.

Tidak semua karakteristik dari entitas penting bagi suatu ruang lingkup masalah. Sebagai contoh pada masalah basis data perpustakaan, karakteristik nomor sepatu siswa bukanlah karakteristik yang penting yang dapat dijadikan sebagai atribut. Tetapi apabila ruang lingkup masalahnya adalah basis data pemesanan seragam sepatu siswa, maka nomor sepatu adalah atribut yang penting.

Selain memilih mana yang benar-benar penting bagi sebuah entitas, kita juga harus menentukan mana yang akan menjadi atribut kunci (Primary Key).

Atribut kunci (Primary Key) :

Atribut yang dapat digunakan sebagai pengidentifikasi suatu entitas dan bersifat paling unik di antara atribut yang lain.

Pada contoh di atas (contoh atribut) kita dapat dengan mudah menentukan primary key dari entitas yang ada.

Contoh Atribut kunci (Primary key). :

- Entitas Siswa.

Misalkan atribut yang dimiliki adalah : NIS (nomor induk siswa), nama, alamat, nomor telepon, tempat lahir, dan tanggal lahir. Dari keenam atribut ini, maka yang paling cocok menjadi primary key adalah NIS karena atribut ini yang paling unik. Tidak ada siswa yang memiliki NIS yang sama. Nama masih mungkin sama, tapi NIS tidak.

- Entitas Guru.

Misalkan atribut yang dimiliki adalah: NIP, nama, alamat, pangkat, nomor telepon, tempat lahir, tanggal lahir, dan bidang keahlian. Atribut yang paling cocok menjadi primary key adalah NIP karena atribut ini yang paling unik. Tidak ada guru yang memiliki NIP yang sama. Sehingga NIP dapat dijadikan pengidentifikasi entitas guru.

- Entitas Mobil.

Misalkan atribut yang dimiliki adalah: Nomor mesin, nomor rangka, warna, tahun keluar, tipe mesin, dan bahan bakar. Di sini ada dua atribut yang unik yaitu nomor mesin dan nomor rangka. Pada kasus ini kita dapat memilih salah satu dari kedua atribut tersebut sebagai primary key.

Sebagai panduan, berikut ini merupakan ciri-ciri dari atribut yang dapat dipertimbangkan sebagai identifier (candidate key) :

- Nilainya tidak berubah-ubah
- Tidak mungkin berisi nilai null (kosong bukan nol)
- Tidak berisi data nama atau lokasi yang mungkin berubah-ubah.

15.4 RELATIONSHIP

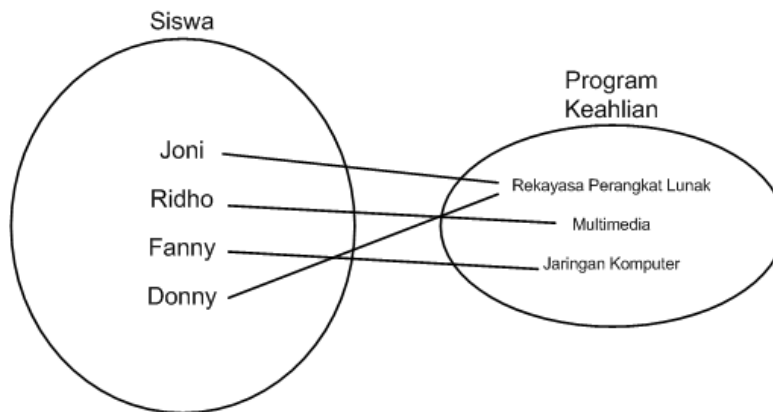
Relationship atau relasi :

Hubungan yang terjadi antara sejumlah entitas.

Relationship set

Kumpulan semua hubungan antara entitas yang terdapat pada entitas set yang berbeda

Misalkan dari entitas siswa ada seorang siswa yang memiliki *NIS* = “GHI007” dan *nama_siswa* = “Donny” mempunyai relasi dengan entitas program keahlian dengan *kode_program* = “RPL” dan *nama_program* = “Rekayasa Perangkat Lunak”. Relasi di antara kedua entitas mengandung arti siswa tersebut sedang mengambil program keahlian tersebut pada sekolah tertentu.



Gambar 15.7. Relationship.

Ramakrishnan and Gehrke (2000) menyebutkan bahwa konsep *relationship* pada model E-R berbeda dengan konsep *relation* di dalam model data relasional. *Relationship* adalah mekanisme yang menghubungkan antara entitas. Dalam implementasi ke dalam DBMS baik entitas maupun *relationship* akan direpresentasikan dalam bentuk tabel (relation).

15.5 KARDINALITAS

Kardinalitas atau Derajat Relasi :

Menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas lain pada himpunan entitas yang lain.

Pada gambar 15.7 kita sebenarnya dapat melihat sebuah kardinalitas antara himpunan entitas siswa dengan himpunan entitas program keahlian. Siswa dapat berelasi hanya dengan satu entitas pada himpunan entitas program keahlian. Sebaliknya satu entitas pada program keahlian dapat berelasi dengan banyak siswa. Pada gambar tersebut terlihat Donny hanya dapat berhubungan dengan Rekayasa Perangkat Lunak, sedangkan Rekayasa Perangkat Lunak dapat berhubungan dengan Donny dan Joni.

Ada beberapa jenis tingkat hubungan (kardinalitas) antara entitas satu dengan entitas lainnya. Kardinalitas menunjukkan jumlah maksimum entitas pada suatu himpunan entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Secara umum ada tiga bentuk kardinalitas antar himpunan entitas, yaitu :

- **satu ke satu (one to one).** Hubungan dengan kardinalitas one-to-one artinya satu anggota suatu entitas set hanya boleh berhubungan dengan satu anggota entitas set yang lain. Hubungan antara entitas set suami dengan istri dapat dikelompokkan dalam hubungan one-to-one.



Gambar 15.8. Hubungan one-to-one suami dan istri.

- **satu ke banyak / banyak ke satu (one to many / many to one).** Kardinalitas satu ke banyak dan banyak ke satu dapat dianggap sama karena tinjauan kardinalitas selalu dilihat dari dua sisi. Contohnya adalah pada suatu sekolah mempunyai aturan satu kelas terdiri dari banyak siswa tetapi tidak sebaliknya, yaitu satu siswa tidak dapat belajar pada kelas yang berbeda.



Gambar 15.9. Hubungan one-to-many kelas dengan siswa.

- **banyak ke banyak (many to many).** Kardinalitas ini cukup rumit untuk dijelaskan namun seringkali kita jumpai. Misalnya hubungan antara mahasiswa dengan matakuliah memiliki kardinalitas many-to-many. Mahasiswa berhak mengambil (mempelajari) lebih dari satu matakuliah dan setiap matakuliah boleh diambil (dipelajari) lebih dari satu mahasiswa.

Basis Data Relasional

TUJUAN

Setelah anda mempelajari bab ini diharapkan anda akan mampu :

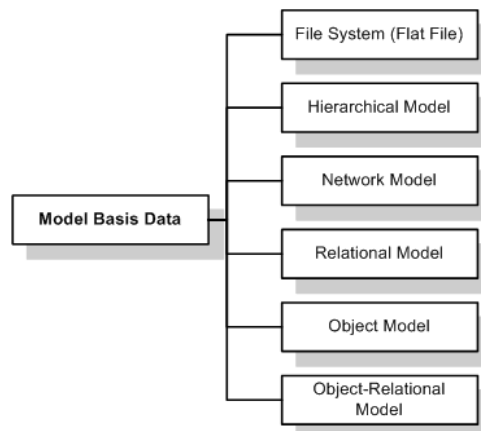
- Mengetahui jenis-jenis model basis data.
- Memahami pengertian basis data relasional.
- Menerapkan prinsip-prinsip pembuatan table, baris, kolom dan relasi antar table yang benar.
- Menerapkan prinsip-prinsip normalisasi dalam pembuatan table.
- Memahami prinsip-prinsip pembuatan query.

16.1 MODEL BASIS DATA

Model Basis Data :

Model yang digunakan dalam menjelaskan / menspesifikasikan bagaimana struktur basis data dan penggunaanya.

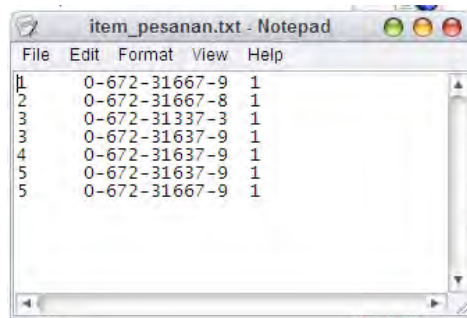
Ada beberapa model basis data yang dikenal, seperti terlihat pada Gambar 16.1.



Gambar 16.1. Model basis data (Powell, 2006).

16.1.1 File System (Flat File)

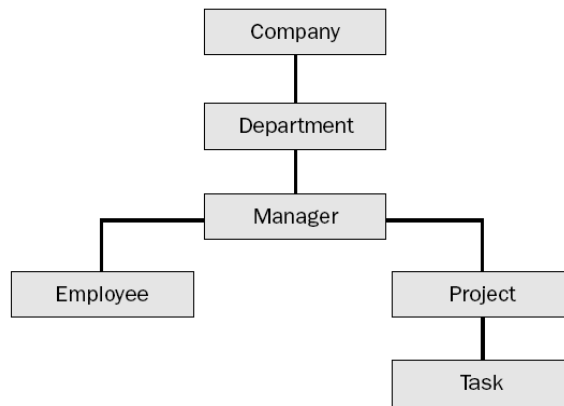
Penggunaan file system sebagai tempat menyimpan basis data sebenarnya tidak mengikuti pola atau model apapun. Artinya data hanya disimpan dalam bentuk mendatar (flat) tanpa ada aturan-aturan tertentu. Tipe file yang digunakan adalah file teks sederhana. Dalam Microsoft Windows, file ini biasanya diakhiri dengan extension txt. Kita dapat membuka file ini dengan notepad atau software pengolah teks. Gambar 16.2. menunjukkan bagaimana sebuah flat file menyimpan data. Data-data dalam file ini dapat diolah dengan menggunakan bahasa pemrograman, seperti ditunjukkan pada bab 11.



Gambar 16.2. Flat file.

16.1.2 Hierarchical Model

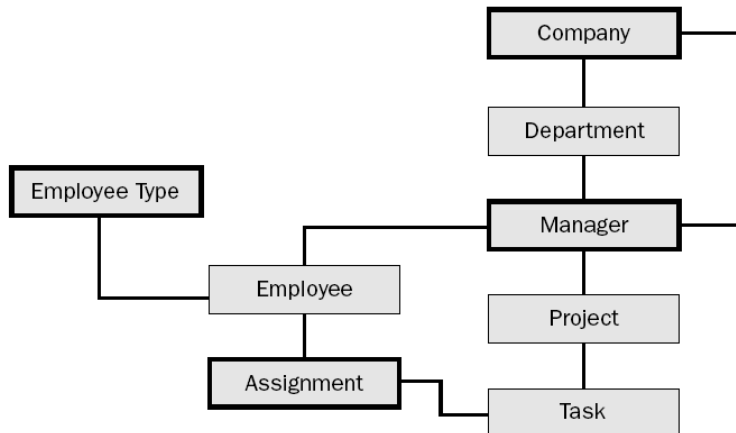
Hierarchical model adalah model basis data yang menggunakan struktur seperti pohon (Gambar 16.3). Model ini menggunakan pendekatan orang tua – anak. Artinya table pada anak tergantung sepenuhnya pada table orang tua. Pada Gambar 16.3 terlihat bagaimana struktur basis data suatu perusahaan. Employee sangat tergantung pada Manager, dan Manager tergantung pada Departement, dan seterusnya. Employee adalah table anak bagi Manager, sedangkan Manager adalah table orang tua bagi Employee dan Project. Sedangkan Manajer adalah table anak bagi Department, sedangkan Department adalah table orang tua bagi Manajer. Untuk mengakses data pada Employee kita harus lebih dulu mengakses secara berturut-turut table Company → Department → Manager → Employee.



Gambar 16.3. Hierarchical model (Powell, 2006)

16.1.3 Network Model

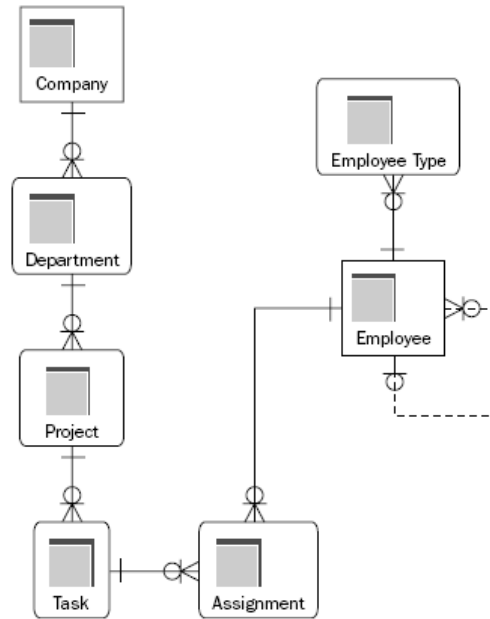
Network model merupakan pengembangan dari hierarchical model. Pada model ini diberikan perluasan, yaitu table anak boleh memiliki lebih dari satu table orang tua sehingga akan membentuk sebuah jaringan (Gambar 16.4).



Gambar 16.4. Network model (Powell, 2006).

16.1.4 Relational Model

Relational model merupakan perbaikan dari hierarchical dan network model dengan memperbolehkan setiap table berhubungan apabila diperlukan tanpa harus melewati table orang tuanya (Gambar 16.5). Pada buku ini, hanya relational model yang akan kita pelajari.



Gambar 16.5. Relational Model (Powell, 2006).

16.1.5 Object Model

Object database model merupakan model basis data berstruktur tiga dimensi. Model ini membolehkan pengguna untuk mengambil / melihat data dari sisi manapun. Pendekatan yang dilakukan diilhami oleh pendekatan berorientasi object. Secara konsep object model sangat baik namun dalam penerapannya, kinerjanya sangat rendah.

16.1.6 Object-Relational Model

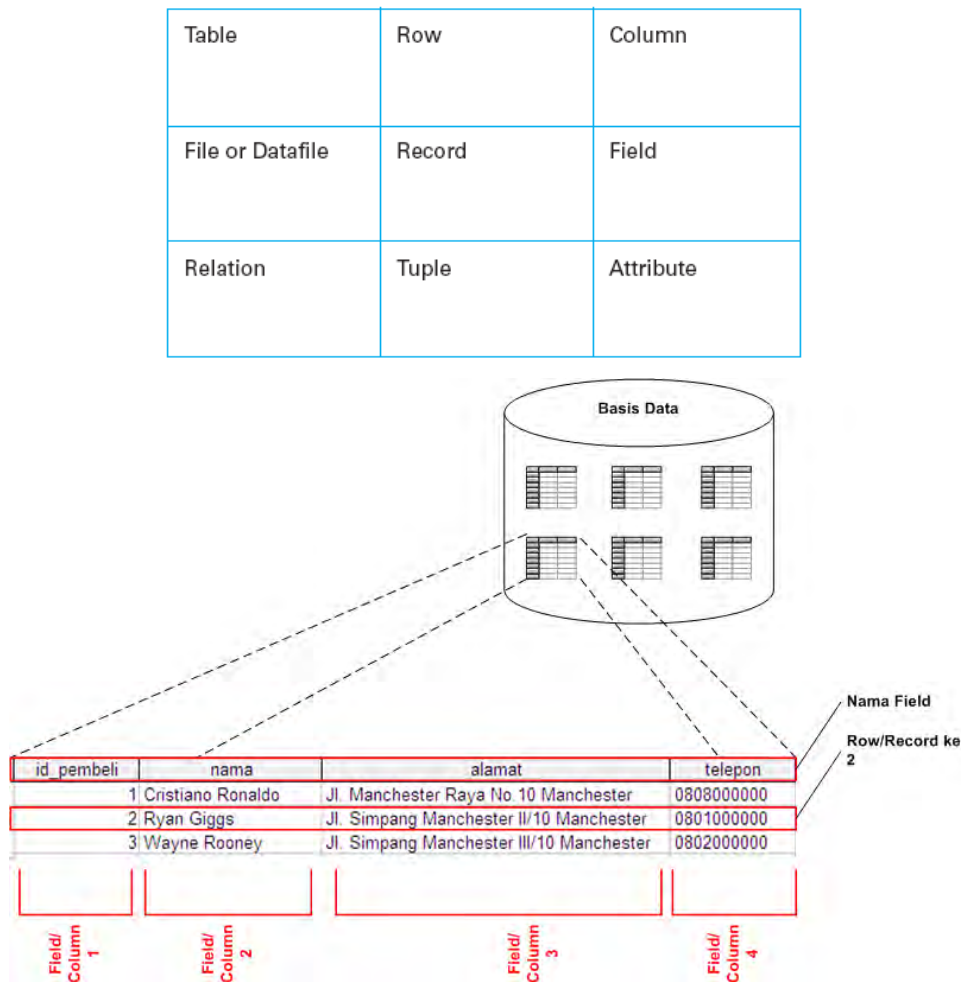
Model ini merupakan gabungan dari relational model dan object model. Tujuan utamanya adalah untuk memperbaiki kinerja pada object model.

16.2 BASIS DATA RELATIONAL

Model basis data relasional diperkenalkan pertama kali oleh E.F. Codd pada tahun 1970. Model data ini didasarkan pada struktur matematis yang mudah dan alami, yaitu *relation* (tabel). Operasi-operasi manipulasi data semuanya berakar pada logika matematika. Hal ini menjadikan ekspresi-ekspresi pada tabel dapat dianalisis dan dioptimasi (Lewis et al., 2002).

Pembentuk utama dalam model data relasional adalah *relation* (tabel). *Relation* terdiri dari dua hal penting yaitu *schema* dan *instance*. *Relation*

instance tidak lebih dari sebuah table dua dimensi dengan baris dan kolom. Baris (row) biasa disebut sebagai tuple, yang memiliki arti sama dengan record dalam suatu file. Tetapi berbeda dengan file record, semua tuple memiliki jumlah kolom yang sama dan tidak ada tuple dalam *relation instance* yang sama. Kolom dalam *relation instance* juga dikenal sebagai *attribute* atau *column* (Ramakrishnan and Gehrke, 2000; Lewis et al., 2002). Gambar 16.6. menunjukkan bagaimana hubungan table/file/relation, row/record/tuple dan column/field/attribute. Gambar ini juga menunjukkan hierarki dari hubungan tersebut.



Gambar 16.6. hubungan table/file/relation, row/record/tuple dan column/field/attribute.

Relation schema terdiri dari nama dari *relation*, nama dari *attribute* yang ada pada suatu *relation* beserta nama domainnya, dan *integrity constraints*. Nama dari *relation* haruslah unik dalam suatu basis data, atau tidak boleh ada nama *relation* yang sama. Nama *attribute* adalah nama kolom dari *relation* dan tidak ada nama *attribute* yang sama pada suatu *relation*. Nama domain dari suatu *attribute* berhubungan dengan tipe data yang digunakan oleh *attribute* tersebut. *Integrity constraints* adalah batasan pada *relational instances* pada suatu *schema* (Ramakrishnan and Gehrke, 2000; Lewis et al., 2002)

16.2.1 Row/Baris/Tuple

Row/Baris/Tuple adalah sekumpulan atribut yang saling berhubungan dalam satu baris (lihat Gambar 16.6). Row bisa juga disebut sebagai record. Row ini akan selalu berulang dengan struktur yang sama namun dengan isi data yang berbeda. Sebagai contoh, pada Gambar 16.6, row pertama memiliki struktur yang sama dengan row yang ke 2 dan ke 3. Namun data pada masing-masing row berbeda. Kita dapat mengidentifikasi untuk pembeli dengan `id_pembeli = 1` pasti memiliki nama = Cristiano Ronaldo dan untuk `id_pembeli = 2` pasti memiliki nama = Ryan Giggs, demikian seterusnya.

16.2.2 Field/Kolom/Attributes

Field/Kolom/Attributes menunjukkan struktur dari data dari baris-baris yang berulang. Pada Gambar 16.6, terdapat 4 kolom, yaitu `id_pembeli`, nama, alamat, dan telepon. Data pada kolom `id_pembeli` misalnya, akan memiliki struktur yang sama, yaitu dalam bentuk angka dan merupakan urutan id pembeli. Demikian juga pada kolom nama yang hanya berisi nama pembeli saja, tidak bercampur dengan data lain.

Sebuah kolom harus memiliki nama kolom dan tipe data untuk data yang berada dalam kolom tersebut (Gambar 16.7). Selain itu, kadang-kadang juga ada pembatas (constraint) dan domain untuk data yang termasuk dalam kolom tersebut. Tipe data akan sangat bergantung pada atribut yang digunakan dan operasi-operasi yang akan dilakukan pada basis data ini. Domain data memiliki banyak kesamaan pengertian dengan fungsi tipe data yang digunakan. Namun, tipe data lebih merujuk pada kemampuan penyimpanan data yang mungkin bagi suatu atribut secara fisik, tanpa melihat layak tidaknya data tersebut bila dilihat dari pemakaian di dunia nyata. Sementara domain data lebih ditekankan pada batas-batas nilai yang diperbolehkan bagi suatu atribut, dilihat dari kenyataan pemakaiannya.

Name	Null?	Type
ISBN	NOT NULL	INTEGER
PUBLISHER_ID	NOT NULL	INTEGER
PUBLICATION_ID	NOT NULL	INTEGER
PRINT_DATE		DATE
PAGES		INTEGER
LIST_PRICE		INTEGER
FORMAT		VARCHAR (32)
RANK		INTEGER
INGRAM_UNITS		INTERGER

Diagram labels: 'Column names' points to the Name column, 'NULL constraint' points to the Null? column, and 'Datatype' points to the Type column.

Gambar 16.7. Kolom, constraint dan tipe data (Powell, 2006).

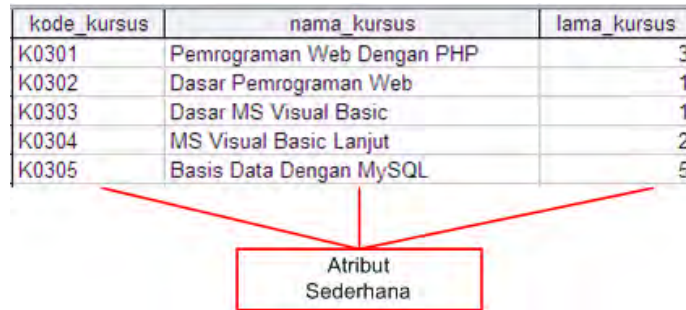
Pada Gambar 16.7 terlihat nama-nama kolom pada sebuah table lengkap dengan tipe data dan constraintnya. Kolom ISBN misalnya bertipe data integer, artinya kolom ini hanya boleh diisi dengan bilangan integer. Selain itu kolom ini memiliki constraint no null, yang berarti ketika mengisi data, kolom ini harus selalu terisi tidak boleh dikosongkan. Pada kolom Pages, tipe datanya adalah date, artinya hanya boleh berisi data berbentuk tanggal. Kolom Pages tidak memiliki constraint null, artinya kolom ini boleh diisi atau tidak diisi.

Jenis-jenis tipe data tergantung pada DBMS yang digunakan. Sebagai contoh MySQL memberikan jenis-jenis tipe data yang lebih luas dari pada Microsoft Access. Pada MySQL misalnya, kita akan menjumpai tipe data set, enum yang tidak dijumpai dalam Microsoft Access. Pemilihan tipe data yang tepat sangat penting karena mempengaruhi konsistensi data dan kinerja basis data. Pada bab 17 dan 18 kita akan mempelajari penggunaan tipe-tipe data ini.

Contoh domain adalah, apabila kita berhadapan dengan atribut / kolom tentang kelas pada Sekolah Dasar (SD). Kolom kelas ini hanya boleh diisi data angka 1 sampai dengan 6, karena tidak ada kelas 7 atau 4.5 di SD. Artinya domain kolom kelas adalah bilangan integer (bulat dan tidak ada pecahan) antara 1 sampai dengan 6 saja. Contoh lainnya adalah kolom nilai ujian, kolom ini domainnya adalah bilangan asli (real dan boleh pecahan) antara 0 sampai dengan 100.

Ada beberapa tipe atribut, yaitu:

- Atribut sederhana (*Simple Attribute*), yaitu atribut atomic yang tidak dapat dipecah lagi.



Gambar 16.8. Contoh atribut sederhana

- Atribut komposit (*Composit Attribute*), yaitu atribut yang masih dapat dipecah lagi menjadi sub-sub atribut yang masing-masing memiliki makna.



Gambar 16.9. Contoh atribut komposit.

- Atribut bernilai tunggal (*Single-Valued Attribute*), yaitu atribut yang memiliki paling banyak satu nilai untuk setiap baris data.
- Atribut bernilai banyak (*Multi-Valued Attribute*), yaitu atribut yang dapat berisi lebih dari satu nilai tetapi dengan jenis yang sama.

kode_instruktur	nama_instruktur	alamat_instruktur	keahlian_instruktur
I04001	Felipe Scolari	Jl. Terusan Sao Paulo III/67 Rio de Janeiro	Sistem Basis Data
I04002	Sven Goran Eriksson	Jl. Upsalla Raya 23 Gothenborg	Pemrograman PHP
I04003	Steve Mc Laren	Jl. Merseyside Utara 12 Liverpool	Pemrograman Visual
			Pemrograman Terstruktur
			Pemrograman Shell

Atribut bernilai tunggal

Atribut bernilai banyak

Gambar 16.10. Contoh atribut bernilai tunggal dan atribut bernilai banyak.

- Atribut turunan (Derived Attribute), yaitu atribut yang nilai-nilainya diperoleh dari hasil pengolahan atau dapat diturunkan dari atribut atau table lain yang berhubungan.

16.3 RELASI ANTAR TABEL

Keistimewaan utama basis data relasional dibandingkan model basis data lainnya adalah kemudahan dalam membangun hubungan antar table dalam bentuk yang masuk akal dapat dimengerti. Relasi antar table dapat kita turunkan langsung ataupun tak langsung dari ER-Diagram yang telah dibahas pada bab 15. Entitas yang ada pada ER-Diagram biasanya merupakan kandidat dari suatu table pada basis data relasional. Relasi antar table biasanya dapat diidentifikasi dari relationship antar entitas pada ER-Diagram.

Untuk memperjelas bagaimana basis data relasional menangani hubungan antar table kita akan menggunakan table-table berikut ini. Tabel pertama adalah Pengarang. Table ini terdiri dari 3 kolom yaitu id_pengarang, pengarang dan tahun_lahir (Gambar 16.11). Pada table ini id_pengarang merupakan primary key.

Pengarang : Table		
id_pengarang	pengarang	tahun_lahir
1	Adams, Pat	
2	Adrian, Merv	
3	Ageloff, Roy	1943
4	Andersen, Virginia	
5	Antonovich Michael P.	
6	Arnott, Steven E.	
7	Arntson, L. Joyce	
8	Ault, Michael R	
9	Avison, D. E.	
10	Bard, Dick	1941
11	Biegel, Richard A	
12	Biow, Lisa	
13	Bisland, Ralph B	
14	Bowman, Judith S	

Gambar 16.11. Tabel Pengarang.

Tabel kedua adalah Penerbit. Table ini mempunyai 6 kolom yaitu, id_penerbit, nama, nama_perusahaan, alamat, kota dan telepon (Gambar 16.12). Primary key pada table ini adalah id_penerbit.

Penerbit : Table					
id_penerbit	nama	nama_perusahaan	alamat	kota	telepon
1	ACM	Association for Computing Machinery	11 W. 42nd St., 3rd flr.	New York	212-869-7440
2	Addison-Wesley	Addison-Wesley Publishing Co Inc.	Rte 128	Reading	617-944-3700
3	Bantam Books	Bantam Books Div of: Bantam Doubleday De	666 Fifth Ave	New York	800-223-6834
4	Benjamin/Cummings	Benjamin-Cummings Publishing Company S	390 Bridge Pkwy.	Redwood City	800-950-2665
5	Brady Pub.	Brady Books Div. of Prentice Hall Pr., Simon	15 Columbus Cir.	New York	212-373-8093
6	Computer Science Press	Computer Science Press Inc Imprint of W H	41 Madison Ave	New York	212-576-9400
7	ETN Corporation	ETN Corp.	RD 4, Box 659	Montoursville	717-435-2202
8	Gale	Gale Research, Incorporated	835 Penobscot Bldg	Detroit	313-961-2242
9	IEEE	IEEE Computer Society Press	10662 Los Vaqueros Circle	Los Alamitos	800-272-6657
10	Intertext	Intertext Publications/Multiscience Press	2633 E. 17th Ave.	Anchorage	
11	M&T Books	M & T Books Div of: M&T Publishing Inc	501 Galveston Dr	Redwood City	800-533-4372
12	Macmillan Education	Macmillan Education Ltd	175 Fifth Ave	New York	212-460-1500
13	McGraw-Hill	McGraw-Hill Inc	1221 Ave of the Americas	New York	212-512-2000
14	Microsoft Press	Microsoft Press Div of: Microsoft Corp	One Microsoft Way	Redmond	800-MSPRESS
15	Morgan Kaufmann	Morgan Kaufmann Publishers Inc.	2929 Campus Dr, Suite 260	San Mateo	415-578-9911

Gambar 16.12. Tabel Penerbit.

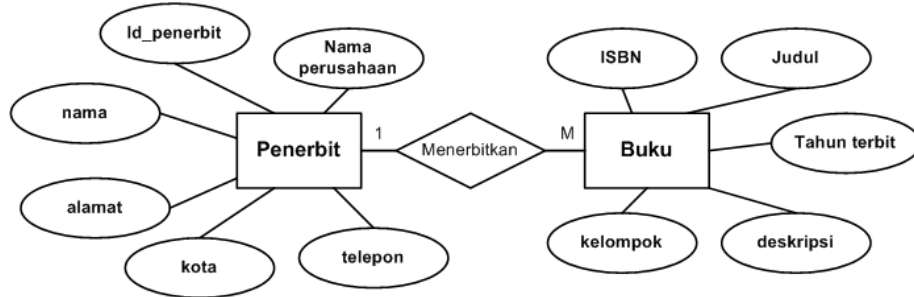
Tabel ketiga adalah Buku. Table ini mempunyai 6 kolom yaitu, judul, tahun_terbit, ISBN, id_penerbit, deskripsi, dan kelompok (Gambar 16.13). Primary key pada table ini adalah ISBN. Ada yang sedikit aneh pada table ini, yaitu kolom id_penerbit yang merupakan salah satu kolom pada table Penerbit, dimasukkan dalam table ini. Sebenarnya ini bukan keanehan atau kesalahan, tetapi memang beginilah salah satu cara basis data relasional menangani hubungan antar table.

Buku : Table					
judul	tahun_terbit	ISBN	id_penerbit	deskripsi	kelompok
Database management: developing application systems using	1989	0-0131985-2-1	17	xx, 441 p. : ill. ;	
Select-- SQL : the relational database language	1992	0-0238669-4-2	12	xv, 446 p. ; 24 c	
dBase IV programming	1994	0-0280042-4-8	73		
Step-by-step dBase IV	1995	0-0280095-2-5	52		
Guide to ORACLE	1990	0-0702063-1-7	13	xii, 354 p. : ill. ; 4	ORACLE (Computer syste
The database experts' guide to SQL	1988	0-0703900-6-1	10		
Oracle/SQL: a professional programmer's guide	1992	0-0704077-5-4	13	xx, 543 p. : ill. ;	
SQL 400: A Professional Programmer's Guide	1994	0-0704079-9-1	52		
Database system concepts	1986	0-0704475-2-7	13		
Microsoft FoxPro 2.5 applications programming	1993	0-0705015-3-X	61	xiii, 412 p. : ill. ;	
First look at-- dBASE IV, version 1.5/2.0 for DOS	1994	0-0705107-5-X	80	ill. ; 24 cm.	
Applying SQL in Business	1992	0-0705184-2-4	13		
Database design	1977	0-0707013-0-X	13		
Introduction to Oracle	1989	0-0770716-4-6	13	xi, 342 p. ; 24 c	

Gambar 16.13. Tabel buku.

Dalam dunia nyata, kita akan menjumpai bahwa satu penerbit tidak hanya menghasilkan satu judul buku saja, tetapi ratusan bahkan mungkin jutaan judul buku. Sehingga secara formal hubungan antara penerbit dengan buku dapat dinyatakan sebagai hubungan dengan kardinalitas one-to-many. Jika digambarkan dalam bentuk ER-Diagram akan tampak seperti Gambar 16.14. Satu penerbit dapat menerbitkan banyak judul buku dan satu judul

buku hanya diterbitkan oleh satu penerbit. Untuk lebih memperjelas perhatikan Gambar 16.15. Penerbit dengan id_penerbit = 13 (McGraw Hill) menerbitkan 6 buah buku. Sebaliknya buku dengan ISBN = 0-0702063-1-7 (Guide To Oracle) hanya diterbitkan oleh penerbit dengan id_penerbit = 13 (McGraw Hill). Sehingga penempatan kolom id_penerbit pada table buku dimaksudkan untuk merepresentasikan hubungan Penerbit dengan Buku. Kolom id_penerbit pada table buku biasa disebut sebagai *foreign key*.



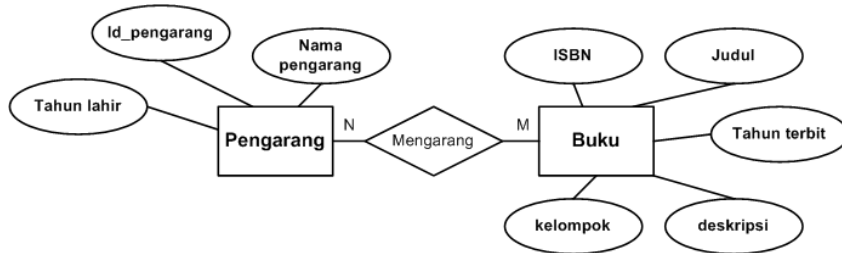
Gambar 16.14. ER-Diagram untuk Penerbit dan Buku

Penerbit : Table					
id_penerbit	nama	nama_perusahaan	alamat	kota	telepon
1	ACM	Association for Computing Machinery	11 W. 42nd St. 3rd flr.	New York	212-869-7440
2	Addison-Wesley	Addison-Wesley Publishing Co Inc.	Rte 128	Reading	617-944-3700
3	Bantam Books	Bantam Books Div of Bantam Doubleday De	666 Fifth Ave	New York	800-223-6834
4	Benjamin/Cummings	Benjamin-Cummings Publishing Company S	390 Bridge Pkwy.	Redwood City	800-950-2665
5	Brady Pub.	Brady Books Div. of Prentice Hall Pr.	Simor	New York	212-373-8093
6	Computer Science Press	Computer Science Press Inc Imprint of W H	41 Madison Ave	New York	212-576-9400
7	ETN Corporation	ETN Corp.	RD 4, Box 659	Montoursville	717-435-2202
8	Gale	Gale Research, Incorporated	835 Penobscot Bldg	Detroit	313-961-2242
9	IEEE	IEEE Computer Society Press	10662 Los Vaqueros Circle	Los Alamitos	800-272-6657
10	Intertext	Intertext Publications/Multiscience Press	2633 E. 17th Ave	Anchorage	
11	M&T Books	M & T Books Div of M&T Publishing Inc	501 Galveston Dr	Redwood City	800-533-4372
12	Macmillan Education	Macmillan Education Ltd.	175 Fifth Ave	New York	212-460-1500
13	McGraw-Hill	McGraw-Hill Inc	1221 Ave of the Americas	New York	212-512-2000
14	Microsoft Press	Microsoft Press Div of Microsoft Corp	One Microsoft Way	Redmond	800-MSPRESS
15	Morgan Kaufmann	Morgan Kaufmann Publishers Inc.	2929 Campus Dr. Suite 260	San Mateo	415-578-9911

Buku : Table					
judul	tahun_terbit	ISBN	id_penerbit	deskripsi	kelompok
Database management, developing application systems usin	1989	0-0131985-2-1	17	xx, 441 p. : ill. :	
Select-- SQL : the relational database language	1992	0-0238669-4-2	12	xv, 446 p. : 24 c	
dBase IV programming	1994	0-0280042-4-8	73		
Step-by-step dBase IV	1995	0-0280095-2-5	52		
Guide to ORACLE	1990	0-0702063-1-7	13	354 p. : ill. :	ORACLE (Computer syste
The database experts' guide to SQL	1988	0-0703900-6-1	10		
Oracle/SQL: a professional programmer's guide	1992	0-0704077-5-4	13	543 p. : ill. :	
SQL 400: A Professional Programmer's Guide	1994	0-0704079-9-1	52		
Database system concepts	1986	0-0704475-2-7	13		
Microsoft FoxPro 2.5 applications programming	1993	0-0705015-3-X	61	xiii, 412 p. : ill. :	
First look at-- dBASE IV, version 1.5/2.0 for DOS	1994	0-0705107-5-X	80	ill. : 24 cm.	
Applying SQL in Business	1992	0-0705184-2-4	13		
Database design	1977	0-0707013-0-X	13		
Introduction to Oracle	1989	0-0770716-4-6	13	xi, 342 p. : 24 ci	

Gambar 16.15. Hubungan table Penerbit dan Buku.

Pada table-tabel yang telah dibuat di atas kita belum melihat table yang menunjukkan seorang pengarang tertentu mengarang buku apa. Untuk membuat table yang berisi pengarang dan buku karangannya, kita dapat menggunakan hubungan antara table pengarang dengan table buku. Namun sebelum itu kita harus melihat bagaimana hubungan antara pengarang dengan buku di dunia nyata. Seorang pengarang mungkin hanya mengarang satu judul buku, tetapi mungkin juga lebih. Sedangkan satu buku, mungkin ditulis hanya oleh satu orang pengarang, tetapi mungkin juga ditulis oleh dua, tiga atau lebih pengarang. Sehingga kita bisa mengatakan kardinalitas hubungan pengarang dengan buku adalah many-to-many. Kita dapat menggambarkan ER-Diagram untuk kasus ini seperti pada gambar 16.16.



Gambar 16.16. ER-Diagram untuk Pengarang – Buku.

Pada kasus dengan kardinalitas many-to-many kita tidak dapat langsung menyisipkan satu foreign key pada table lain. Kita harus membuat table baru agar kardinalitas antar table yang terlibat dapat diubah menjadi one-to-many. Table Pengarang_Buku merupakan table yang dibentuk untuk menangani hubungan table Buku dengan table Pengarang. Tabel ini hanya berisi dua atribut (kolom) yaitu ISBN yang berasal dari table Buku dan id_pengarang yang berasal dari table Pengarang. Pada Gambar 16.17, terlihat pada table Pengarang_Buku ada beberapa buku yang dikarang lebih dari satu pengarang.

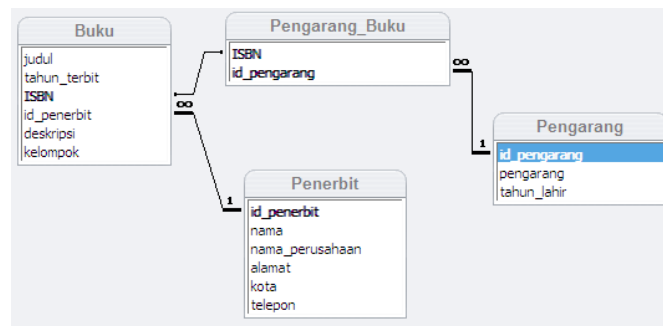
Buku : Table					
judul	tahun terbit	ISBN	id_penerbit	deskripsi	kelompok
Database management: developing application systems using	1989	0-0131985-2-1	17	xx, 441 p. : ill. ;	
Select-- SQL : the relational database language	1992	0-0238669-4-2	12	xv, 446 p. : 24 c	
dBase IV programming	1994	0-0280042-4-8	73		
Step-by-step dBase IV	1995	0-0280095-2-5	52		
Guide to ORACLE	1990	0-0702063-1-7	13	xii, 354 p. : ill. ;	ORACLE (Computer syste
The database experts' guide to SQL	1988	0-0703900-6-1	10		
Oracle/SQL : a professional programmer's guide	1992	0-0704077-5-4	13	xx, 543 p. : ill. ;	
SQL 400: A Professional Programmer's Guide	1994	0-0704079-9-1	52		
Database system concepts	1986	0-0704475-2-7	13		
Microsoft FoxPro 2.5 applications programming	1993	0-0705015-3-X	61	xi	
First look at-- dBASE IV, version 1.5/2.0 for DOS	1994	0-0705107-5-X	80	ill	
Applying SQL in Business	1992	0-0705184-2-4	13		
Database design	1977	0-0707013-0-X	13		
Introduction to Oracle	1989	0-0770716-4-6	13	xi	

Pengarang : Table		
id_pengarang	pengarang	tahun_lahir
1	Adams, Pat	
2	Adnan, Merv	
3	Ageloff, Roy	1943
4	Andersen, Virginia	
5	Antonovich, Michael P	
6	Arnott, Steven E	
7	Amtson, L. Joyce	
8	Ault, Michael R	
9	Avison, D. E	
10	Bard, Dick	1941
11	Biegel, Richard A	
12	Blow, Lisa	
13	Bisland, Ralph B	
14	Bowman, Judith S	

Pengarang_Buku : Table	
ISBN	id_pengarang
0-0131985-2-1	13
0-0238669-4-2	113
0-0280042-4-8	11
0-0280042-4-8	120
0-0280095-2-5	171
0-0702063-1-7	26
0-0702063-1-7	65
0-0702063-1-7	104
0-0703900-6-1	96
0-0704077-5-4	59
0-0704077-5-4	99
0-0704079-9-1	59

Buku yang dikarang lebih dari satu pengarang

Gambar 16.17. Hubungan table Pengaran dan Buku.



Gambar 16.18. Relasi antar table.

16.4 NORMALISASI

Normalisasi :

Proses menghilangkan data yang berulang (redundant) dari table-table basis data relasional dengan cara memecah table menjadi lebih kecil.

Teori normalisasi didasarkan pada apa yang disebut sebagai bentuk normal sebuah table. Sebuah table disebut sebagai normal jika memenuhi kriteria-kriteria tertentu yang menyatakan derajat table terhadap kemungkinan ketidakkonsistenan logika dan anomaly yang muncul. Dalam membuat sebuah basis data yang baik biasanya disyaratkan minimal table sudah dalam keadaan normal bentuk ke 3 (3NF). Kita akan pelajari tentang bentuk-bentuk normal table pada bagian berikutnya. Namun sebelum itu kita akan pelajari dasar-dasar teori dalam normalisasi yaitu functional dependency, superkey, candidate key dan primary key.

16.4.1 Functional Dependency

Functional Dependency (Ketergantungan Fungsional):

Sebuah kolom X pada table R disebut tergantung secara fungsional pada kolom Y pada table R jika dan hanya jika setiap nilai pada kolom X berhubungan secara tepat dengan satu nilai pada kolom Y atau dituliskan dengan notasi : $R.Y \rightarrow R.X$

Kalau kita balik definisi diatas menjadi satu nilai pada kolom Y pada table R secara fungsional akan menentukan nilai X pada table yang sama. Kolom Y dan X boleh terdiri dari satu kolom saja atau kumpulan dari beberapa kolom. Agak susah mungkin membaca definisi di atas, tetapi dengan melihat contoh berikut kita akan dapat memahami pengertian functional dependency. Perhatikan gambar 16.19 berikut ini.

Y

isbn	pengarang	judul	harga
0-672-31337-3	Juande Ramos	Sejarah Sepakbola	105000
0-672-31637-9	Harry Redknapp	Teori Sepakbola Pra Sejarah	55000
0-672-31667-8	Arsene Wenger	Teori Sepakbola Kuno	75000
0-672-31667-9	Harry Redknapp	Teori Sepakbola Modern	65000

Gambar 16.19. Contoh ketergantungan fungsional.

Gambar 16.19 menunjukkan sebuah table dengan 4 kolom yaitu isbn, pengarang, judul dan harga. Kita misalkan kolom isbn adalah Y dan kolom pengarang adalah X. Dengan menggunakan definisi diatas kita dapat mengatakan bahwa kolom pengarang tergantung secara fungsional pada kolom isbn karena nilai-nilai pada kolom pengarang akan berhubungan secara tepat dengan satu nilai pada kolom isbn. Atau dengan bahasa lain, nilai pada isbn akan menentukan nilai yang akan muncul pada kolom pengarang. Pada table di atas kita bisa melihat bahwa jika kolom isbn berisi nilai 0-672-31637-9 maka di kolom pengarang pasti berisi "Harry Redknapp" dan bukan yang lain. Demikian juga dengan kolom yang lain. Misalnya kolom judul, jika isbn berisi nilai 0-672-31667-8 maka pada kolom judul pasti berisi "Teori Sepakbola Kuno".

Dengan melihat contoh di atas kita bisa mengatakan bahwa baik kolom pengarang maupun judul, tergantung secara fungsional pada kolom isbn. Namun tidak sebaliknya. Kolom pengarang misalnya, tidak menentukan secara pasti nilai pada kolom isbn. Sebagai contoh, jika pada kolom pengarang berisi nilai "Harry Redknapp" maka kita tidak bisa menentukan secara pasti nilai pada isbn. Hal ini karena ada dua nilai isbn yang pada kolom pengarang berisi "Harry Redknapp".

16.4.2 Super-Key, Candidate-Key dan Primary-Key

Super-Key merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data dalam sebuah table secara unik. Candidat-Key merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah table secara unik. Jika di dalam sebuah table terdapat lebih dari satu candidate-key, maka salah satu dari candidate-key ini dapat dijadikan sebagai Primary-Key. Untuk lebih mudah memahami pengertian ini kita akan memakai Gambar 16.20 sebagai ilustrasi.

kode_siswa	nama_siswa	alamat_siswa	tgl_lahir_siswa
S090001	Cristiano Ronaldo	Jl. Manchester Raya No.10 Manchester 65144	22-Nov-90
S090002	Ryan Giggs	Jl. Simpang Manchester II/10 Manchester 65123	12-Jul-91
S090003	Wayne Rooney	Jl. Simpang Manchester III/1 Manchester 65132	02-Apr-90
S090004	Cesc Fabregas	Jl. Raya Camp Nou 13 Barcelona 54003	30-Jun-93
S090005	Robbie van Persie	Jl. Simpang Ponderland I/45 Eindhoven 47222	25-Sep-89

Gambar 16.20. Table siswa.

Pada Gambar 16.20, table siswa memiliki 4 buah atribut yaitu kode_siswa, nama_siswa, alamat_siswa dan tgl_lahir_siswa. Tabel siswa ini dalam notasi basis data dapat dinyatakan seperti berikut:

Siswa (kode_siswa, nama_siswa, alamat_siswa, tgl_lahir_siswa)

Dengan menggunakan pengertian super-key, candidate_key dan primary-key maka kita dapat menentukan sebagai berikut.

Tabel 16.1. Super-key, candidate key, dan primary key untuk Table Siswa.

Key	Atribut
Super-key	• (kode_siswa, nama_siswa, alamat_siswa, tgl_lahir_siswa) • (kode_siswa, nama_siswa, alamat_siswa) • (kode_siswa, nama_siswa) • (nama_siswa), jika kita bisa memastikan tidak akan ada nama yang sama atau tidak akan ada duplikasi data pada atribut ini • (kode_siswa)
Candidate-key	• (nama_siswa), jika kita bisa memastikan tidak akan ada nama yang sama atau tidak akan ada duplikasi data pada atribut ini • (kode_siswa)
Primary-key	• (kode_siswa)

Jadi pada table siswa, sebenarnya ada dua candidate-key yaitu nama_siswa dan kode_siswa. Pilihan jatuh pada kode_siswa sebagai primary key, karena kode_siswa lebih konsisten dibanding nama_siswa. Dalam artian, kode_siswa memiliki bentuk yang relative seragam dan panjang karakter yang sama (perhatikan nilai-nilai pada kolom kode_siswa di Gambar 16.20). Nilai-nilai pada kolom nama_siswa (Gambar 16.20) terlihat tidak seragam panjang pendeknya dan sangat rawan kesalahan pengetikan.

16.4.3 Bentuk-Bentuk Normal dan Proses Normalisasi

Ada beberapa bentuk normal yang dikenal dalam basis data yaitu bentuk normal pertama (1NF) sampai dengan bentuk normal ke 5 (5NF), dan bentuk normal Boyce-Codd (BCNF). Sebuah table disebut baik atau normal untuk operasi-operasi basis data biasanya telah memenuhi bentuk normal ke tiga (3NF) atau dalam bentuk normal Boyce-Codd (BCNF). Pada buku ini kita hanya akan membahas sampai dengan bentuk normal ke tiga. Pembahasan BCNF membutuhkan dasar pengetahuan tentang Aljabar Relasional, sehingga tidak akan dibahas.

- **Bentuk normal pertama (1NF)**

Sebuah table disebut memenuhi bentuk normal pertama (1NF) jika syarat-syarat berikut terpenuhi :

- Setiap nilai dalam setiap kolom/field/atribut harus bernilai tunggal (tidak boleh bernilai banyak).
- Semua baris/row/record harus dapat diidentifikasi dengan primary key – nya.
- Semua kolom yang bukan primary key harus tergantung secara fungsional pada primary key, baik langsung atau tidak langsung.

Untuk membuat table dalam bentuk normal pertama, kita akan menggunakan contoh table pada Gambar 16.21. Tabel pada gambar ini sering disebut sebagai Tabel Universal, karena sebenarnya isinya merupakan gabungan dari beberapa table. Table dalam bentuk seperti inilah yang biasanya kita buat untuk mengorganisasi data.

Kode Guru	Nama Guru	Kode PS	PS	Keahlian
I04001	Felipe Scolari	RPL	Rekayasa Perangkat Lunak	Sistem Basis Data, Pemrograman PHP
I04002	Sven Goran Eriksson	RPL	Rekayasa Perangkat Lunak	Pemrograman Visual, Pemrograman Python
I04003	Steve Mc Laren	TKJ	Teknik Komputer & Jaringan	Jaringan Wireless
I04004	Martin O'Neil	RPL	Rekayasa Perangkat Lunak	Algoritma, Pemrograman PHP, Pemrograman Python

Atribut Bernilai Banyak
(Multivalued Attribute)

Gambar 16.21. Table Guru dengan salah satu kolom bernilai banyak.

Langkah awal untuk membentuk table bentuk normal pertama adalah memeriksa apakah ada atribut bernilai banyak. Pada Gambar 16.21, terlihat bahwa kolom keahlian termasuk kategori atribut bernilai banyak. Kita harus merubah table tersebut menjadi seperti Gambar 16.22. Syarat pertama bentuk normal pertama sudah dapat kita penuhi, karena pada Gambar 16.22, tidak kita jumpai lagi atribut yang bernilai banyak.

Kode Guru	Nama Guru	Kode PS	PS	Keahlian
I04001	Felipe Scolari	RPL	Rekayasa Perangkat Lunak	Sistem Basis Data
I04001	Felipe Scolari	RPL	Rekayasa Perangkat Lunak	Pemrograman PHP
I04002	Sven Goran Eriksson	RPL	Rekayasa Perangkat Lunak	Pemrograman Visual
I04002	Sven Goran Eriksson	RPL	Rekayasa Perangkat Lunak	Pemrograman Python
I04003	Steve Mc Laren	TKJ	Teknik Komputer & Jaringan	Jaringan Wireless
I04004	Martin O'Neil	RPL	Rekayasa Perangkat Lunak	Algoritma
I04004	Martin O'Neil	RPL	Rekayasa Perangkat Lunak	Pemrograman PHP
I04004	Martin O'Neil	RPL	Rekayasa Perangkat Lunak	Pemrograman Python

Atribut sudah dalam bentuk bernilai tunggal

Gambar 16.22. Table Guru dengan semua kolom sudah bernilai tunggal.

Langkah berikutnya adalah menentukan primary key dari table. Kita tidak dapat memilih Kode Guru, Nama Guru atau Kode PS sebagai primary key, karena kolom-kolom tersebut memiliki nilai yang tidak unik. Sebagai contoh, pada kolom Kode Guru, nilai I04001 berulang dua kali, bahkan I04004 berulang kali. Demikian juga pada kolom lainnya. Kalau kita telusuri maka sebenarnya primary key pada table Guru adalah gabungan dari Kode Guru, Kode PS, dan Keahlian. Dengan menggunakan primary key ini maka setiap baris dapat kita bedakan dengan baris yang lain. Syarat ke dua telah dapat kita penuhi pula.

Langkah ketiga adalah memeriksa apakah atribut yang tidak termasuk kunci, tergantung secara fungsional pada primary key baik langsung maupun tidak langsung. Atribut yang tidak termasuk primary key adalah Nama Guru dan PS. Kita dapat menguji notasi Functional Dependency (FD) berikut ini:

(Kode Guru, Kode PS, Keahlian) → (Nama Guru)
 (Kode Guru, Kode PS, Keahlian) → (PS)

Dengan melihat nilai-nilai pada Gambar 16.22, kita bisa katakan bahwa kedua FD di atas terpenuhi (lihat kembali pengertian FD pada sub bab sebelumnya). Dengan demikian seluruh syarat bentuk normal pertama telah terpenuhi dan kita bisa mengatakan bahwa Tabel Guru berada dalam bentuk normal pertama (1NF).

- **Bentuk normal kedua (2NF)**

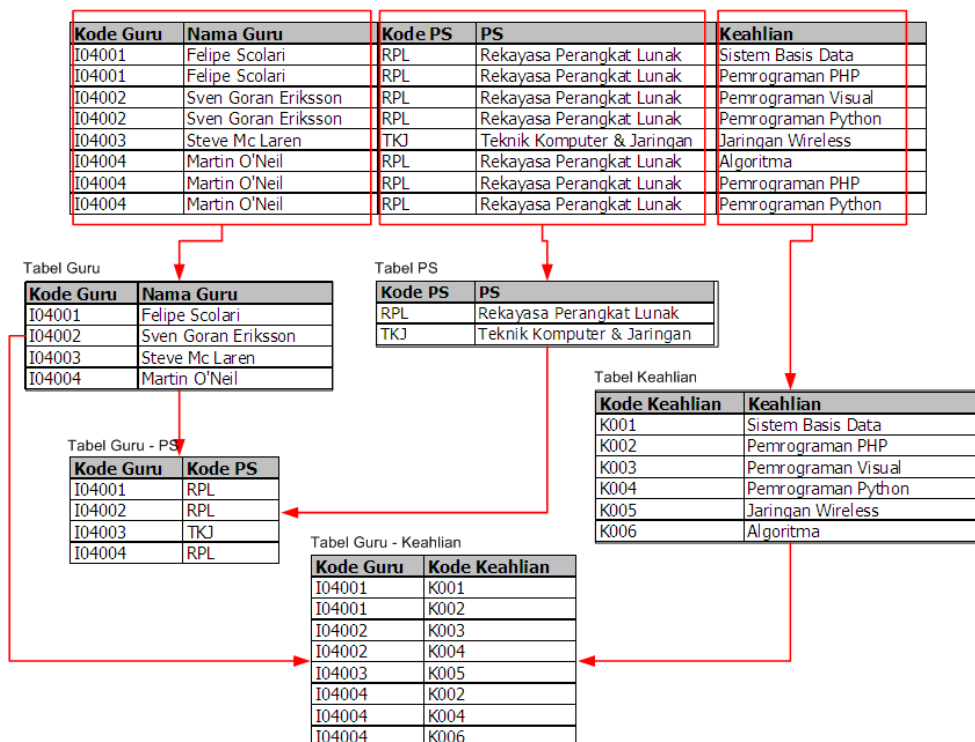
Sebuah table disebut memenuhi bentuk normal kedua (2NF) jika syarat-syarat berikut terpenuhi :

- Tabel sudah berada dalam bentuk normal pertama (1NF).

- Sudah tidak ada ketergantungan fungsional secara parsial, yaitu seluruh kolom yang bukan primary key hanya tergantung pada sebagian dari primary key.

Langkah pertama adalah memeriksa apakah table sudah memenuhi syarat bentuk normal pertama. Pada Gambar 16.22, kita sudah memeriksa dan menyatakan table sudah dalam kondisi normal pertama. Sehingga syarat pertama bentuk normal kedua (2NF) sudah terpenuhi.

Langkah kedua adalah memeriksa apakah ada ketergantungan, jika ada maka table harus diuraikan (dekomposisi). Seperti telah disebutkan, primary key pada Gambar 16.22 adalah (Kode Guru, Kode PS, Keahlian) dan ada dua atribut yang tidak termasuk primary key yaitu Nama Guru dan PS. Kalau diperhatikan lebih seksama, Nama Guru tidak tergantung sepenuhnya pada (Kode Guru, Kode PS, Keahlian) namun hanya tergantung pada Kode Guru. Inilah yang disebut sebagai ketergantungan fungsional secara parsial. Hal yang sama juga terjadi pada atribut PS. Atribut ini hanya tergantung pada Kode PS. Selain itu atribut keahlian sebenarnya juga tergantung secara fungsional hanya pada atribut Kode Guru. Tabel Guru di atas harus didekomposisi dengan memperhatikan ketergantungan fungsional ini. Hasil dekomposisi dapat dilihat pada Gambar 16.23.



Gambar 16.23. Hasil dekomposisi tabel.

Hasil dekomposisi pada Gambar 16.23 menghasilkan lima buah table sebagai berikut:

Tabel 16.2. Tabel-tabel hasil dekomposisi.

Nama Tabel	Atribut (Kolom)	Primary Key	Functional Dependency
Guru	o Kode Guru, o Nama Guru	(Kode Guru)	(Kode Guru)→(Nama Guru)
PS	o Kode PS, o PS	(Kode PS)	(Kode PS)→(PS)
Keahlian	o Kode Keahlian, o Keahlian	(Kode Keahlian)	(Kode Keahlian)→(Keahlian)
Guru-PS	o Kode Guru, o Kode PS	(Kode Guru, Kode PS)	(Kode Guru, Kode PS)→(Kode Guru, Kode PS)
Guru-Keahlian	o Kode Guru, o Kode Keahlian	(Kode Guru, Kode Keahlian)	(Kode Guru, Kode Keahlian)→(Kode Guru, Kode Keahlian)

Tabel-tabel hasil dekomposisi seperti terlihat pada Tabel 16.2 sudah tidak mengandung ketergantungan fungsional parsial. Sehingga kita bisa menyatakan bahwa semua table sudah dalam bentuk normal kedua (2NF).

- **Bentuk normal ketiga (3NF)**

Sebuah table disebut memenuhi bentuk normal ketiga (3NF) jika syarat-syarat berikut terpenuhi :

- o Tabel sudah berada dalam bentuk normal kedua (2NF).
- o Tidak ada ketergantungan transitif (dimana atribut yang bukan primary key tergantung pada atribut lain yang juga bukan primary key).

Seperti telah disebutkan, semua table pada Tabel 16.2, telah berada dalam bentuk normal kedua (2NF), sehingga syarat pertama telah dipenuhi. Pada semua table juga tidak dijumpai adanya ketergantungan transitif. Dengan dipenuhinya kedua syarat ini, kita dapat menyatakan semua table dalam bentuk normal ketiga (3NF).

16.5 QUERY

Mudahnya informasi yang didapat dari basis data sering menentukan nilai basis data di mata pengguna. Salah satu keunggulan model data relasional adalah memungkinkan banyak bentuk permintaan data / informasi diajukan dengan mudah. Fitur ini memberikan kontribusi yang luar biasa terhadap popularitas model data relasional. Sebagai contoh, pada basis data

akademik sebuah sekolah kita dapat mengajukan pertanyaan atau permintaan data seperti berikut :

- Siapa nama siswa dengan nomor induk 0200010002?
- Berapa rata-rata siswa yang mengambil program keahlian dengan kode RPL per tahunnya?
- Apakah ada siswa yang rata-rata nilai semesternya nya kurang dari 7.0 yang mengambil program keahlian dengan kode RPL?
- Apakah ada guru yang mengajar kurang dari 6 jam seminggu?
- Berapa siswa yang memperoleh nilai kurang dari 70 pada matapelajaran dengan kode XXXX79?
- Dan seterusnya?

Pertanyaan-pertanyaan di atas akan melibatkan data yang disimpan dalam basis data.

Query :

permintaan / pertanyaan terhadap basis data pada data / informasi tertentu dengan menggunakan struktur dan standar tertentu.

DBMS menyediakan bahasa khusus yang disebut bahasa query (Query Language), dimana query dapat diajukan. DBMS memungkinkan pengguna untuk membuat, memodifikasi, dan meng-query data melalui Data Manipulation Language (DML). Jadi, bahasa query adalah satu-satunya bagian DML yang juga menyediakan konstruksi untuk menyisipkan, menghapus, dan memodifikasi data.

Ada dua pendekatan dasar dalam query yaitu Query by Example (QBE) dan SQL. QBE adalah pendekatan fill-in-the-form untuk membuat query. Kita memilih table dan kolom dari daftar yang tersedia dan mengisi bagian untuk kondisi kemudian mengeksekusi perintah tersebut. QBE relative lebih mudah digunakan, hanya membutuhkan kemampuan minimal untuk mengetik, dan ditujukan untuk para pemula. Microsoft Access merupakan salah satu DBMS yang memberikan fasilitas QBE ini. Bab 17 akan memberikan ilustrasi bagaimana menggunakan QBE dalam Microsoft Access.

SQL merupakan kebalikan dari QBE, karena SQL merupakan bahasa yang harus ditulis dengan sintaks tertentu. Meskipun lebih sulit, SQL lebih fleksibel dan mudah dicangkokkan dalam aplikasi-aplikasi berbasis computer yang membutuhkan basis data. SQL lebih dibanyak digunakan karena dukungan dari vendor DBMS yang besar serta adanya standar yang telah

ditetapkan bersama. Pada Bab 18 akan dibahas lebih mendalam tentang SQL.

Ada beberapa pertanyaan dasar yang dapat memandu kita dalam merancang sebuah query. Post (1999) menyebutkan sebagai berikut :

- *Output apa yang ingin kita lihat?*

Pertanyaan ini berhubungan pemilihan kolom mana dari berbagai macam table yang akan kita tampilkan. Kita harus tahu pasti nama-nama kolom tersebut. Pada QBE hal ini mudah dilakukan, karena daftar kolom dari semua table sudah tersedia otomatis. Problem utama yang sering muncul adalah menentukan kolom mana yang benar-benar kita butuhkan, apalagi bila dalam basis data terdapat puluhan atau bahkan ratusan table.

- *Apa yang sudah kita ketahui?*

Pada beberapa situasi kita ingin membatasi pencarian data berdasarkan kriteria-kriteria tertentu. Criteria atau kondisi ini harus sudah kita ketahui dengan pasti.

- *Tabel mana yang akan terlibat?*

Apabila dalam basis data hanya ada sedikit table, maka kita dengan mudah menentukan table mana yang terlibat dalam query. Namun bila jumlah table sangat banyak, butuh waktu yang tidak sedikit untuk menentukan table mana yang terlibat. Pembuatan kamus data akan sangat membantu menjawab pertanyaan ini.

- *Bagaimana table akan digabungkan (join)?*

Pertanyaan ini berhubungan dengan normalisasi data (lihat kembali bab 17). Table-table dalam basis data saling berhubungan dengan menggunakan aturan-aturan relasi tertentu. Oleh karena itu apabila kita ingin melakukan penggabungan (join) antar table, aturan-aturan tersebut harus dipahami dan diperhatikan.

Mengelola Basis Data Dengan DBMS

TUJUAN

Setelah anda mempelajari bab ini diharapkan anda akan mampu :

- Menjalankan perangkat lunak DBMS (Microsoft Access)
 - Membuat basis data, table, dan atribut dengan DBMS (Microsoft Access)
 - Membuat query dengan DBMS (Microsoft Access)
 - Membuat form dengan DBMS (Microsoft Access)
 - Membuat report dengan DBMS (Microsoft Access)
-

Pada bagian berikut ini dan seterusnya kita akan mencoba menerapkan apa yang sudah kita pelajari pada bab-bab sebelumnya dengan menggunakan DBMS berbasis GUI, yaitu Microsoft Access. Microsoft Access adalah suatu aplikasi yang dapat membantu kita membuat sebuah aplikasi database dalam waktu yang relatif singkat. Biasanya digunakan untuk pembuatan aplikasi-aplikasi yang kecil. Penjelasan lebih detil tentang Microsoft Access dapat dilihat pada Bab 14.

Sebelum kita memulai dengan pembahasan tentang operasi bagian-bagian Microsoft Access, kita akan sekilas membahas tentang contoh kasus basis data yang akan kita buat. Kasus yang akan kita buat adalah *Basis Data Penjualan Buku*. Pada kasus ini setiap pembeli akan melakukan pembelian terhadap buku yang diinginkan dan membayar sejumlah uang sesuai buku yang dibeli. Data pembeli akan dicatat. Demikian juga setiap data pesanan, baik itu pemesanan ringkasan maupun item-item pemesanannya. Data pesanan berisi pembeli yang melakukan pemesanan, total pembelian dan tanggal pembelian. Sedangkan data item pemesanan berisi data buku yang dipesan dan jumlahnya untuk tiap pemesanan.

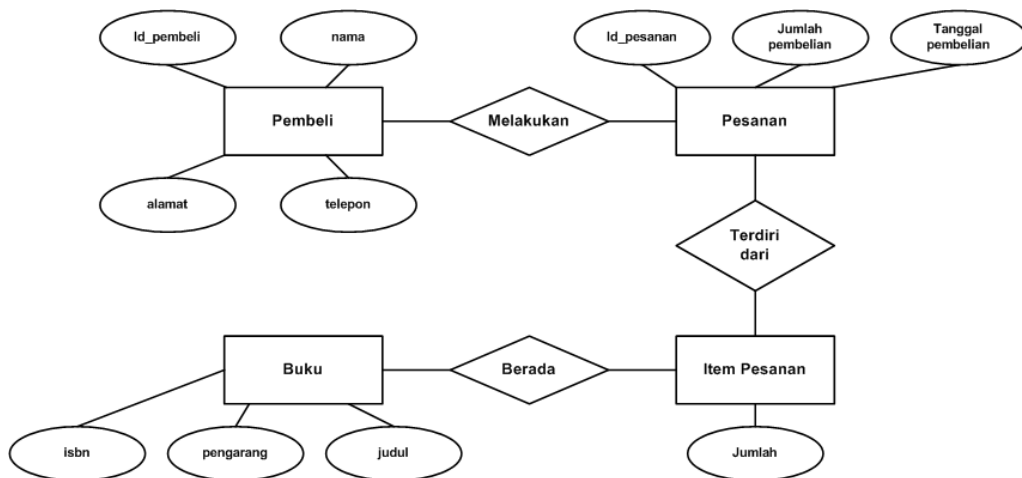
Dengan membaca kasus di atas, maka apabila kita akan membuat ER Diagramnya maka langkah pertama adalah identifikasi kandidat entity yang terlibat. Dari teks di atas kita dapat mengidentifikasi ada minimal 4 kandidat entity yaitu *pembeli*, *buku*, *pesanan* dan *item pemesanan*. Sedangkan relasinya dapat kita identifikasi sebagai berikut :

- pembeli melakukan pemesanan
- pada setiap pesanan terdapat item-item pesanan
- pada item-item pesanan terdapat daftar buku-buku yang dipesan.

Atribut-atribut untuk masing-masing kandidat entity dapat kita tentukan sebagai berikut:

- *Entity Pembeli* dengan atribut *id_pembeli*, *nama*, *alamat*, dan *telepon*.
- *Entity Buku* dengan atribut nomor *ISBN*, *pengarang*, dan *judul*
- *Entity Pesanan* dengan atribut *id_pesanan*, *jumlah pembelian*, *tanggal pembelian*.
- *Entity Item Pemesanan* dengan atribut jumlah masing-masing buku yang dipesan.

Setelah semua informasi lengkap maka kita dapat menggambarkan ER Diagram untuk kasus di atas seperti pada Gambar 17.1.



Gambar 17.1. ER Diagram untuk kasus Basis Data Penjualan Buku.

Berdasarkan ER Diagram di atas kita dapat menentukan table-table apa yang dibutuhkan pada basis data penjualan buku. Ada empat table yaitu: table *pembeli*, *buku*, *pesanan*, dan *item pesanan*. Untuk mengakomodasi relasi yang ada pada ER Diagram, maka akan dibuat atribut-atribut yang berperan sebagai perwujudan relasi. Tabel 17.1 menunjukkan table dan atribut yang ada pada *basis data penjualan buku*.

Tabel 17.1. Tabel dan atribut pada Basis Data Penjualan Buku.

Tabel	Atribut	Keterangan
Pembeli	- <u>id_pembeli</u> - nama - alamat - telepon	Id_pesanan adalah atribut kunci (primary key) karena bersifat unik.
Buku	- <u>isbn</u> - pengarang - judul	Isbn adalah primary key karena bersifat unik
Pesanan	- <u>id_pesanan</u> - id_pembeli - jumlah_pembelian - tanggal_pembelian	Id_pesanan adalah primary key karena bersifat unik. Sedangkan id_pembeli adalah atribut penghubung (foreign key) entity Pembeli dengan Pesanan. Hal ini merupakan perwujudan dari relasi Pembeli melakukan Pesanan.
Item_Pesanan	- <u>id_pesanan</u> - <u>isbn</u> - jumlah	Id_pesanan dan isbn secara bersama-sama adalah primary key untuk table ini. Id_pesanan sendiri adalah foreign key sebagai perwujudan relasi Pesanan terdiri dari Item_Pesanan. Sedangkan isbn adalah perwujudan relasi Buku berada dalam Item_Pesanan.

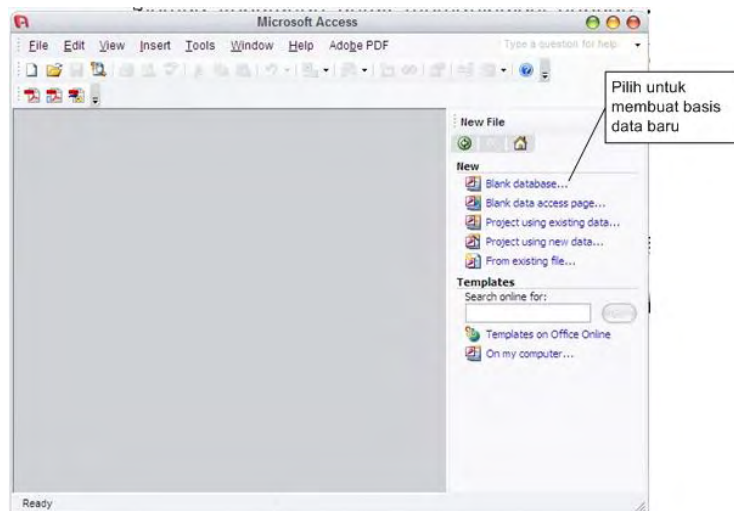
Tabel 17.1 dapat kita lengkapi dengan tipe data dan constraint/domain seperti pada Tabel 17.2. Hal ini untuk mempermudah pembuatan table pada DBMS. Apabila kita memeriksa apakah table-table yang terbentuk sudah dalam bentuk normal atau belum, maka kita akan menjumpai semua table sudah dalam bentuk normal bentuk ketiga (3NF).

Tabel 17.2. Tabel, atribut, tipe data dan constraint/domain pada Basis Data Penjualan Buku.

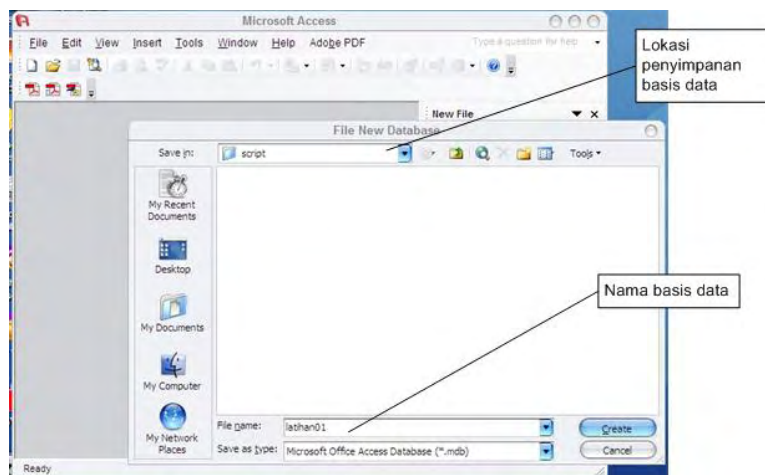
Tabel	Atribut	Tipe Data	Constraint/Domain
Pembeli	- id_pembeli - nama - alamat - telepon	Integer Char/Text (30) Char/Text (60) Char/Text (15)	Not Null
Buku	- isbn - pengarang - judul	Char/Text (15) Char/Text (30) Char/Text (50)	Not Null
Pesanan	- id_pesanan - id_pembeli - jumlah_pembelian - tanggal_pembelian	Integer Integer Real/Float (10,2) Date	Not Null Not Null
Item_Pesanan	- id_pesanan - isbn - jumlah	Integer Char/Text (30) Integer	Not Null Not Null Not Null dan > 0

17.1 MEMULAI MICROSOFT ACCESS

Untuk memulai Microsoft Access, kita dapat melakukan klik Start pada Windows pilih Programs lalu pilih Microsoft Access. Tampilan awal Microsoft Access akan tampak seperti Gambar 17.2. Microsoft Access memberikan beberapa opsi dalam pembuatan basis data seperti tampak pada Gambar 17.2. Namun opsi yang paling sering kita gunakan adalah *Blank database*. Apabila kita klik pada bagian ini maka akan muncul permintaan untuk mendefinisikan basis data seperti terlihat pada Gambar 17.3. Nama dan lokasi penyimpanan basis data harus ditentukan.

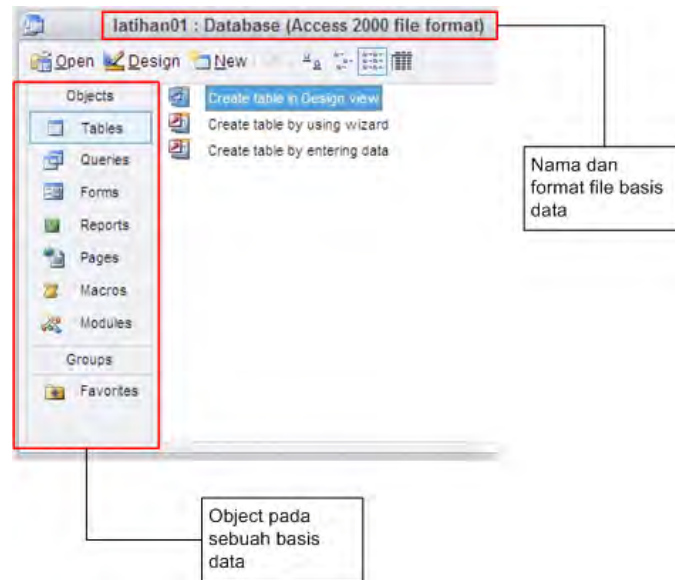


Gambar 17.2. Tampilan awal Microsoft Access.



Gambar 17.3. Penentuan nama dan lokasi basis data.

Setelah kita tekan tombol create berarti kita telah mempunyai sebuah basis data, namun masih belum terisi table atau data apapun (Gambar 17.4). Pada gambar tersebut dapat kita lihat nama file basis data adalah *latihan01* dan format file basis data menggunakan Access 2000. Pada gambar tersebut juga tampak bagian-bagian (object) basis data tersebut. Dari object-object yang ada, ada 4 object yang sangat penting dan selalu akan digunakan yaitu *Tables*, *Queries*, *Forms* dan *Reports*. Sedangkan object-object lain yaitu *Pages*, *Macros* dan *Modules* merupakan bagian yang bersifat opsional.



Gambar 17.4. Bagian-bagian sebuah basis data pada Microsoft Access.

17.2 TABLE

Ketika kita selesai membuat file basis data (lihat Gambar 17.4), kita sudah punya sebuah basis data namun belum berisi apa-apa karena kita belum membuat table-table dalam basis data tersebut. Seperti telah dijelaskan pada Bab 16, table akan berisi kolom dan baris. Kolom disebut *field* dan baris disebut *record* dalam Microsoft Access.

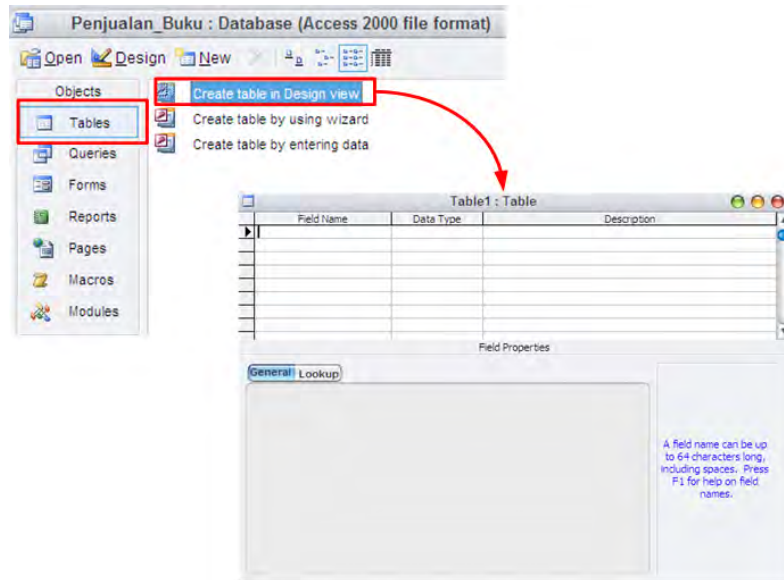
17.2.1 Pendefinisian Field dan Tipe Data.

Tahap pertama dalam membuat table adalah mendefinisikan field-field yang dibutuhkan baru kemudian mengisi baris-baris data.

Langkah-langkah pembuatan table adalah sebagai berikut:

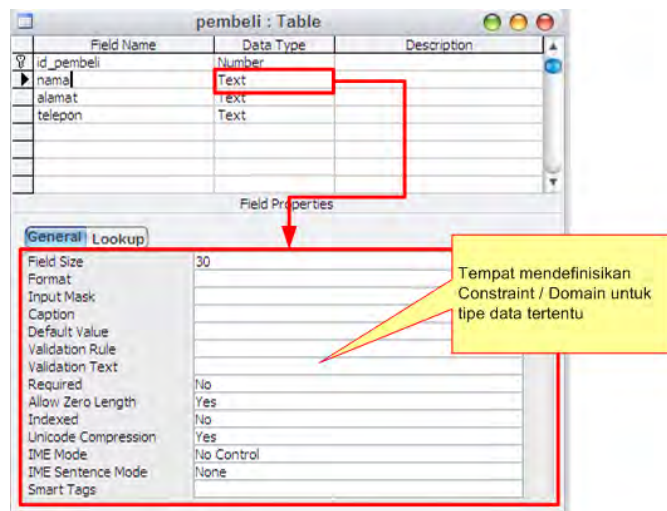
1. Pada jendela *Database* click pada *Table* (Gambar 17.5).

2. Double click pada Create table in Design View (Gambar 17.5). Kemudian akan muncul jendela untuk mendefinisikan field-field yang dibutuhkan (Gambar 17.5)



Gambar 17.5. Tahap awal pembuatan table.

3. Kita dapat mulai memasukkan field-field yang dibutuhkan. Untuk contoh awal kita akan memasukkan field-field untuk table *Pembeli* seperti yang sudah didefinisikan pada Tabel 17.2. Perhatikan Gambar 17.6 berikut ini.



Gambar 17.6. Pendefinisian field, tipe data, constraint dan domain.

Microsoft Access menyediakan fasilitas yang sangat baik untuk mendefinisikan field-field suatu table. Pada Gambar 17.6 tampak proses pendefinisian field. Bagian atas adalah untuk menentukan nama field, tipe data dan keterangan. Sedangkan bagian bawah merupakan tempat menentukan lebar data, format, domain atau constraint dari suatu tipe data. Pada gambar tersebut terlihat bahwa field *nama* didefinisikan bertipe data teks, dengan lebar data 30 (lihat bagian bawah gambar).

4. Setelah semua field untuk table *pembeli* selesai didefinisikan maka kita harus menentukan field mana yang berperan sebagai primary key. Pada penjelasan kasus di atas kita sudah menetapkan bahwa *id_pembeli* akan menjadi primary key. Pilih / sorot baris *id_pembeli*, kemudian click tombol bergambar kunci (Gambar 17.7) pada toolbar Microsoft Access.



Gambar 17.7. Toolbar Microsoft Access.

5. Kita dapat menyimpan table yang sudah kita definisikan dan memberi nama table tersebut dengan cara menekan tombol bergambar disket (lihat Gambar 17.7). Kemudian kita dapat menutup jendela Design View tersebut. Dengan cara yang sama, table-table lainnya yaitu table *buku*, table *pesanan*, dan table *item_pesanan* dapat kita definisikan. Gambar-gambar berikut menunjukkan hasil pendefinisian keseluruhan table.

pembeli : Table			
	Field Name	Data Type	Description
	id_pembeli	Number	No Id pembeli
	nama	Text	Nama pembeli buku
	alamat	Text	Alamat pembeli
	telepon	Text	No telepon pembeli

Gambar 17.8. Struktur table *pembeli*.

buku : Table			
	Field Name	Data Type	Description
	isbn	Text	Nomor ISBN buku
	pengarang	Text	Nama pengarang buku
	judul	Text	Judul buku
	harga	Number	Harga per eksemplar buku

Gambar 17.9. Struktur table *buku*.

pesanan : Table			
	Field Name	Data Type	Description
?	id_pesanan	Number	No Id pesanan
	id_pembeli	Number	No Id pembeli
	jumlah_pembelian	Number	Nilai total pembelian per pesanan
	tanggal_pembelian	Date/Time	Tanggal pembelian di lakukan

Gambar 17.10. Struktur table *pesanan*.

item_pesanan : Table			
	Field Name	Data Type	Description
?	id_pesanan	Number	No id pesanan
?	isbn	Text	No ISBN buku
	jumlah	Number	Jumlah eksemplar buku yang dibeli

Gambar 17.11. Struktur table *item_pesanan*.

17.2.2 Pengisian Data Pada Tabel.

Table-table yang telah kita buat di atas, belum mempunyai isi data apa-apa. Hanya strukturnya yang telah kita buat. Untuk mengisi data pada table, caranya cukup mudah, yaitu: *double click* pada nama table yang kita ingin isikan datanya. Setelah terbuka jendela seperti pada Gambar 17.12, kita dapat segera mengisi datanya. Cara pengisian datanya sama seperti kalau kita bekerja dengan Microsoft Excell atau software spreadsheet lainnya. Kemudian kita dapat menyimpan hasil pengisian data dengan menekan tombol bergambar disket pada toolbar.

pembeli : Table				
	id_pembeli	nama	alamat	telepon
▶ +	1	Cristiano Ronaldo	Jl. Manchester Raya No.10 Manchester	0808000000
+	2	Ryan Giggs	Jl. Simpang Manchester II/10 Manches	0801000000
+	3	Wayne Rooney	Jl. Simpang Manchester III/10 Manche	0802000000
*	0			

Gambar 17.12. Hasil pengisian data pada table *pembeli*.

Gambar 17.12 menunjukkan isi data pada table *pembeli*. Dengan cara yang sama kita dapat mengisi data pada table-table lain. Perhatikan hasil pengisian data pada gambar-gambar berikut.

	isbn	pengarang	judul	harga
▶ +	0-672-31337-4	Juande Ramos	Sejarah Sepakbola	105000
+	0-672-31637-9	Harry Redknapp	Teori Sepakbola Pra Sejarah	55000
+	0-672-31667-9	Arsene Wenger	Teori Sepakbola Kuno	75000
+	0-672-31667-9	Harry Redknapp	Teori Sepakbola Modern	65000
*				0

Gambar 17.13. Hasil pengisian data pada table *buku*.

	id_pesanan	id_pembeli	jumlah_pembelian	tanggal_pembelian
▶ +	1	2	65000	12-Apr-07
+	2	1	75000	22-Jul-07
+	3	3	160000	25-Jul-07
+	4	2	55000	10-Aug-07
+	5	1	120000	15-Sep-07
*	0	0	0	

Gambar 17.14. Hasil pengisian data pada table *pesanan*.

	id_pesanan	isbn	jumlah
▶	1	0-672-31667-9	1
	2	0-672-31667-8	1
	3	0-672-31337-3	1
	3	0-672-31637-9	1
	4	0-672-31637-9	1
	5	0-672-31637-9	1
	5	0-672-31667-9	1
*	0		0

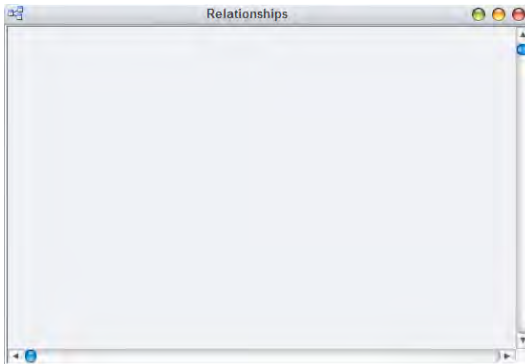
Gambar 17.15. Hasil pengisian data pada table *item_pesanan*.

17.3 QUERY

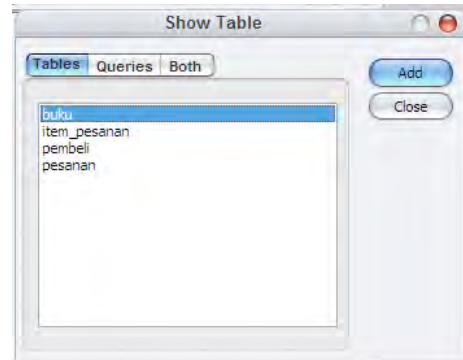
Pada bagian ini kita akan menerapkan teori-teori query yang telah kita pelajari pada Bab 16. Seperti telah dijelaskan, query adalah 'permintaan data'. Dengan query kita dapat menampilkan data-data tertentu dari satu atau lebih table, atau melakukan perhitungan pada data di dalam table. Namun sebelum mempelajari bagaimana membuat query, kita akan pelajari dulu bagaimana membuat relasi antar table agar ketika membuat query menjadi lebih mudah.

17.3.1 Membuat Relasi Antar Tabel.

Pada Gambar 17.1, kita telah mendefinisikan hubungan antar entity dalam ER Diagram. Gambar ini merupakan dasar dalam membuat relasi antar table. Untuk membuat relasi antar table, pilih menu Tools kemudian Relationship sehingga akan muncul jendela seperti pada Gambar 17.16. Pada jendela relationships tersebut klik kanan sehingga muncul menu pilihan dan pilih Show Tabel. Setelah muncul jendela seperti Gambar 17.17, kita dapat mulai menentukan table mana saja yang akan kita relasikan.



Gambar 17.16. Jendela *Relationships*.



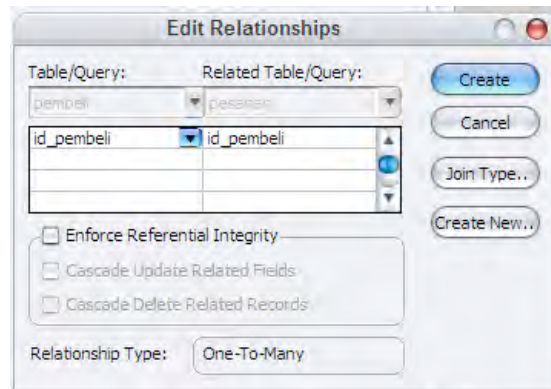
Gambar 17.17. Jendela *Show Table*.

Pada Gambar 17.17, pilih table yang akan direlasikan kemudian *click* tombol **Add**. Setelah selesai proses pemilihan *click* tombol **Close**. Pada kasus yang akan kita terapkan ini keempat table itu berhubungan langsung maupun tidak langsung. Sehingga kita memilih seluruh table untuk direlasikan. Gambar 17.18 menunjukkan hasil pemilihan table pada jendela *Relationships*. Pada gambar ini yang diperlihatkan adalah struktur dari masing-masing table, bukan isinya. Nama kolom yang dicetak tebal menunjukkan kolom tersebut adalah primary key. Kita dapat mengatur posisi table dengan cara *drag & drop*.

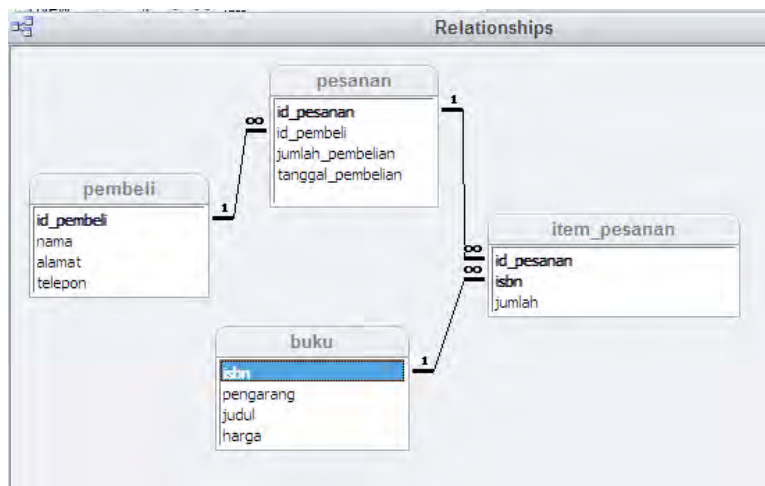


Gambar 17.18. Tabel-tabel yang akan direlasikan.

Untuk membuat relasi antar table, dapat dilakukan dengan memilih primary key pada suatu table kemudian seret mouse menuju key dengan nama yang sama pada table lainnya (foreign key pada table lain). Sebagai contoh pada table *pembeli*, primary key – nya *id_pembeli* dan pada table *pesanan*, *id_pembeli* adalah *foreign key*. Click *id_pembeli* pada table *pembeli* kemudian seret mouse menuju *id_pembeli* pada table *pesanan*. Apabila prosedur ini benar dilakukan, maka akan muncul jendela seperti Gambar 17.19. Click pada bagian **Enforce Referential Integrity** dan click tombol **Create**. Kita dapat melakukan prosedur ini pada relasi-relasi yang lain. Sehingga pada jendela *Relationships* akan tampak seperti pada Gambar 17.20.



Gambar 17.19. Jendela untuk edit relationships



Gambar 17.20. Relasi untuk keseluruhan table.

Gambar 17.20 menunjukkan relasi antar table yang dapat kita bandingkan dengan ER Diagram pada Gambar 17.1. Pada relasi antar table ini, kita juga menentukan kardinalitas antar table. Perhatikan pada garis yang menghubungkan table *pembeli* dengan *pesanan*. Di ujung yang berada pada table *pembeli* ditandai dengan angka 1 dan di ujung yang ada pada table *pesanan* ditandai dengan notasi ∞ . Hal ini menunjukkan adanya hubungan *one-to-many* antara table *pembeli* dengan table *pesanan*. Demikian juga dengan relasi antar table yang lain.

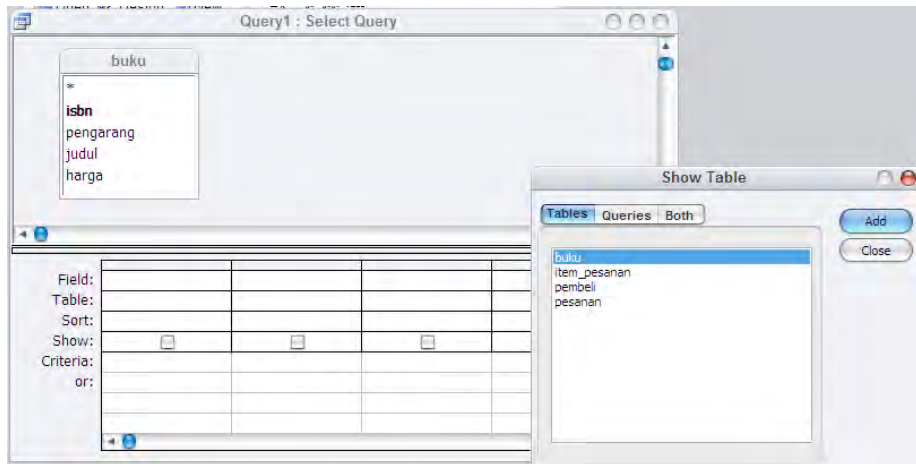
17.3.2 Membuat Query

Microsoft Access menyediakan fasilitas query yang sangat baik dan memudahkan pengguna. Selain karena berbasis GUI juga prosedurnya tidak terlalu rumit. Berikut contoh-contoh membuat query di Microsoft Access.

- Query pada satu table

Query pada satu table hanya akan melibatkan satu table saja. Misalkan kita akan melakukan query pada table *buku*. Prosedur yang ditempuh adalah sebagai berikut:

1. Pada jendela *Database* pilih *Query* lalu klik dua kali *Create Query In Design View*. Sehingga akan muncul jendela seperti pada Gambar 17.21. Pilih table *buku* pada jendela *Show Table*, kemudian click **Add** dan kemudian **Close**. Jendela *Show Table* akan tertutup dan kita dapat mulai melakukan query.



Gambar 17.21. Jendela query pada mode design view.

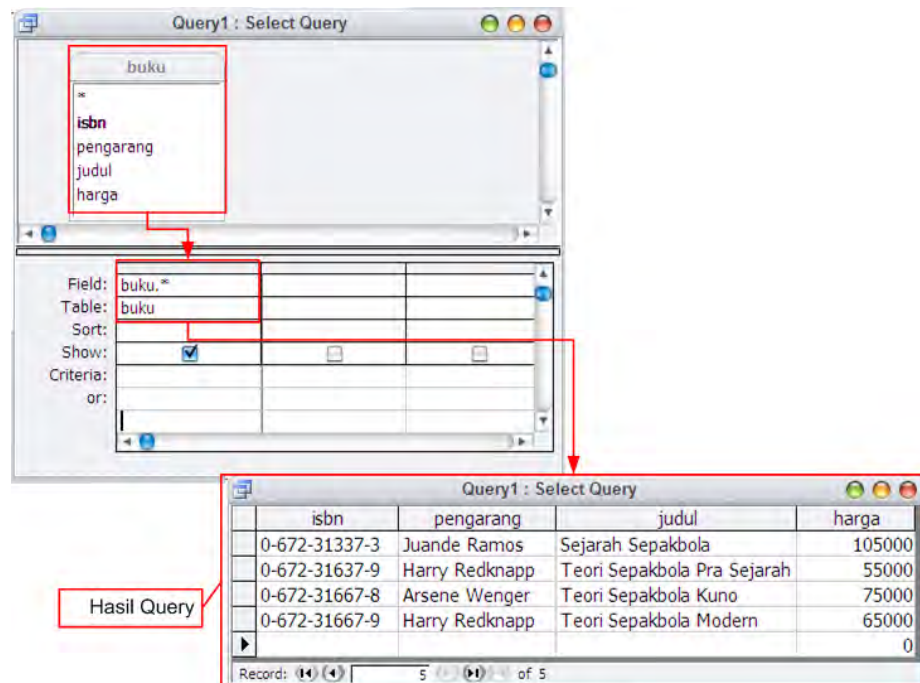
2. Pada jendela *Query* bagian bawah (lihat Gambar 17.21), ada beberapa hal penting yang harus diketahui dan berguna dalam query yaitu

Field : Nama Field yang ingin ditampilkan
Table : Nama Table dari Filed tersebut
Sort : Mengurutkan Data hasil query
Show : Mengatur Field ditampilkan atau tidak
Criteria : Syarat dari data yang ingin ditampilkan

3. Contoh query yang pertama adalah bagaimana menampilkan semua data, misalnya:

Tampilkan semua data yang ada di table buku.

Untuk menampilkan seluruh data pada table *buku*, pada Field, click tombol panah ke bawah pilih **buku.*** (lihat Gambar 17.22). Kita dapat mengeksekusi query dengan memilih menu Query kemudian click Run, atau click langsung tombol tanda seru (!) yang ada di toolbar. Hasil query dapat dilihat pada Gambar 17.22. Simpan query dengan nama yang diinginkan (misalnya: *query_buku_semua_data*) kemudian tutup jendela *Query*.

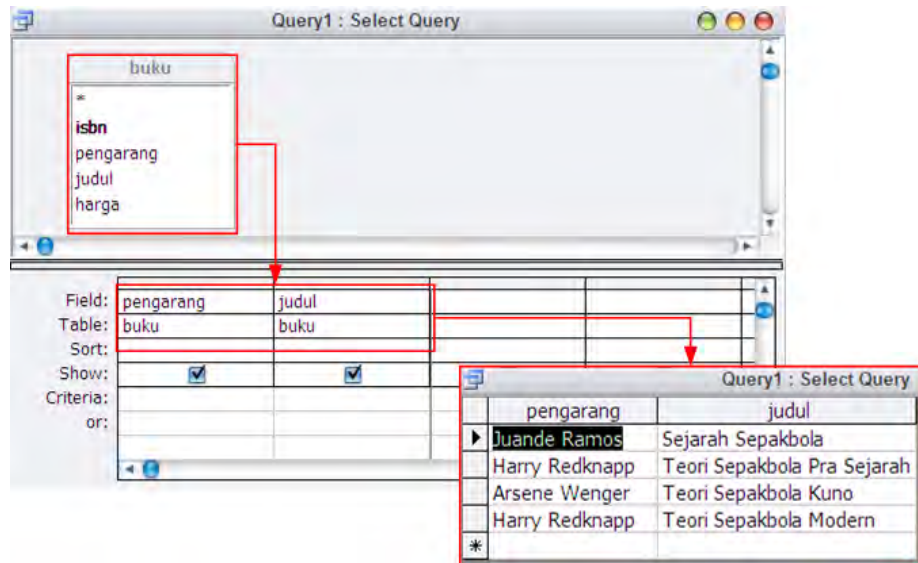


Gambar 17.22. Prosedur dan hasil query table *buku*.

4. Contoh query yang kedua adalah memilih kolom mana saja yang akan ditampilkan, misalnya :

Tampilkan semua nama pengarang dan judul buku yang dikarangnya

Query ini tidak menampilkan seluruh data tetapi hanya data dari kolom *pengarang* dan *judul* buku saja. Pada Gambar 17.23 terlihat bagaimana query dilakukan. Table buku tetap dipilih dari jendela *Show Table*, kemudian di jendela *Query*, pada bagian *Field* dipilih field *pengarang* dan *judul*. Hasil eksekusi query adalah daftar nama seluruh pengarang dan buku yang dikarangnya.

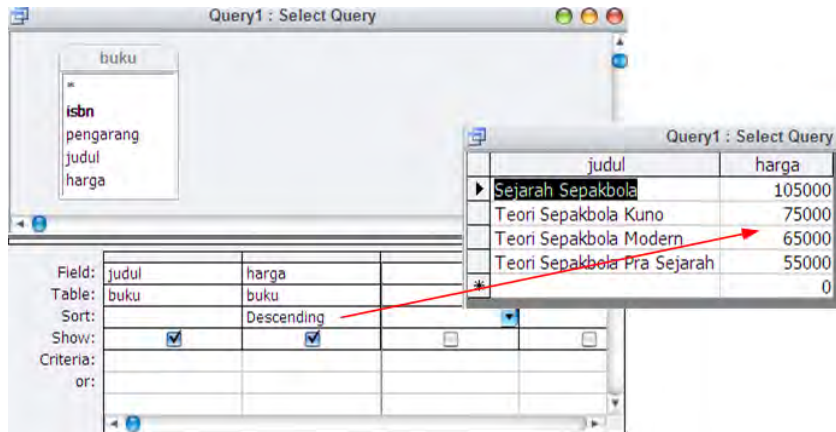


Gambar 17.23. Query nama pengarang dan bukunya.

5. Contoh query yang ketiga adalah bagaimana membuat tampilan dataurut sesuai yang dikehendaki, misalnya:

Tampilkan semua judul buku dan harganya dengan urutan harga yang paling mahal lebih dahulu

Query ini juga tidak menampilkan seluruh data tetapi hanya data dari *judul buku* dan *harga* saja. Namun urutan tampilan dirubah. Pada Gambar 17.24 terlihat bagaimana query dilakukan. Table *buku* tetap dipilih dari jendela *Show Table*, kemudian di jendela *Query*, pada bagian *Field* dipilih field *judul* dan *harga*. Pada bagian **Sort**, pada kolom yang sama dengan harga, kita gunakan opsi **Descending** untuk mengurutkan dari besar ke kecil. Hasil eksekusi query adalah daftar seluruh judul buku dan harganya dengan urutan judul buku yang berharga paling mahal di atas.. Bandingkan urutan baris pada hasil dengan hasil query pada Gambar 17.22.

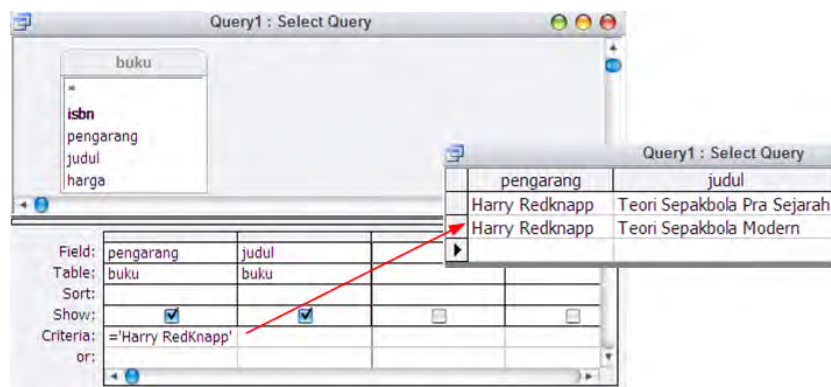


Gambar 17.24. Query judul buku dan harga dengan urutan.

6. Contoh query yang ketiga adalah bagaimana memilih baris-baris mana saja yang akan ditampilkan, misalnya:

Tampilkan semua judul buku yang pengarangnya adalah Harry Redknapp.

Query ini juga hanya berhubungan dengan data dari field *judul buku* dan pengarang saja. Namun tidak seluruh judul, tetapi hanya judul buku yang ditulis 'Harry Redknapp'. Pada Gambar 17.25 terlihat bagaimana query dilakukan. Table *buku* tetap dipilih dari jendela *Show Table*, kemudian di jendela Query, pada bagian Field dipilih field *nama pengarang* dan *judul*. Pada bagian criteria, kita masukkan criteria yang kita maksudkan, yaitu = 'Harry Redknapp'. Hasil eksekusi query adalah daftar seluruh judul buku yang ditulis 'Harry Redknapp'.

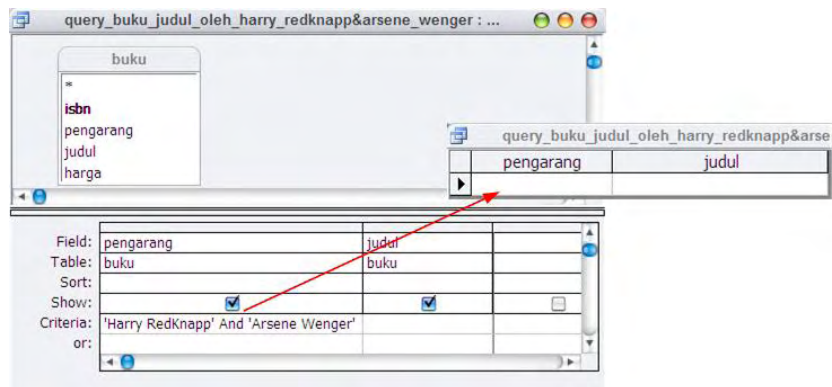


Gambar 17.25. Query dengan criteria tertentu.

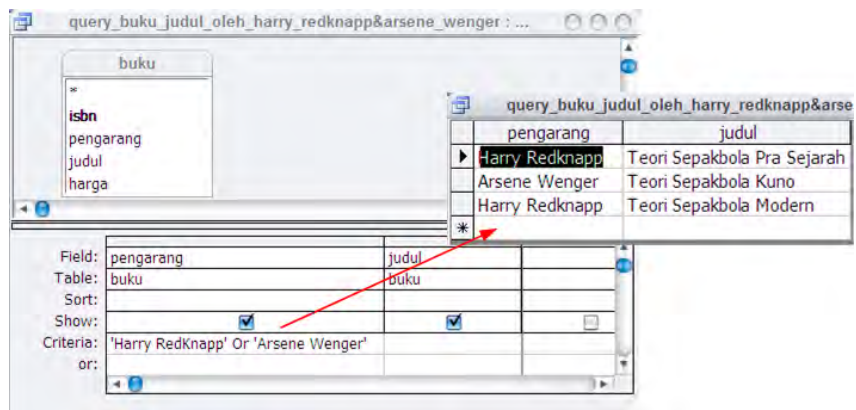
7. Contoh query yang ketiga adalah bagaimana kita menggunakan operator pada kriteria, misalnya:

Tampilkan semua judul buku yang pengarangnya adalah Harry Redknapp dan Arsene Wenger.

Seperti pada no 6, query ini juga hanya berhubungan dengan data dari field judul dan pengarang saja. Namun tidak seluruh judul, tetapi hanya judul buku yang ditulis secara bersama oleh 'Harry Redknapp dan Arsene Wenger'. Pada Gambar 17.26 terlihat bagaimana query dilakukan. Gunakan cara yang sama seperti pada no 6, hanya pada bagian criteria, kita masukkan criteria yang kita maksudkan, yaitu = *'Harry Redknapp' and 'Arsene Wenger'*. Hasil eksekusi menunjukkan tidak ada satu recordpun yang memenuhi. Hal ini karena memang tidak ada buku yang ditulis secara bersama oleh 'Harry Redknapp' dan 'Arsene Wenger'. Bagaimana kalau kita ganti **and** dengan **or**? Perhatikan pada Gambar 17.27. Ada 3 record yang sesuai.



Gambar 17.26. Query dengan menggunakan operator **and**.



Gambar 17.27. Query dengan menggunakan operator **or**.

- Query pada lebih dari satu table

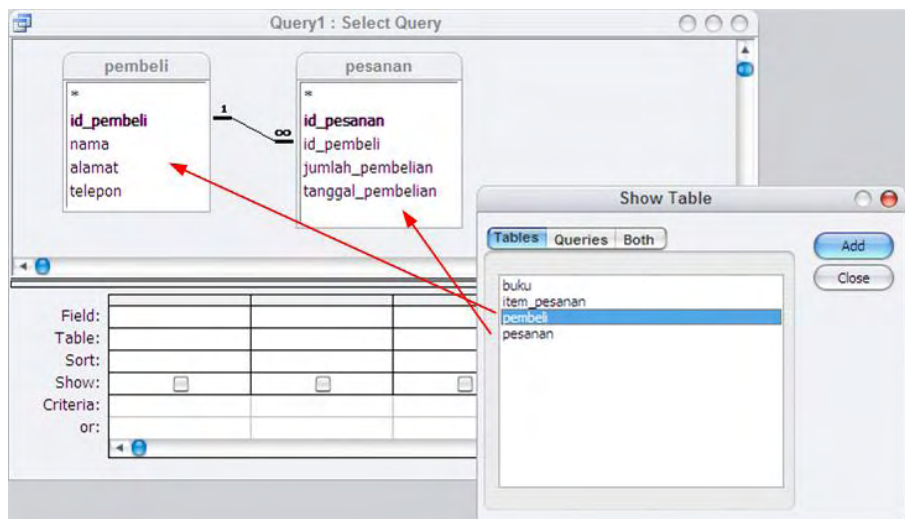
Query pada lebih dari satu table, relasi antar table yang telah kita buat sebelumnya akan menjadi sangat penting. Karena relasi ini akan menentukan bagaimana hasil dari query. Prosedur untuk membuat query sama seperti membuat query satu table, namun table yang dipilih pada jendela Show Tabel tidak lagi satu, tetapi mungkin dua, tiga atau lebih, sesuai dengan kebutuhan. Berikut ini beberapa contoh query dengan lebih dari satu table.

1. Contoh query yang pertama adalah query yang melibatkan dua table, misalnya:

Tampilkan nama dan alamat pembeli yang jumlah pembeliannya lebih dari 100000.

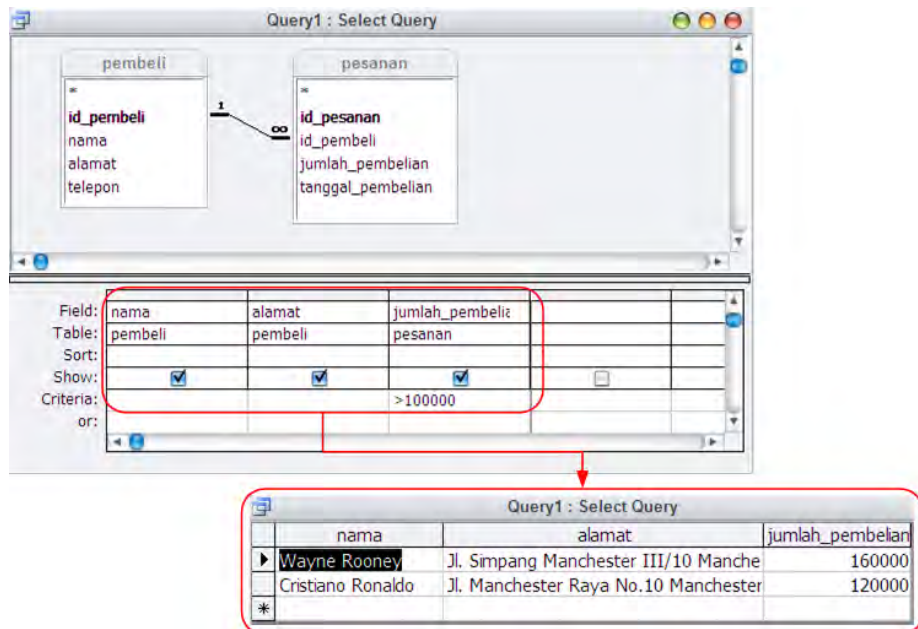
Pada query ini kita membutuhkan table pembeli karena kolom nama dan alamat pembeli ada pada table pembeli. Kita juga membutuhkan table pesanan karena kolom jumlah pembelian ada pada table ini. Dari relasi antar table pada Gambar 17.20, kita tahu bahwa ada relasi antara table pembeli dan table pesanan dengan kardinalitas one-to-many. Prosedur membuat query akan seperti berikut:

Pada jendela *Database* pilih *Query* lalu klik dua kali *Create Query In Design View*. Sehingga akan muncul jendela Show Tabel. Pilih table *pembeli* dan *pesanan* pada jendela *Show Table* (Gambar 17.28), kemudian click **Add** dan kemudian **Close**. Jendela *Show Table* akan tertutup dan kita dapat mulai melakukan query.



Gambar 17.28. Pemilihan table untuk query dua table.

Ketika kita memilih table pembeli dan pesanan seperti tampak pada Gambar 17.28, secara otomatis Microsoft Access akan menampilkan garis relasi antara kedua table tersebut. Apabila kita belum membuat relasi, maka garis relasi tidak akan muncul. Pada Gambar 17.29 terlihat pada bagian Field untuk kolom nama dan alamat, Table nya adalah pembeli. Sedangkan pada jumlah_pembelian, tabelnya adalah pesanan. Selain itu pada kolom jumlah_pembelian, kita juga membuat criteria, yaitu yang lebih besar dari 100000. Hasil eksekusi query menunjukkan ada dua orang pembeli yang jumlah_pembeliannya lebih dari 100000.



Gambar 17.29. Query dua tabel

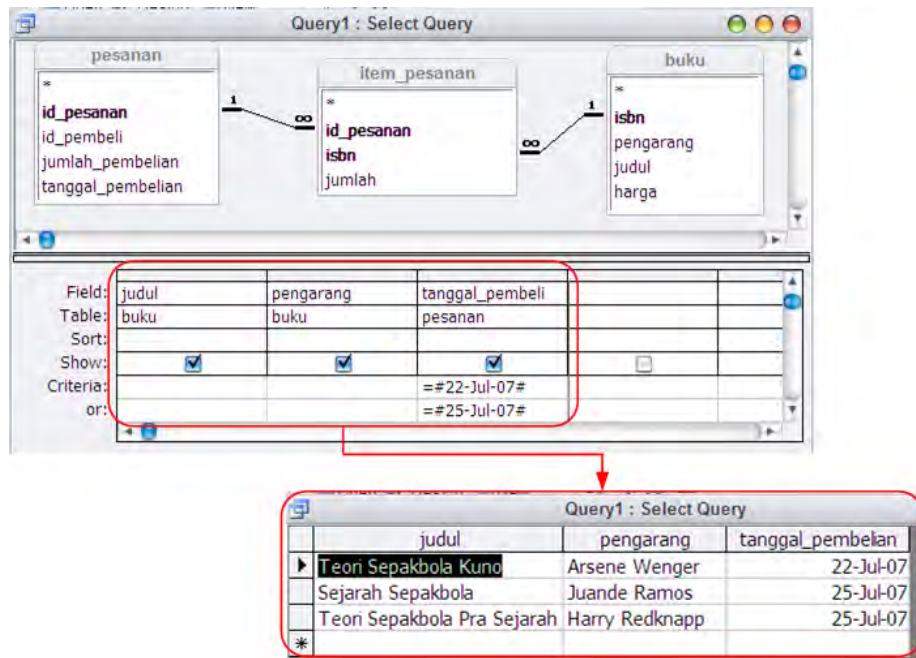
2. Contoh query yang kedua adalah query yang melibatkan tiga table, misalnya:

Tampilkan judul buku dan pengarangnya yang dibeli pada tanggal 22 Juli 2007 atau 25 Juli 2007.

Pada query ini kita membutuhkan table buku karena kolom judul dan pengarang ada pada table buku. Kita juga membutuhkan table pesanan karena kolom tanggal pembelian ada pada table ini. Namun dari relasi antar table pada Gambar 17.20, kita tahu bahwa table pesanan dan table buku tidak ada relasi langsung. Tabel pesanan berhubungan langsung dengan table item_pesanan dan table

item_pesanan berhubungan langsung dengan table buku. Semua relasinya berkardinalitas one-to-many. Sehingga pembuatan query akan seperti berikut:

Pilih table buku, item_pesanan dan pesanan pada jendela Show Tabel. Kemudian click **Add** dan kemudian **Close**. Seperti tampak pada Gambar 17.30 terlihat relasi antara ketiga table tersebut. Hasil eksekusi query ini menghasilkan tiga record seperti terlihat pada gambar.



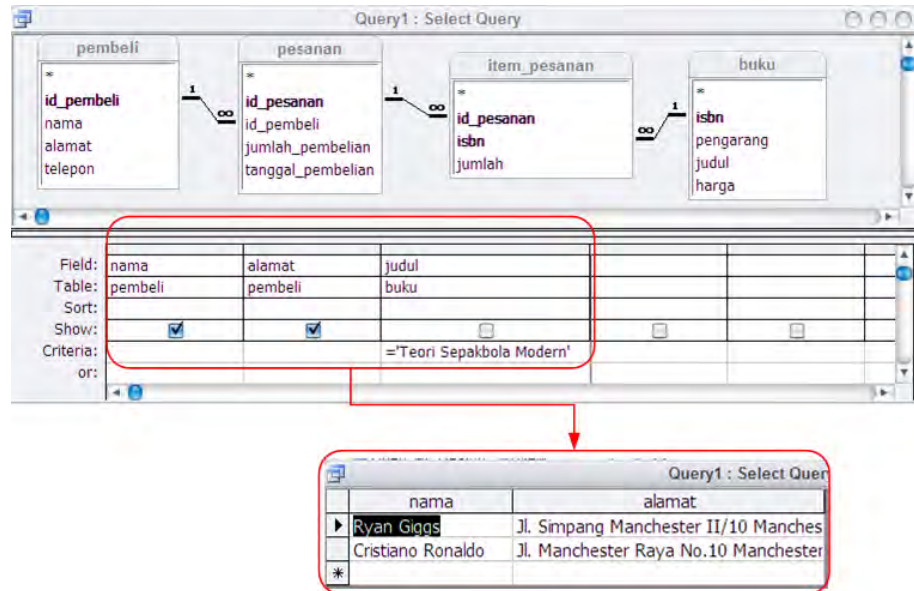
Gambar 17.30. Query tiga table.

3. Contoh query yang ketiga adalah query yang melibatkan semua table pada basis data penjualan buku, misalnya:

Tampilkan nama dan alamat pembeli yang membeli buku dengan judul Teori Sepakbola Modern.

Pada query ini kita membutuhkan table pembeli karena kolom nama dan alamat ada pada table pembeli. Kita juga membutuhkan table buku karena kolom judul hanya ada pada table ini. Dari relasi antar table pada Gambar 17.20, kita tahu bahwa table pembeli dan table buku tidak ada relasi langsung, namun harus melalui table pesanan dan table item_pesanan. Oleh karena itu kita membutuhkan keempat table tersebut dalam query ini. Pada Gambar 17.31 terlihat bagaimana

query ini harus dibuat. Hasil eksekusi query ini menghasilkan tiga record seperti terlihat

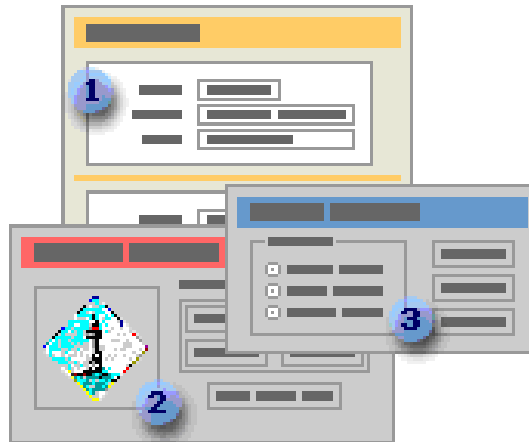


Gambar 17.31. Query empat table.

17.4 FORM

Form adalah salah satu obyek basis data dalam Microsoft Access yang digunakan sebagai antar muka bagi pengguna untuk memasukkan data atau menampilkan data. Bagi pengguna awam, memasukkan data seperti pada Gambar 17.12 sampai dengan 17.15 agak menyulitkan. Jauh lebih mudah menggunakan form. Pada Microsoft Access, dikenal ada tiga model form, yaitu: form data entry (Gambar 17.31 no 1), form switchboard (no 2) dan form custom dialog (no. 3).

Ada dua cara pembuatan form pada Micosoft Access, yaitu dengan menggunakan Wizard dan dengan menggunakan Design View. Wizard merupakan cara yang paling mudah, karena kita Microsoft Access akan melakukan pembuatan form secara otomatis. Sedangkan pada Design View kita melakukan rancangan form secara manual. Kita dapat memodifikasi hasil dari Wizard dengan Design View.

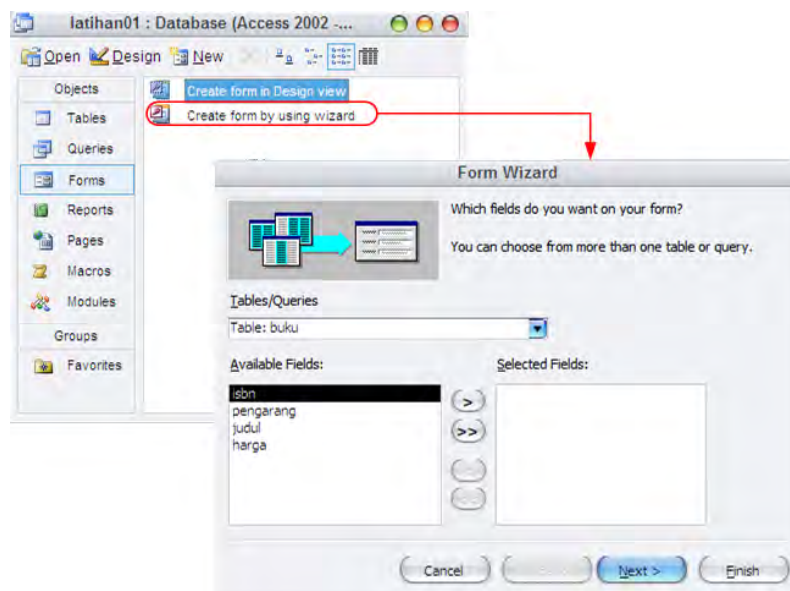


Gambar 17.32. Jenis-jenis form.

17.4.1 Membuat Form

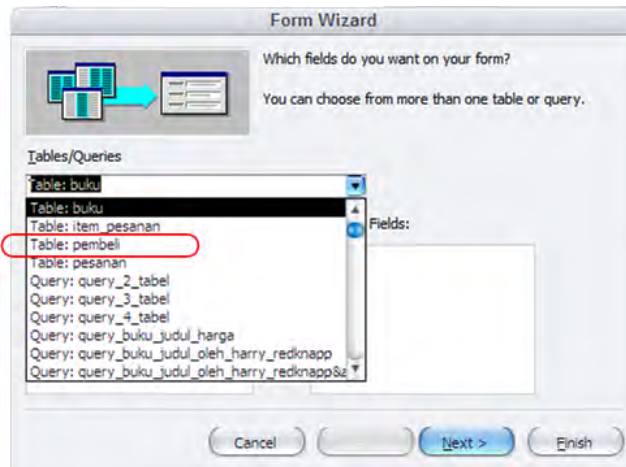
Pada bagian ini kita akan membuat form dengan mode Wizard. Form pertama yang akan kita buat adalah form untuk input data *Pembeli*.

1. Pada jendela Database, click pada object Form, kemudian double click pada Create form by using Wizard. Jendela Form Wizard akan terbuka (Gambar 17.33)

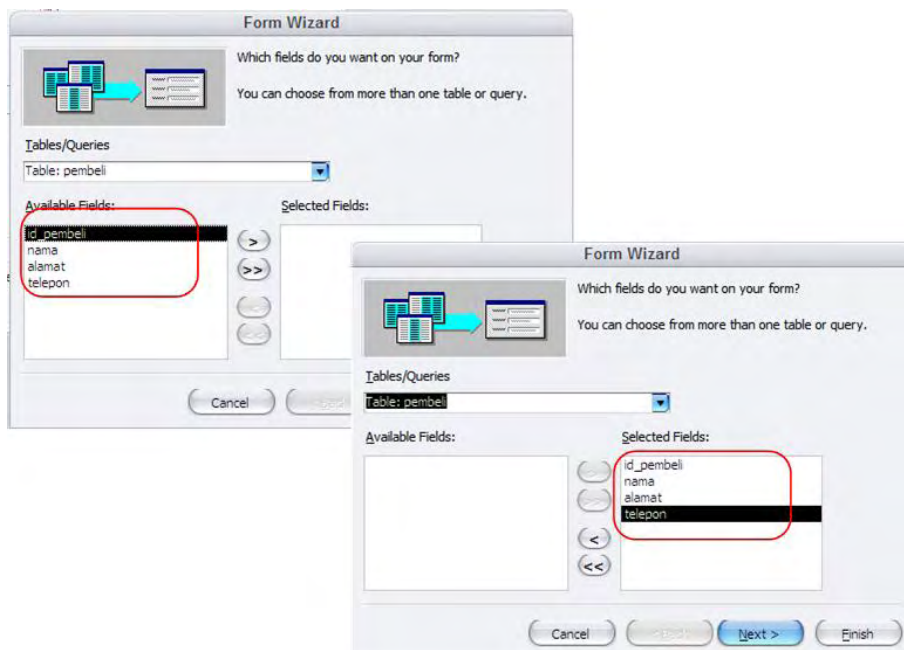


Gambar 17.33. Membuka jendela Form Wizard.

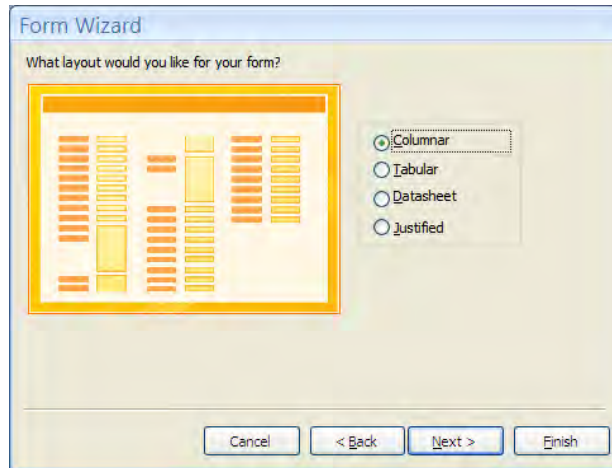
2. Pada jendela Form Wizard, click pada combo box Table/Queries dan pilih table pembeli (Gambar 17.34), kemudian click tombol >> sehingga semua field yang ada pada bagian Available Fields berpindah ke Selected Fields (lihat Gambar 17.35). Kemudian click Next untuk memunculkan jendela untuk memilih model tampilan form (Gambar 17.36)



Gambar 17.34. Pemilihan table yang akan dibuat formnya.

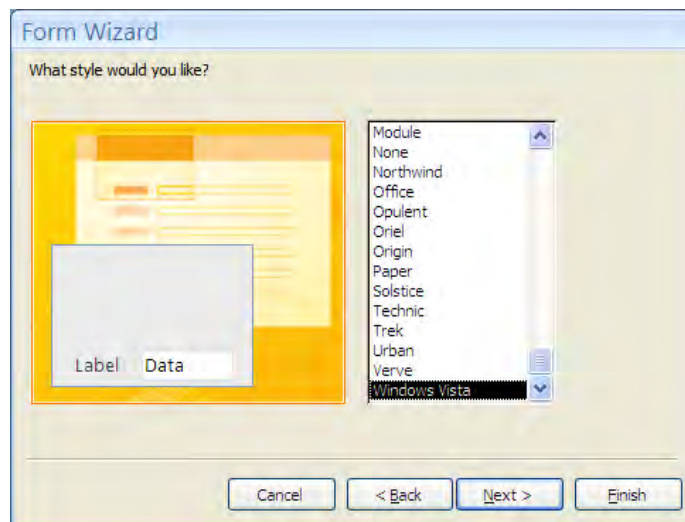


Gambar 17.35. Pemilihan field untuk form.

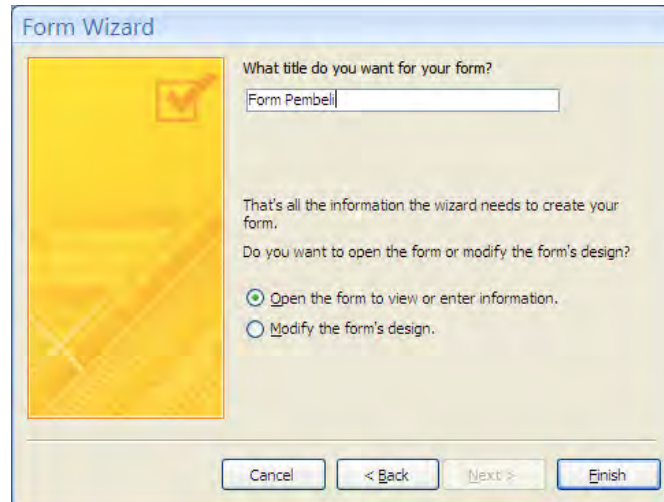


Gambar 17.36. Jendela untuk memilih model tampilan form.

3. Pada Gambar 17.36, tersedia beberapa opsi model tampilan form. Kita akan mencoba membuat dengan model Columnar. Namun pembaca dapat mencoba membuat dengan model tampilan lain yang tersaji pada pilihan. Click tombol Next maka akan muncul jendela untuk memilih style (Gambar 17.37). Pada jendela ini kita dapat memilih sesuai keinginan kita dan setelah selesai kita click Next sehingga muncul jendela untuk member Title atau judul form yang telah kita buat. Isikan nama form pada Textbox yang telah disediakan (Gambar 17.38).

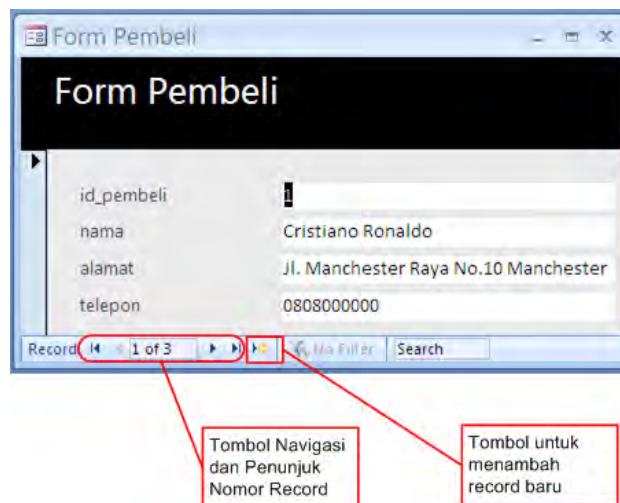


Gambar 17.37. Jendela untuk memilih style form.



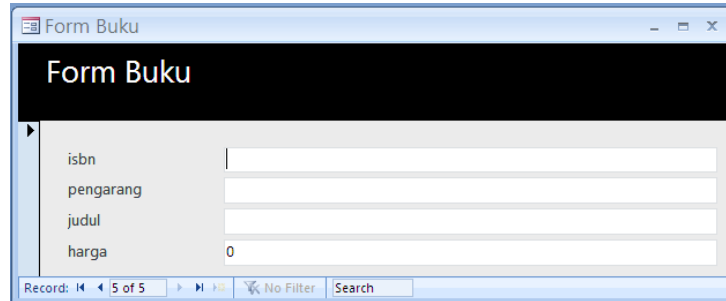
Gambar 17.38. Jendela untuk memberi nama form.

4. Click tombol Finish pada Gambar 17.38. Form pembeli telah selesai kita buat dan hasilnya tampak pada Gambar 17.39. Ada bagian yang sangat penting pada Form telah kita buat. Yaitu tombol navigasi dan tombol menambah record baru. Tombol navigasi berfungsi untuk melihat data yang telah kita buat secara urut berdasarkan id_pembeli. Sedangkan apabila tombol menambah form di-*click*, maka teks yang ada pada seluruh textbox akan dikosongkan dan kita dapat mulai mengisi data baru.



Gambar 17.39. Form Pembeli.

Dengan cara yang sama kita dapat membuat form-form yang lain. Sebagai contoh berikut ini tampilan akhir untuk Form Buku.

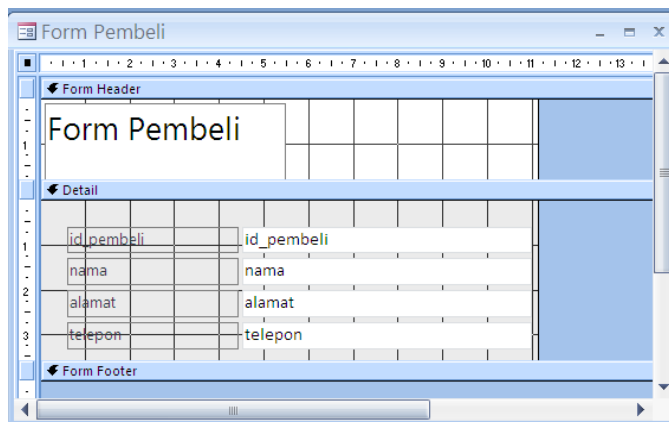


Gambar 17.40. Form Buku.

17.4.2 Memodifikasi Form

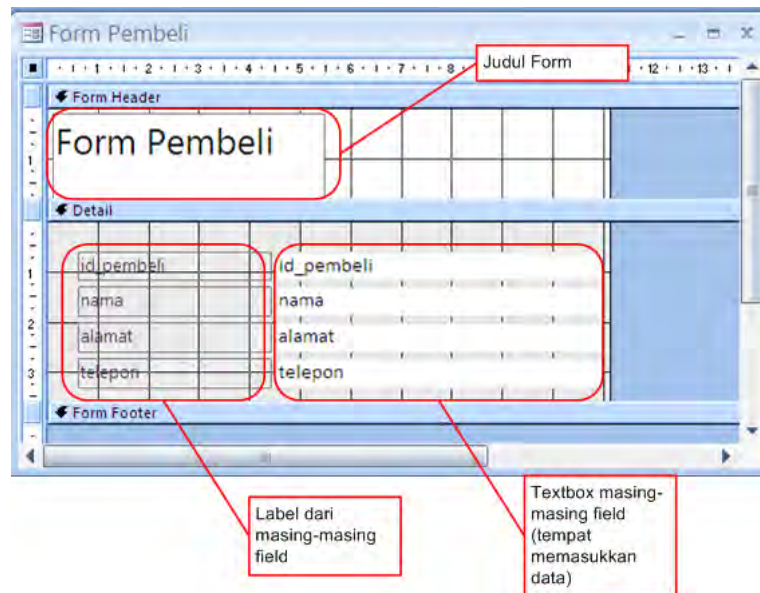
Membuat form dengan menggunakan mode wizard memiliki keuntungan yaitu, kemudahan dan kecepatan dalam pembuatan. Namun disisi lain juga memiliki kerugian karena semua format telah ditentukan secara otomatis. Sebagai contoh baik pada Gambar 17.39 dan 17.40, kita melihat nama-nama fields pada form dibuat sama persis seperti pada nama fields pada tabelnya. Untungnya pada Microsoft Access menyediakan fasilitas untuk memodifikasi form. Berikut ini langkah-langkah untuk memodifikasi form.

1. Pada jendela Database, pilih Form. Kemudian click kanan pada nama form yang ingin kita modifikasi. Pilih Design View pada pop menu yang muncul, sehingga jendela form yang akan kita modifikasi muncul seperti pada Gambar 17.41.



Gambar 17.41. Jendela Form Pembeli pada mode Design View.

2. Pada Gambar 17.41, yang akan kita akan modifikasi adalah Form Pembeli. Sebelum kita modifikasi kita perlu tahu bagian-bagian dari suatu form. Gambar 17.42. menunjukkan bagian-bagian utama dari suatu form. Secara umum ada tiga bagian pada form, yaitu Form Header, Detail dan Form Footer. Form Header biasanya berisi judul form dan logo tertentu. Bagian Detail merupakan bagian utama karena disinilah tempat kita meletakkan field-field yang akan diisi datanya. Bagian Form Footer bersifat opsional (boleh dipakai boleh tidak). Biasanya Form Footer akan berisi keterangan-keterangan lain misalnya nama pembuat, nama pengisi data dan lain-lain.



Gambar 17.42. Bagian-bagian suatu form.

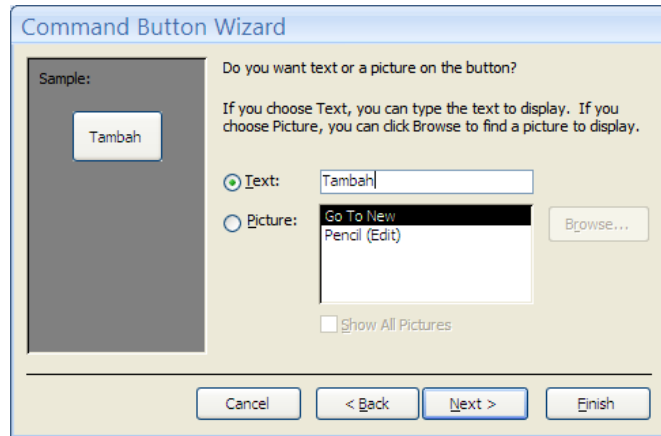
3. Microsoft Access menggunakan control-control yang sama seperti pada bahasa pemrograman berbasis GUI (lihat kembali Bab 13). Pada form kita akan menjumpai control Label, Text Box, Combo Box, Command Button dan lain-lain.
4. Modifikasi pertama yang kita lakukan adalah memperbaiki tampilan pada nama-nama fields. Caranya dengan click pada Label yang ingin kita perbaiki. Kemudian ganti atau ubah dengan cara mengetikkan nama yang baru. Setelah semua selesai tampilan Form Pembeli akan tampak seperti pada Gambar 17.43. Perhatikan perbedaan Label fields pada Gambar 17.41 dan Gambar 17.43.

Gambar 17.43. Perubahan pada Label fields pada Form Pembeli.

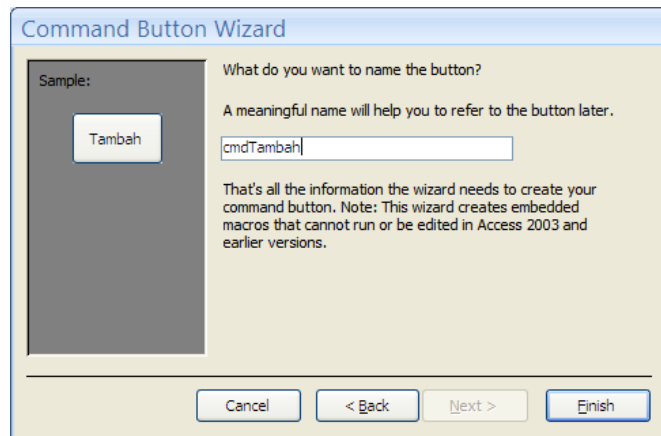
5. Modifikasi berikutnya yang akan kita lakukan adalah menambahkan tombol-tombol untuk memudahkan pengguna dalam mengisi data. Untuk melakukan ini kita harus mengatur form menjadi seperti pada Gambar 17.44. Kemudian kita tambahkan tombol dengan cara click pada icon Command Button pada Toolbar, lalu letakan pada Form. Jendela Command Button Wizards akan muncul seperti pada Gambar 17.45. Pada *Categories* pilih *Record Operations* dan pada *Actions* pilih *Add New Record*. Click Next untuk melanjutkan. Pada jendela berikutnya (Gambar 17.46) pilih Text dan ketikkan *Tambah* lalu klik Next. Pada jendela berikutnya (Gambar 17.47) ketikkan nama untuk command button ini, kita berikan nama misalnya *cmdTambah* lalu klik Finish.

Gambar 17.44. Modifikasi tampilan form.

Gambar 17.45. Mendefinisikan aksi untuk suatu Command Button.

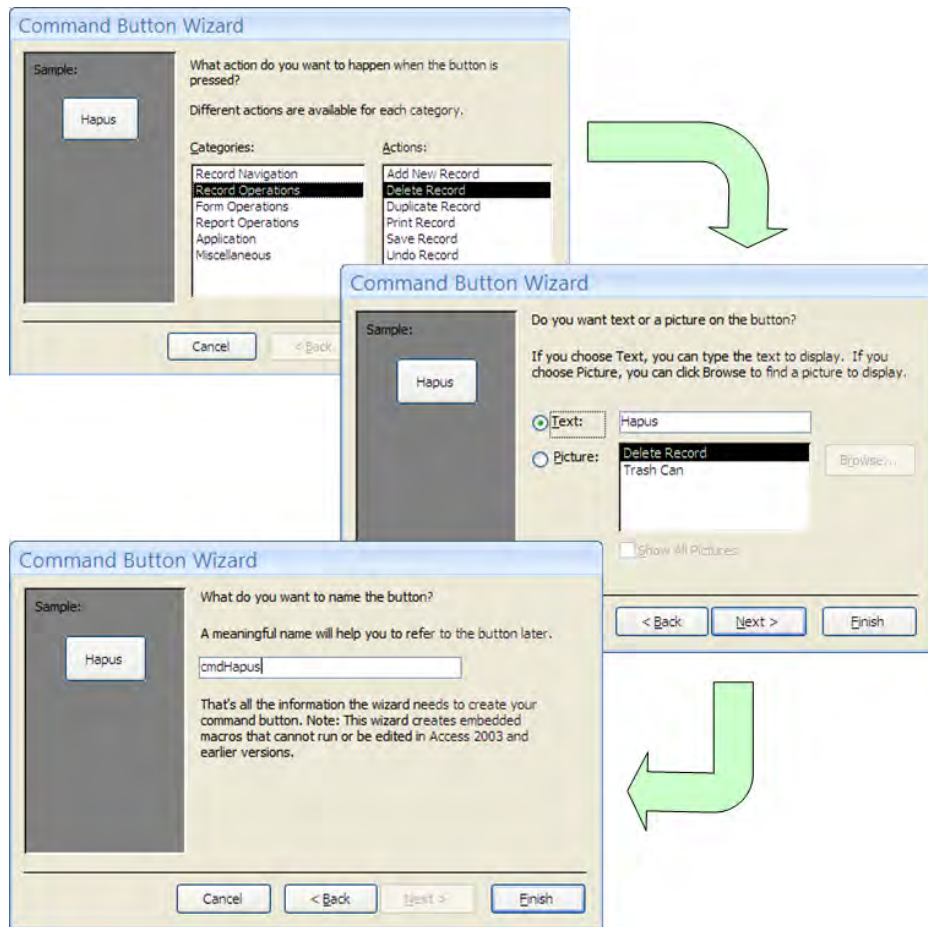


Gambar 17.46. Mendefinisikan teks pada Command Button.



Gambar 17.47. Mendefinisikan nama Command Button.

6. Kita tambahkan tombol yang berfungsi menghapus data. Cara pembuatannya persis seperti di atas hanya saja pada jendela Command Button Wizards, Categories dipilih *Record Operations* dan Actions yang dipilih adalah *Delete Record* (Gambar 17.48). Tentukan teks dan nama command button yang sesuai.



Gambar 17.48. Mendefinisikan nama Command Button.

7. Tampilan akhir dari form setelah dimodifikasi akan tampak seperti pada Gambar 17.49.

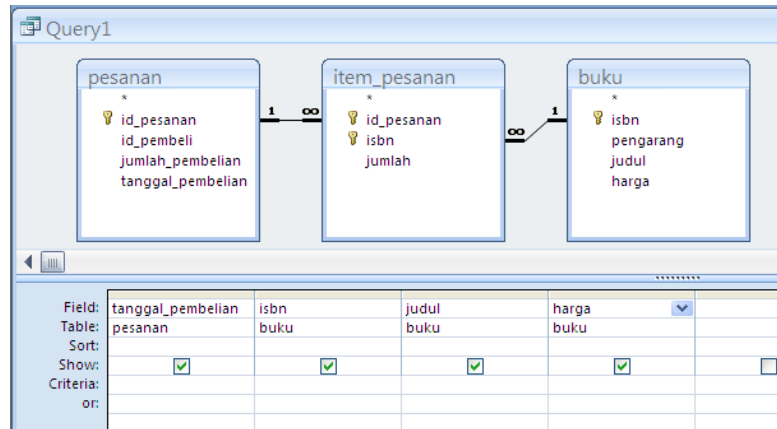
Gambar 17.49. Hasil modifikasi Form Pembelian.

Kita dapat menambahkan tombol-tombol lain atau control-control lain dengan cara yang kurang lebih sama.

17.5 REPORT

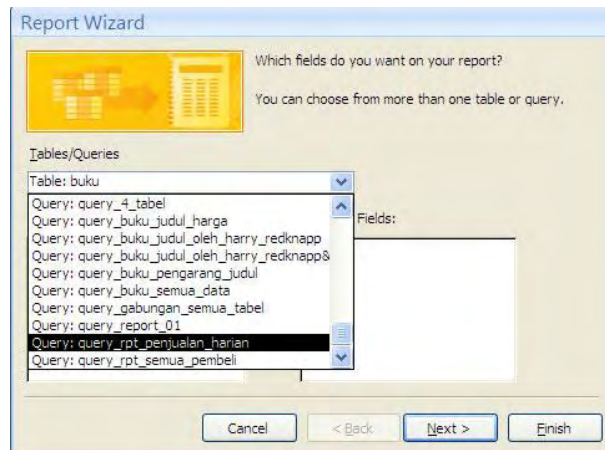
Report, seperti halnya form, digunakan untuk merepresentasikan hasil olahan data menjadi informasi yang siap di cetak di lembaran kertas. Kita dapat saja mencetak langsung dari table database namun hasil cetakannya tidak seperti laporan yang diinginkan. Cara yang terbaik adalah dengan membuat model laporan dengan fasilitas Report. Report dapat dibuat dengan dua cara yaitu manual dan wizards. Seperti halnya pada form, wizard memberikan kemudahan dalam pembuatan report, karena semuanya sudah diatur otomatis. Berikut ini kita akan membuat laporan penjualan harian. Laporan ini berisi tanggal transaksi, ISBN dan judul buku yang terjual, harga masing-masing buku dan total nilai penjualan per hari/tanggal.

1. Untuk membuat report ini kita tidak bias langsung dari table, karena report ini berisi gabungan dari beberapa table yang ada. Sehingga kita harus membuat query sebagai sumber data untuk laporan. Cara membuat query sama dengan yang kita lakukan sebelumnya. Perhatikan Gambar 17.50 berikut ini. Pada Gambar tersebut kita memilih table pesanan, item_pesanan dan buku. Kemudian kita memilih tanggal_pembelian, isbn, judul, dan harga pada bagian Field. Kemudian kita simpan dengan nama query_rpt_penjualan_harian (atau dengan nama yang lain).

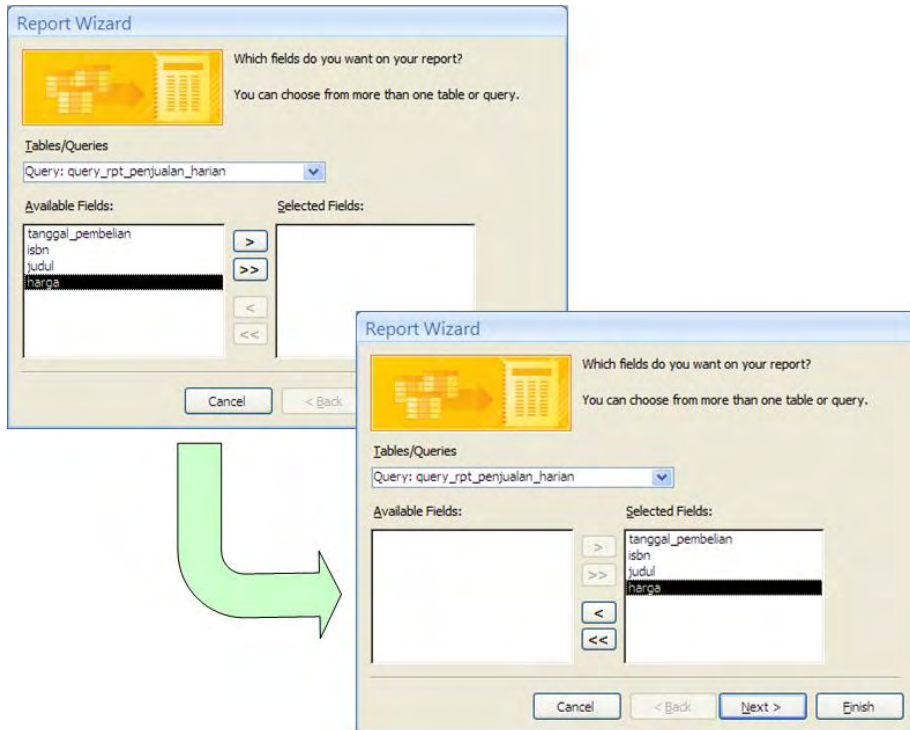


Gambar 17.50. Jendela query untuk sumber report.

2. Pada jendela Database pilih object Report dan double click pada Create Report by using Wizard. Jendela Report Wizard akan terbuka dan pada bagian Tables/Queries pilih query yang telah kita buat pada bagian 1 (lihat Gambar 17.51). Kemudian click tombol >> sehingga semua field yang ada pada bagian Available Fields berpindah ke Selected Fields (lihat Gambar 17.52).

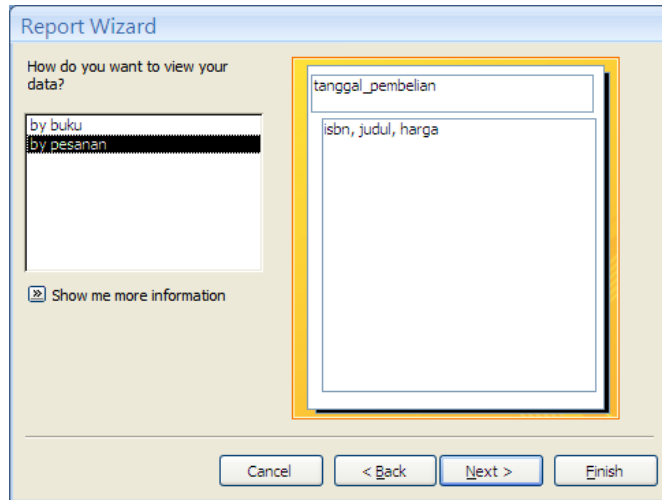


Gambar 17.51. Pemilihan query sebagai sumber data laporan.

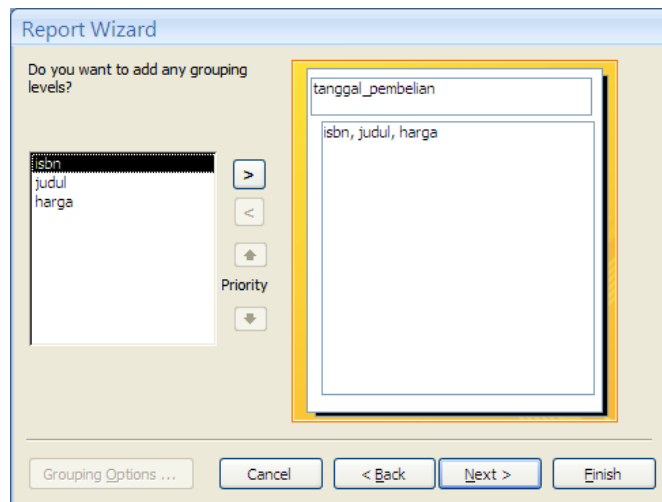


Gambar 17.52. Pemilihan fields yang terlibat.

3. Click Next untuk membuka jendela berikutnya (Gambar 17.53). Pada jendela ini kita menentukan dasar tampilan laporan, apakah berdasarkan buku atau pesanan. Karena kita akan membuat laporan harian maka tampilan berdasarkan pesanan yang kita pilih (ingat, field tanggal_pembelian ada pada table pesanan).
4. Click Next untuk melanjutkan dengan jendela berikutnya (Gambar 17.54). Jendela ini digunakan untuk mengelompokkan (grouping) data pada field yang sama. Pada contoh kali ini kita tidak melakukan grouping sehingga kita tidak perlu mengatur apa-apa pada jendela ini. Kita dapat langsung click Next untuk melanjutkan pada jendela berikutnya.



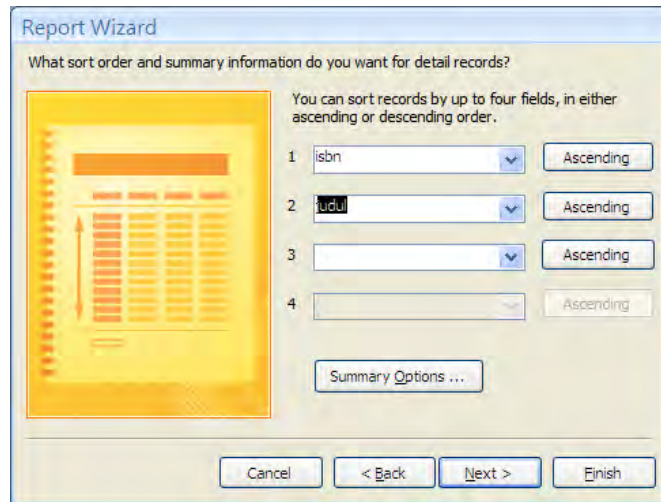
Gambar 17.53. Jendela untuk menentukan dasar tampilan report.



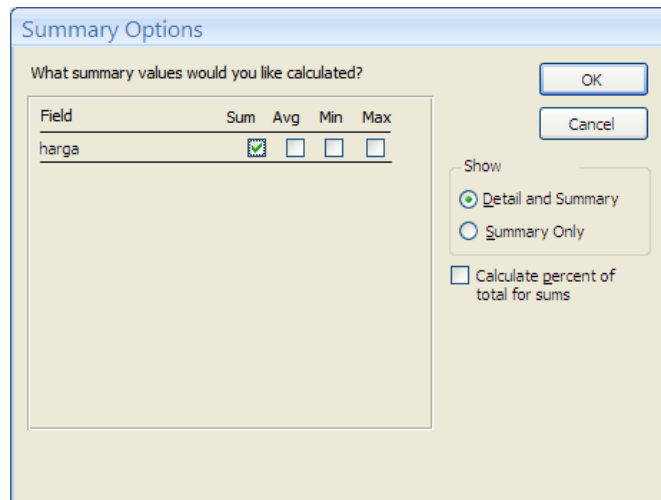
Gambar 17.54. Jendela untuk menentukan grouping data.

5. Pada Gambar 17.55, kita dapat memilih melakukan pengurutan data atau tidak. Pada bagian ini kita akan mengurutkan berdasarkan nomor ISBN dan kemudian berdasarkan judul buku. Pada Combo Box no 1 kita pilih isbn dan pada Combo Box no 2 kita pilih judul. Selain itu pada bagian ini kita juga mengatur apakah kita membuat ringkasan laporan atau tidak. Click pada Summary Options untuk membuka jendela pengaturan ringkasan (Gambar 17.56). Pada field harga kita check pada kotak di bawah Sum. Maksud dari bagian ini adalah kita akan menampilkan jumlah pembelian atau transaksi tiap harinya dan

total transaksi. Jika sudah selesai kita dapat menutup jendela Summary Option dengan meng-click tombol OK. Kemudian kita click Next untuk melanjutkan dengan jendela berikutnya.



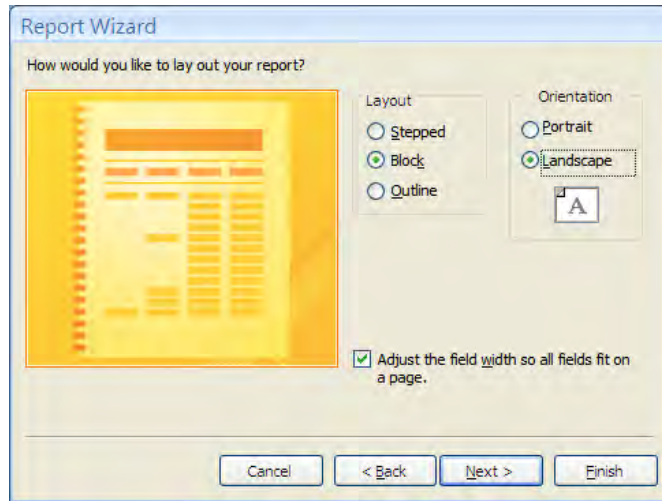
Gambar 17.55. Jendela untuk menentukan urutan data.



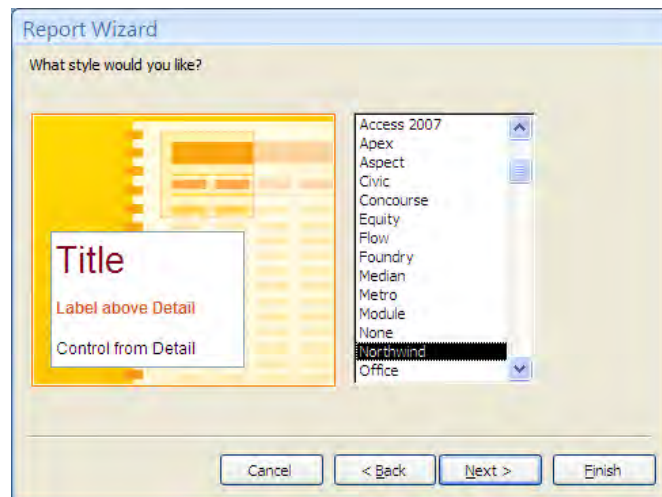
Gambar 17.56. Jendela untuk mengatur tampilan ringkasan.

6. Gambar 17.57 menunjukkan jendela untuk mengatur lay-out dari laporan. Kita dapat mencoba-coba lay-out mana yang sesuai. Pada contoh ini kita akan mencoba dengan lay-out Block dan Orientation Landscape. Click Next jika sudah selesai dan jendela untuk mengatur style dari laporan (Gambar 17.58). Pada bagian ini kita dapat memilih

sesuai keinginan kita. Setelah selesai click Next untuk membuka jendela terakhir dari Report Wizard. Pada bagian Text Box yang tersedia kita tentukan judul dari laporan yang telah kita buat. Click tombol Finish untuk mengakhiri proses pembuatan laporan. Hasil pembuatan laporan akan ditampilkan seperti pada Gambar 17.59.



Gambar 17.57. Jendela untuk mengatur lay-out dan orientation.



Gambar 17.58. Jendela untuk mengatur style laporan.

judul kolom, double-click pada judul kolom yang ingin diganti kemudian ketikkan nama yang baru. Kita juga dapat mengatur lebar kolom dengan cara click pada bagian paling atas kolom untuk memilih seluruh kolom kemudian letakkan kursor pada samping kanan kolom dan geser untuk memperlebar kolom (sama persis dengan mengatur lebar table pada Microsoft Word).

9. Perbaikan yang kedua adalah kita menghilangkan kata-kata Summary for tanggal pembelian dan seterusnya yang tampak pada Gambar 17.59. Caranya dengan click pada bagian tersebut kemudian tekan tombol Del pada keyboard. Selain itu kita hilangkan kata Sum dan kita ganti kata Grand Total dengan Total Penjualan. Penambahan garis juga dapat dilakukan dengan memilih control line pada Toolbar Report, kemudian diletakkan pada posisi yang diinginkan. Tampilan akhir pada Design View akan tampak seperti pada Gambar 17.61. Sedangkan hasil Print Preview dari laporan akan tampak seperti pada Gambar 17.62.

Laporan Penjualan Harian									
Page Header									
Tanggal Transaksi		ISBN	Judul Buku			Harga (Rp)		Sub Total	
tanggal_pembelian Header									
Detail									
tanggal_pembelian	isbn	judul			harga				
tanggal_pembelian Footer									
							=Sum([harga])		
Page Footer									
=Now()					=Page & [Page] of [Total Pages]				
Report Footer									
							Total Penjualan		=Sum([harga])

Gambar 17.61. Design laporan setelah dilakukan perbaikan.

Laporan Penjualan Harian

Tanggal Transaksi	ISBN	Judul Buku	Harga (Rp)	Sub Total
12-Apr-07	0-672-31667-9	Teori Sepakbola Modern	65000	
				65000
22-Jul-07	0-672-31667-8	Teori Sepakbola Kuno	75000	
				75000
25-Jul-07	0-672-31337-3	Sejarah Sepakbola	105000	
	0-672-31637-9	Teori Sepakbola Pra Sejarah	55000	
				160000
10-Aug-07	0-672-31637-9	Teori Sepakbola Pra Sejarah	55000	
				55000
15-Sep-07	0-672-31637-9	Teori Sepakbola Pra Sejarah	55000	
	0-672-31667-9	Teori Sepakbola Modern	65000	
				120000
Total Penjualan				475000

Gambar 17.62. Print Preview laporan setelah perbaikan.

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian Data Definition Language dan Data Manipulation Language.
- Menjelaskan pengertian Query.
- Membuat query pada suatu basis data.
- Menggunakan SQL untuk mendefinisikan dan memanipulasi suatu basis data.

18.1 BAHASA BASIS DATA

Dalam DBMS biasanya tersedia paket bahasa yang digunakan untuk mengorganisasi basis data yang ada, yaitu Data Definition Language (DDL) dan Data Manipulation Language (DML)

18.1.1 Data Definiton Language

Data Definition Language (DDL) :

satu paket bahasa DBMS yang berguna untuk melakukan spesifikasi terhadap skema basis data

Hasil kompilasi dari DDL adalah satu set tabel yang disimpan dalam file khusus yang disebut Data Directory/Dictionary. Secara umum perintah perintah dalam DDL berhubungan dengan operasi-operasi dasar seperti membuat basis data baru, menghapus basis data, membuat table baru, menghapus table, membuat index, mengubah struktur table. Contoh perintah DDL misalnya, *Create Table, Create Index, Alter, dan Drop Database*.

18.1.2 Data Manipulation Language

Data Manipulation Language (DML) :

satu paket DBMS yang memperbolehkan pemakai untuk mengakses atau memanipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat.

Dengan DML dapat dilakukan :

- Mengambil informasi yang tersimpan dalam basis data.
- Menyisipkan informasi baru dalam basis data.
- Menghapus informasi dari tabel.

Terdapat dua tipe DML yaitu prosedural dan non prosedural. Prosedural DML membutuhkan pemakai untuk menspesifikasikan data apa yang dibutuhkan dan bagaimana cara mendapatkannya, sedang non prosedural DML membutuhkan pemakai untuk menspesifikasikan data apa yang dibutuhkan tanpa tahu bagaimana cara mendapatkannya.

18.2 SQL

SQL merupakan kependekan dari *Structured Query Language* (Bahasa Query Terstruktur). Seperti dijelaskan bahasa ini lebih dekat dengan DML dari pada DDL. Namun tidak berarti SQL tidak menyediakan perintah DDL. SQL lebih menekankan pada aspek pencarian dari dalam table. Aspek pencarian ini sedemikian penting karena di sinilah sebenarnya inti dari segala upaya kita melakukan pengelolaan data. Data dalam basis data diorganisasi sedemikian rupa dengan tujuan untuk memudahkan pencarian di kemudian hari.

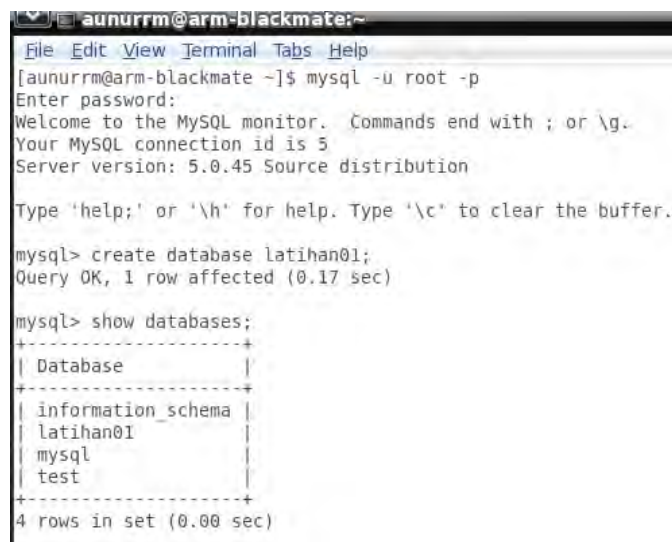
Sebagai sebuah bahasa, SQL telah distandarisasi dan mengalami beberapa perubahan atau penyempurnaan. SQL muncul pertama kali pada tahun 1970 dengan nama Sequel (nama yang masih sering digunakan hingga saat ini). Standarisasi yang pertama dibuat pada tahun 1986 oleh ANSI (American National Standards Institute) dan ISO (International Standard Organization), yang disebut SQL-86. Pada tahun 1989 SQL-86 diperbaharui menjadi SQL-89. Standar terakhir yang dibuat adalah SQL-92.

Pernyataan-pernyataan SQL digunakan untuk melakukan beberapa tugas seperti : update data pada basis data, atau menampilkan data dari basis data. Beberapa software RDBMS yang dapat menggunakan SQL, seperti : Oracle, Sybase, Microsoft SQL Server, MySQL, Microsoft Access, Ingres, dsb. Setiap software basis data mempunyai bahasa perintah / sintaks yang

berbeda, namun pada prinsipnya mempunyai arti dan fungsi yang sama. Perintah-perintah tsb antara lain : "Select", "Insert", "Update", "Delete", "Create", dan "Drop", yang dapat digunakan untuk mengerjakan hampir semua kebutuhan untuk memanipulasi sebuah basis data. Pada sub bab berikutnya akan disampaikan perintah-perintah SQL utama dan contoh penggunaannya. Pada contoh-contoh yang digunakan, kita akan menggunakan RDBMS MySQL Server dengan mode teks. Kasus yang kita pakai sama persis dengan kasus pada Bab 17 yaitu tentang Basis Data Penjualan Buku. Ada empat buah table yang akan kita buat yaitu table Pembeli, Buku, Pesanan dan Item_Pesanan.

18.2.1 Pembuatan Basis Data

Perintah untuk pendefinisian atau pembuatan basis data baru adalah **create database**. Perintah ini termasuk dalam kelompok DDL. Perhatikan gambar berikut ini.



```
aunurrm@arm-blackmate:~  
File Edit View Terminal Tabs Help  
[aunurrm@arm-blackmate ~]$ mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 5  
Server version: 5.0.45 Source distribution  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql> create database latihan01;  
Query OK, 1 row affected (0.17 sec)  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| latihan01 |  
| mysql |  
| test |  
+-----+  
4 rows in set (0.00 sec)
```

Gambar 18.1. Contoh perintah SQL untuk pembuatan basis data baru.

Pada gambar di atas kita menggunakan MySQL client sebagai antar muka untuk bekerja dengan MySQL Server. Perintah **mysql -u root -p** adalah perintah untuk dapat masuk (login) ke MySQL. User yang digunakan adalah root dan passwordnya tertentu. Pada gambar juga tampak kita membuat basis data dengan nama latihan01. Perhatikan cara pembuatan basis data dengan perintah **create database**. Untuk dapat melihat daftar basis data yang ada, pada MySQL disediakan perintah **show databases**. Setiap perintah dalam MySQL akan dieksekusi jika pada bagian akhir perintah diakhiri dengan tanda titik koma (;).

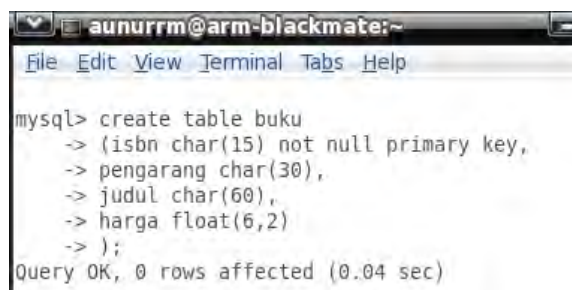
18.2.2 Pembuatan Tabel

Perintah untuk pendefinisian atau pembuatan tabel baru adalah **create table**. Perintah ini juga termasuk dalam DDL. Perhatikan Gambar 18.2 berikut ini. Pada gambar ini, kita membuat *table pembeli* dengan kolom-kolom yang sama pada Bab 17. Ada empat kolom yaitu *id_pembeli*, *nama*, *alamat* dan *telepon*. *Id_pembeli* bertipe data *int* (integer), memiliki constraint *not null*, bersifat *auto_increment* (nilai otomatis naik satu tingkat setiap kali ada penambahan record) dan merupakan *primary key* pada *table pembeli*. Kolom yang lainnya bertipe data *char* (sama dengan *text* pada Microsoft Access). MySQL menyediakan tipe data yang lebih banyak daripada Microsoft Access. Tipe data tersebut antara lain: *tinyint*, *smallint*, *int*, *float*, *double*, *real*, *decimal*, *date*, *time*, *datetime*, *timestamp*, *char*, *varchar*, *text*, *blob*, *enum* dan *set*. Penjelasan tentang masing-masing tipe data dapat dibaca pada manual MySQL.

```
mysql> create table pembeli
-> (id_pembeli int unsigned not null auto_increm
-> nama char(30) not null,
-> alamat char(40) not null,
-> telepon char(15)
-> );
Query OK, 0 rows affected (0.03 sec)
```

Gambar 18.2. Perintah pembuatan pembeli.

Tabel-tabel lain dapat kita buat dengan cara yang sama. Gambar 18.3 memperlihatkan pembuatan *table buku*. Pada gambar tersebut terlihat bahwa field *harga* memiliki tipe data *float(6,2)*. Maksud tipe data ini adalah *harga* akan berisi nilai-nilai numerik dalam bentuk bilangan real dengan lebar data maksimal 6 digit termasuk dua angka desimal (dalam bentuk *xxxx.xx*). Untuk melihat daftar *table* yang sudah dibuat, MySQL memberi fasilitas perintah **show tables** seperti terlihat pada Gambar 18.4.



```
aunurrm@arm-blackmate:~
File Edit View Terminal Tabs Help

mysql> create table buku
-> (isbn char(15) not null primary key,
-> pengarang char(30),
-> judul char(60),
-> harga float(6,2)
-> );
Query OK, 0 rows affected (0.04 sec)
```

Gambar 18.3. Perintah pembuatan table buku.

```

aunurrm@arm-blackmate:~
File Edit View Terminal Tabs Help

Database changed
mysql> show tables;
+-----+
| Tables in latihan01 |
+-----+
| buku              |
| item_pesanan      |
| pembeli           |
| pesanan           |
+-----+
4 rows in set (0.00 sec)

```

Gambar 18.4. Perintah **show tables** untuk melihat daftar table.

Untuk memeriksa atau melihat struktur masing-masing table, kita dapat menggunakan perintah **describe**. Perhatikan contoh penggunaan perintah tersebut pada Gambar 18.5.

18.2.3 Perubahan Pada Atribut Tabel

Seringkali setelah kita membuat table dalam basis data, kita ingin merubah struktur table tersebut. Biasanya karena kita salah menggunakan tipe data atau salah menetapkan lebar data. Untuk merubah atau menambah struktur table yang sudah kita buat, kita dapat menggunakan perintah **alter**. Perintah ini masih termasuk dalam DDL. Contoh penggunaan perintah ini dapat dilihat pada Gambar 18.6.

```

aunurrm@arm-blackmate:~
File Edit View Terminal Tabs Help

mysql> describe buku;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| isbn       | char(15)  | NO   | PRI |          |       |
| pengarang  | char(30)  | YES  |     | NULL    |       |
| judul      | char(60)  | YES  |     | NULL    |       |
| harga      | float(6,2)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe item_pesanan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pesanan | int(10) unsigned | NO   | PRI |          |       |
| isbn       | char(15)  | NO   | PRI |          |       |
| jumlah     | tinyint(3) unsigned | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

Gambar 18.5. Penggunaan perintah **describe**.

```
File Edit View Terminal Tabs Help
mysql> describe pesanan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pesanan     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| id_pembeli     | int(10) unsigned | NO   |     | NULL    |               |
| jumlah_pembelian | float(8,2)      | YES  |     | NULL    |               |
| tanggal_pembelian | date          | NO   |     |         |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

↓

```
mysql> alter table pesanan
-> modify jumlah_pembelian float(10,2);
Query OK, 5 rows affected (0.16 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

↓

```
mysql> describe pesanan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pesanan     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| id_pembeli     | int(10) unsigned | NO   |     | NULL    |               |
| jumlah_pembelian | float(10,2)     | YES  |     | NULL    |               |
| tanggal_pembelian | date          | NO   |     |         |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Gambar 18.6. Penggunaan perintah alter.

Pada Gambar 18.6 di atas, di table awal (bagian atas), terlihat bahwa jumlah_pembelian memiliki tipe data float dengan lebar data (8,2). Kita mungkin merasa lebar data ini tidak mencukupi sehingga kita ingin merubahnya menjadi (10,2). Pada table akhir (bagian bawah), terlihat lebar data untuk jumlah_pembelian sudah berubah.

18.2.4 Pengisian Data Pada Tabel

Setelah semua table selesai dibuat, maka kita dapat mengisi data pada masing-masing table dengan menggunakan perintah insert. Perintah ini termasuk dalam kelompok DML. Gambar 18.7 memperlihatkan bagaimana perintah insert digunakan. Yang harus diperhatikan dalam pengisian data adalah kolom-kolom yang ada pada table untuk setiap record. Apabila kita kekurangan menempatkan data maka ada kemungkinan data tidak bisa ditambahkan. Pada gambar tersebut terlihat bahwa table pembeli mempunyai 4 kolom, sehingga kita juga harus memasukkan sebanyak 4 item data. Pada kolom pertama yaitu id_pembeli, kita memasukkan data Null. Hal ini karena kolom id_pembeli didefinisikan sebagai auto_increment (lihat Gambar 18.2) sehingga akan secara otomatis terisi bilangan.

```
aunurrm@arm-blackmate:~  
File Edit View Terminal Tabs Help  
mysql> insert into pembeli values  
-> (NULL,  
-> "Cristiano Ronaldo",  
-> "Jl. Manchester Raya No. 10, Manchester",  
-> "0808000000"  
-> );  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into pembeli values  
-> (NULL,  
-> "Ryan Giggs",  
-> "Jl. Simpang Manchester II / 10, Manchester",  
-> "0801000000"  
-> );  
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Gambar 18.7. Penggunaan perintah insert.

Cara pengisian yang ditunjukkan pada Gambar 18.7, tidak terlalu menguntungkan karena satu persatu record harus kita masukkan. Untuk mempercepat pengisian table, kita dapat memasukkan beberapa record sekaligus dalam satu perintah seperti tampak pada Gambar 18.8. Pada gambar ini kita memasukkan sekaligus 4 buah record. Setiap record ditandai dengan tanda kurung (...). Gambar 18.8 juga memperlihatkan bagaimana cara menuliskan item data. Untuk data dengan tipe char, kita harus menuliskan dengan menggunakan tanda petik dua, misalnya "Arsene Wenger". Sedangkan untuk data numerik (pada kolom harga), kita tidak perlu menggunakan tanda petik dua.

```
aunurrm@arm-blackmate:~  
File Edit View Terminal Tabs Help  
mysql> insert into buku values  
-> ("0-672-31667-8", "Arsene Wenger", "Teori Sepakbola Kuno", 75000.00),  
-> ("0-672-31667-9", "Harry Redknapp", "Teori Sepakbola Modern", 65000.00),  
-> ("0-672-31637-9", "Harry Redknapp", "Teori Sepakbola Pra Sejarah", 55000.00),  
-> ("0-672-31337-3", "Juande Ramos", "Sejarah Sepakbola", 105000.00);  
Query OK, 4 rows affected, 4 warnings (0.00 sec)  
Records: 4 Duplicates: 0 Warnings: 4
```

Gambar 18.8. Memasukkan beberapa record data sekaligus.

18.2.5 Membuat Query

Query atau permintaan data tertentu pada basis data merupakan aktivitas yang paling banyak dilaksanakan. Perintah utama dalam SQL untuk melakukan query adalah **select**. Struktur utama perintah adalah sebagai berikut:

```
Select <kolom>  
From <table>  
Where <kondisi>
```

Ada banyak opsi yang dapat dilakukan dengan struktur perintah tersebut. Berikut ini beberapa contoh penggunaan struktur tersebut.

- ***menampilkan semua data pada sebuah table***

Query untuk menampilkan semua data adalah query yang paling sering dilakukan dan paling mudah. Kita cukup memberikan tanda * didepan pernyataan **select** dan menentukan table mana yang akan ditampilkan di depan pernyataan **from**. Pernyataan **where** tidak kita gunakan pada query ini. Gambar 18.9 memperlihatkan bagaimana cara menampilkan semua data pada sebuah table.

Pada Gambar 18.9, kita menampilkan semua data pada table-tabel yang telah kita buat sebelumnya, yaitu *table pembeli*, *table buku*, *table pesanan* dan *table item_pesanan*. Pada dasarnya query untuk kasus ini, jika kita tuliskan dalam bahasa sehari-hari akan berbunyi sebagai berikut:

Tampilkan semua isi table pembeli.

Tampilkan semua isi table buku.

Tampilkan semua isi table pesanan.

Tampilkan semua isi table item_pesanan.

Dengan menampilkan semua data kita dapat mengetahui banyaknya record untuk masing-masing table.


```
File Edit View Terminal Tabs Help
mysql> select * from pembeli;
```

id_pembeli	nama	alamat	telepon
1	Cristiano Ronaldo	Jl. Manchester Raya No. 10, Manchester	0808000000
2	Ryan Giggs	Jl. Simpang Manchester II / 10, Manchest	0801000000

```
2 rows in set (0.00 sec)

mysql> select * from buku;
```

isbn	pengarang	judul	harga
0-672-31667-8	Arsene Wenger	Teori Sepakbola Kuno	75000.00
0-672-31667-9	Harry Redknapp	Teori Sepakbola Modern	65000.00
0-672-31637-9	Harry Redknapp	Teori Sepakbola Pra Sejarah	55000.00
0-672-31337-3	Juande Ramos	Sejarah Sepakbola	105000.00

```
4 rows in set (0.00 sec)

mysql> select * from pesanan;
```

id_pesanan	id_pembeli	jumlah_pembelian	tanggal_pembelian
1	2	65000.00	2007-04-12
2	1	75000.00	2007-07-22
3	3	160000.00	2007-07-25
4	2	55000.00	2007-08-10
5	1	120000.00	2007-09-15

```
5 rows in set (0.00 sec)

mysql> select * from item_pesanan;
```

id_pesanan	isbn	jumlah
1	0-672-31667-9	1
2	0-672-31667-8	1
3	0-672-31637-9	1
3	0-672-31337-3	1
4	0-672-31637-9	1
5	0-672-31637-9	1
5	0-672-31667-9	1

```
7 rows in set (0.00 sec)
```

Gambar 18.9. Contoh query untuk menampilkan semua data dari sebuah tabel.

- **menampilkan semua data pada kolom tertentu**

Kadang-kadang kita tidak ingin menampilkan data dari seluruh kolom pada sebuah table. Kita mungkin hanya tertarik pada kolom-kolom tertentu saja. Perintah **select** dapat juga digunakan untuk menampilkan kolom-kolom tertentu saja dengan cara menuliskan nama kolom yang kita inginkan didepan pernyataan **select**. Kolom yang kita tuliskan bisa satu atau lebih. Gambar 18.10 memperlihatkan contoh menggunakan pernyataan **select** untuk menampilkan kolom tertentu. Pada gambar ini

kita ingin menampilkan record-record pada kolom nama dan alamat dari table pembeli.



```
aunurrm@arm-blackmate:~  
File Edit View Terminal Tabs Help  
mysql> select nama, alamat from pembeli;  
+-----+-----+  
| nama          | alamat                                     |  
+-----+-----+  
| Cristiano Ronaldo | Jl. Manchester Raya No. 10, Manchester |  
| Ryan Giggs      | Jl. Simpang Manchester II / 10, Manchest |  
| Wayne Rooney    | Jl. Simpang Manchester III/10, Mancheste |  
+-----+-----+  
3 rows in set (0.00 sec)
```

Gambar 18.10. Contoh query untuk menampilkan semua data pada kolom tertentu.

- **menampilkan data dengan kriteria tertentu**

Pada dua contoh di atas, pada dasarnya kita ingin menampilkan semua record. Baik untuk kolom tertentu saja maupun semua kolom dalam sebuah table. Namun sering kita menginginkan menampilkan hanya baris-baris (record) tertentu saja. Pernyataan **where** merupakan pernyataan untuk membatasi atau memilih baris-baris mana saja yang ingin ditampilkan.

Perhatikan pada dua contoh di Gambar 18.11. Pada contoh pertama kita ingin menampilkan seluruh kolom pada *table pesanan* yang record-nya memiliki *id_pembeli* = 2. Ada dua record yang terpilih, yaitu record dengan *id_pesanan* = 1 dan 4.

```
mysql> select * from pesanan where id_pembeli = 2;  
+-----+-----+-----+-----+  
| id_pesanan | id_pembeli | jumlah_pembelian | tanggal_pembelian |  
+-----+-----+-----+-----+  
| 1          | 2          | 65000.00         | 2007-04-12         |  
| 4          | 2          | 55000.00         | 2007-08-10         |  
+-----+-----+-----+-----+  
2 rows in set (0.01 sec)  
  
mysql> select * from pesanan where jumlah_pembelian >= 75000.00;  
+-----+-----+-----+-----+  
| id_pesanan | id_pembeli | jumlah_pembelian | tanggal_pembelian |  
+-----+-----+-----+-----+  
| 2          | 1          | 75000.00         | 2007-07-22         |  
| 3          | 3          | 160000.00        | 2007-07-25         |  
| 5          | 1          | 120000.00        | 2007-09-15         |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

Gambar 18.11. Penggunaan pernyataan **where** sederhana.

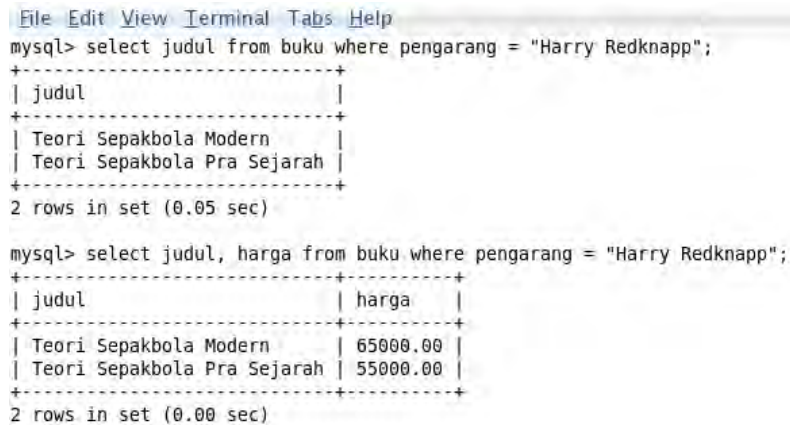
Kita juga dapat menggunakan operator perbandingan seperti >, <, >=, <=, dan <> pada kondisi dibelakang where. Pada Gambar 18.11 contoh yang kedua, kita melihat penggunaan operator >=. Query ini berisi permintaan menampilkan seluruh kolom pada table pesanan yang recordnya memiliki harga lebih besar atau sama dengan 75000.00.

Contoh lain penggunaan where dapat dilihat pada Gambar 18.12 berikut ini. Pada bagian pertama, query-nya dalam bahasa sehari berbunyi sebagai berikut:

Tampilkan judul-judul buku yang dikarang oleh Harry Redknapp.

Sedangkan pada bagian yang kedua, query-nya berbunyi:

Tampilkan judul-judul buku yang dikarang oleh Harry Redknapp beserta harganya.



```
File Edit View Terminal Tabs Help
mysql> select judul from buku where pengarang = "Harry Redknapp";
+-----+
| judul |
+-----+
| Teori Sepakbola Modern |
| Teori Sepakbola Pra Sejarah |
+-----+
2 rows in set (0.05 sec)

mysql> select judul, harga from buku where pengarang = "Harry Redknapp";
+-----+-----+
| judul | harga |
+-----+-----+
| Teori Sepakbola Modern | 65000.00 |
| Teori Sepakbola Pra Sejarah | 55000.00 |
+-----+-----+
2 rows in set (0.00 sec)
```

Gambar 18.12. Contoh lain penggunaan **where**.

Kondisi setelah pernyataan where juga dapat menggunakan operator logika seperti And, Or, atau Not. Aturan tentang operator logika dapat dilihat pada Bab 8. Gambar 18.13 menunjukkan contoh penggunaan operator logika. Pada query bagian pertama, kita menggunakan operator and. Tidak ada hasil yang diperoleh pada bagian pertama ini, karena memang tidak ada record di dalam table pesanan yang id_pembelinya = 1 dan 3. Operator and mensyaratkan kedua kondisi benar, dan ini tidak dipenuhi. Sedangkan pada query bagian kedua kita menggunakan operator or. Ada 3 record yang memenuhi syarat kondisi. Operator or mensyaratkan minimal satu kondisi dalam keadaan benar.

```
File Edit View Terminal Tabs Help
mysql> select * from pesanan where id_pembeli = 1 and id_pembeli = 3;
Empty set (0.00 sec)

mysql> select * from pesanan where id_pembeli = 1 or id_pembeli = 3;
+-----+-----+-----+-----+
| id_pesanan | id_pembeli | jumlah_pembelian | tanggal_pembelian |
+-----+-----+-----+-----+
|          2 |          1 |          75000.00 | 2007-07-22         |
|          3 |          3 |         160000.00 | 2007-07-25         |
|          5 |          1 |         120000.00 | 2007-09-15         |
+-----+-----+-----+-----+
3 rows in set (0.15 sec)
```

Gambar 18.13. Pernyataan where dengan menggunakan operator logika.

- **menampilkan data dari gabungan table**

Sampai dengan Gambar 18.13, kita hanya melakukan query pada satu table. Tetapi dibanyak kasus, kita harus melakukan query yang melibatkan lebih dari satu table. Seperti telah disebutkan, kasus yang kita pakai pada Bab ini sama persis dengan apa yang ada pada Bab 17. Pada penjelasan tentang kasus ini di Bab 17 kita melihat ada hubungan antara keempat buah table ini.

Perhatikan pernyataan query berikut ini:

Tampilkan nama pembeli yang memesan buku tanggal 22-07-2007.

Pernyataan query ini tidak akan bisa dipenuhi hanya oleh *table pembeli*. Hal ini karena data tanggal pembelian tidak ada pada *table pembeli*. Data tanggal pembelian hanya terdapat pada *table pesanan*. Untuk memperoleh hasil yang diinginkan, maka kita harus menggabungkan terlebih dahulu *table pembeli* dan *pesanan* kemudian baru memilih kolom dan baris yang diinginkan. Penggabungan table ini biasa disebut **join**.

Untuk lebih mudah memahami bagaimana join dilakukan, perhatikan Gambar 18.14. Query untuk gambar tersebut berbunyi :

Tampilkan semua data pesanan untuk pembeli dengan nama Wayne Rooney.

Pada query ini kita dapat mengidentifikasi ada dua table yang akan terlibat yaitu table pembeli dan pesanan (perhatikan bagian from pada Gambar 18.14). Untuk menggabungkan kedua table tersebut, kita harus mempunyai penghubung antar kedua table. Pada Bab 17 telah dijelaskan bahwa table pembeli dan table pesanan dihubungkan dengan atribut *id_pembeli*. *Id_pembeli* merupakan primary key pada table pembeli dan foreign key pada table pesanan. Dengan menggunakan atribut ini kita bisa membuat join. Pada Gambar 18.14, pernyataan join diletakkan sebagai kondisi pada pernyataan **where** dengan menggunakan operator logika and.

```

File Edit View Terminal Tabs Help
mysql> select pesanan.id_pesanan, pesanan.jumlah_pembelian, pesanan.tanggal_pembelian
-> from pembeli, pesanan
-> where pembeli.nama = 'Wayne Rooney'
-> and pembeli.id_pembeli = pesanan.id_pembeli;
+-----+-----+-----+
| id_pesanan | jumlah_pembelian | tanggal_pembelian |
+-----+-----+-----+
| 3          | 160000.00        | 2007-07-25         |
+-----+-----+-----+
1 row in set (0.15 sec)

```

Pernyataan untuk join
2 tabel

Baris yang dipilih

Tabel yang dipilih
untuk digabung

Kolom yang akan
ditampilkan

Gambar 18.14. Penggabungan dua table.

Pada Gambar 18.14 juga tampak ada sedikit perbedaan dalam menuliskan kolom yang ingin ditampilkan. Kolom yang dipilih harus ditulis dengan nama tabelnya, misalnya **pesanan.id_pesanan**. Pesanan adalah nama tabelnya dan id_pesanan adalah nama kolomnya. Hal ini untuk menghindari kesalahan dari DBMS mengidentifikasi kolom dari table mana yang diinginkan.

Gambar 18.15. memperlihatkan contoh lain dari operasi join. Query untuk kasus ini adalah:

Tampilkan nama pembeli dan tanggal pembelian untuk pesanan yang jumlah pembeliannya lebih dari 100000.00.

```

File Edit View Terminal Tabs Help

mysql> select pembeli.nama, pesanan.tanggal_pembelian
-> from pembeli, pesanan
-> where pesanan.jumlah_pembelian > 100000.00
-> and pembeli.id_pembeli = pesanan.id_pembeli;
+-----+-----+
| nama          | tanggal_pembelian |
+-----+-----+
| Wayne Rooney  | 2007-07-25         |
| Cristiano Ronaldo | 2007-09-15         |
+-----+-----+
2 rows in set (0.00 sec)

```

Gambar 18.15. Contoh lain operasi join.

Gambar 18.14 dan 18.15 memperlihatkan contoh operasi join untuk 2 tabel. Cara yang sama dapat kita lakukan untuk operasi lebih dari 2 tabel. Perhatikan Gambar 18.16 berikut ini.

```

mysql> select buku.judul, buku.pengarang
-> from pesanan, item_pesanan, buku
-> where pesanan.id_pesanan = 3
-> and pesanan.id_pesanan = item_pesanan.id_pesanan
-> and item_pesanan.isbn = buku.isbn;

```

Tiga tabel yang akan digabung

Join tabel pesanan dengan item_pesanan

Join tabel item_pesanan dengan buku

judul	pengarang
Sejarah Sepakbola	Juande Ramos
Teori Sepakbola Pra Sejarah	Harry Redknapp

2 rows in set (0.00 sec)

Gambar 18.16. Operasi join tiga buah table.

Query untuk Gambar 18.16, dalam bahasa sehari-hari berbunyi:

Tampilkan judul buku dan pengarangnya yang id_pesanan nya adalah 3.

Query ini membutuhkan tiga buah table untuk memberikan hasil yang diharapkan, yaitu: *pesanan*, *item_pesanan* dan *buku* (lihat pada bagian from). Hubungan antara ketiga buah table tersebut adalah table pesanan – *item_pesanan* – *buku* (untuk lebih jelas, lihat kembali hubungan antara ketiga table ini pada Bab 17). *Tabel pesanan* dan *item_pesanan* dihubungkan dengan atribut *id_pesanan* dan *Tabel item_pesanan* dan *buku* dihubungkan dengan atribut *isbn*.

Gambar 18.17 memperlihatkan contoh yang lebih rumit, karena menggabungkan semua table yang ada dalam *Basis Data Penjualan Buku*. Dalam bahasa sehari-hari query ini akan berbunyi:

Tampilkan nama pembeli yang memesan judul buku yang ada kata “Sepakbola Modern” dan tanggal pembeliannya.

Seperti halnya contoh pada Gambar 18.16, kita harus melihat lebih dulu bagaimana table-table tersebut berhubungan dan atribut apa yang menghubungkan table-table tersebut.

Pada Gambar 18.17 juga diperkenalkan penggunaan pernyataan *Like*. Pernyataan *Like* dapat kita gunakan untuk tipe data *char*, *text*, *string* atau yang sejenisnya. Pernyataan **where buku.judul like '%Sepakbola Modern%'** berarti kita mencari baris yang kolom judulnya berisi teks yang ada kata 'Sepakbola Modern'. Kalau kita lihat isi data pada *table buku* (Gambar 18.9) ada satu baris yang sesuai dengan kondisi ini yaitu pada judul buku '*Teori Sepakbola Modern*'. Bagaimana jika pernyataan kita rubah menjadi **where buku.judul like '%Sejarah%'**? Untuk kasus ini maka akan terpilih dua record pada *table buku* yaitu yang judulnya '*Teori*

Sepakbola Pra Sejarah dan *Sejarah Sepakbola*. Kita dapat menggunakan pernyataan Like dengan berbagai variasi tergantung kebutuhan.

```
File Edit View Terminal Tabs Help
mysql> select pembeli.nama, pesanan.tanggal_pembelian
-> from pembeli, pesanan, item_pesanan, buku
-> where buku.judul like '%Sepakbola Modern%'
-> and pembeli.id_pembeli = pesanan.id_pembeli
-> and pesanan.id_pesanan = item_pesanan.id_pesanan
-> and item_pesanan.isbn = buku.isbn;
+-----+-----+
| nama          | tanggal_pembelian |
+-----+-----+
| Cristiano Ronaldo | 2007-09-15         |
| Ryan Giggs      | 2007-04-12         |
+-----+-----+
2 rows in set (0.00 sec)
```

Gambar 18.17. Operasi join empat buah table.

18.2.6 Update Data

Seringkali dalam pengisian data pada suatu table, kita melakukan kesalahan. Kesalahan dapat berupa kesalahan ketik atau kesalahan pembacaan data. Sehingga ketika diperiksa, kita menginginkan untuk merubah/memperbaiki data tersebut. Proses ini biasa disebut sebagai update data. SQL menyediakan perintah **update** untuk melakukan proses ini. Perintah ini masih termasuk dalam kelompok DML.

Gambar 18.18 memperlihatkan bagaimana proses update dilakukan. Pada gambar ini kita ingin merubah data yang ada pada record no 2 saja. Pada table awal (bagian atas), record no 2 berisi alamat 'Jl. Simpang Manchester II / 10 Manchester'. Table yang akan kita update adalah *table pembeli* sehingga setelah perintah **update**, kita tuliskan nama *table pembeli*. Untuk merubah data, kita cukup merujuk pada kolom yang ingin kita ganti saja. Kemudian kita ganti isinya dengan data yang baru dengan perintah **set**. Hasil perubahan dapat dilihat pada table akhir (bagian bawah). Kolom alamat pada record no 2 telah berganti isi datanya.


```
File Edit View Terminal Tabs Help
mysql> select * from pembeli;
+-----+-----+-----+-----+
| id_pembeli | nama          | alamat                                     | telepon |
+-----+-----+-----+-----+
| 1          | Cristiano Ronaldo | Jl. Manchester Raya No. 10, Manchester | 0808000000 |
| 2          | Ryan Giggs       | Jl. Simpang Manchester II / 10, Manchest | 0801000000 |
| 3          | Wayne Rooney    | Jl. Simpang Manchester III/10, Mancheste | 0802000000 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> update pembeli
-> set alamat = "Jl. Goodison No. 123, London"
-> where id_pembeli = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from pembeli;
+-----+-----+-----+-----+
| id_pembeli | nama          | alamat                                     | telepon |
+-----+-----+-----+-----+
| 1          | Cristiano Ronaldo | Jl. Manchester Raya No. 10, Manchester | 0808000000 |
| 2          | Ryan Giggs       | Jl. Goodison No. 123, London           | 0801000000 |
| 3          | Wayne Rooney    | Jl. Simpang Manchester III/10, Mancheste | 0802000000 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Gambar 18.18. Penggunaan perintah update.

18.2.7 Menghapus record, table dan basis data

SQL menyediakan fasilitas menghapus baik *record*, *table* maupun *database*. Perintah menghapus record adalah **delete** (Gambar 18.19), menghapus table adalah **drop table** (Gambar 18.20) dan menghapus basis data adalah **drop database** (Gambar 18.21). **Delete** termasuk kelompok DML, sedangkan **drop table** dan **drop database** termasuk dalam DDL.

```
Command Prompt - mysql
mysql> select * from pembeli;
+-----+-----+-----+-----+
| id_pembeli | nama          | alamat                                     | telepon |
+-----+-----+-----+-----+
| 1          | Cristiano Ronaldo | Jl. Manchester Raya No. 10, Manchester | 0808000000 |
| 2          | Ryan Giggs       | Jl. Simpang Manchester II/10, Manchester | 0801000000 |
| 3          | Wayne Rooney    | Jl. Simpang Manchester III/10, Manchester | 0802000000 |
| 4          | Garry Neville   | Jl. Raya Bobby Robson No 15, Manchester | 0809000000 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> delete from pembeli
-> where id_pembeli = 4;
Query OK, 1 row affected (0.16 sec)

mysql> select * from pembeli;
+-----+-----+-----+-----+
| id_pembeli | nama          | alamat                                     | telepon |
+-----+-----+-----+-----+
| 1          | Cristiano Ronaldo | Jl. Manchester Raya No. 10, Manchester | 0808000000 |
| 2          | Ryan Giggs       | Jl. Simpang Manchester II/10, Manchester | 0801000000 |
| 3          | Wayne Rooney    | Jl. Simpang Manchester III/10, Manchester | 0802000000 |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Gambar 18.19. Penggunaan perintah delete.

```
Command Prompt - mysql

mysql> show tables;
+-----+
| Tables_in_latihan03 |
+-----+
| buku                |
| item_pesanan        |
| pembeli             |
| pesanan             |
+-----+
4 rows in set (0.00 sec)

mysql> drop table item_pesanan;
Query OK, 0 rows affected (0.16 sec)

mysql> show tables;
+-----+
| Tables_in_latihan03 |
+-----+
| buku                |
| pembeli             |
| pesanan             |
+-----+
3 rows in set (0.00 sec)
```

Gambar 18.20. Penggunaan perintah drop table.

```
Command Prompt - mysql

mysql> show databases;
+-----+
| Database |
+-----+
| latihan01 |
| latihan02 |
| latihan03 |
| mysql    |
| test     |
+-----+
5 rows in set (0.00 sec)

mysql> drop database latihan03;
Query OK, 3 rows affected (0.08 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| latihan01 |
| latihan02 |
| mysql    |
| test     |
+-----+
4 rows in set (0.00 sec)
```

Gambar 18.21. Penggunaan perintah drop database.

18.2.8 Menggunakan opsi dan perintah tambahan

Perintah-perintah SQL yang telah kita gunakan di atas merupakan perintah-perintah standar yang sering digunakan. Namun masih ada banyak perintah dalam SQL yang dapat digunakan. Biasanya perintah-perintah ini tergantung pada DBMS yang digunakan. Masing-masing DBMS mempunyai daftar perintah yang sedikit berbeda dengan yang lainnya. Pada bagian berikut akan ditunjukkan beberapa opsi dan perintah tambahan yang disediakan secara umum oleh hampir sebagian besar DBMS.

- **Mengurutkan tampilan data**

Pada sub bab 18.2.5, telah dibahas bagaimana menampilkan data dengan perintah select, namun tampilan data biasanyaurut berdasarkan primary key – nya. Kita dapat mengatur urutan tampilan dengan menambahkan opsi **order by**. Perhatikan contoh pada Gambar 18.22. Pada gambar ini kita menggunakan table pesanan sebagai contoh. Urutan asli tampilan data dapat dilihat pada Gambar 18.9. Kita dapat mengurutkan berdasarkan kolom yang kita pilih. Pada Gambar 18.22 bagian atas kita menggunakan kolom jumlah_pembelian sebagai dasar pengurutan dan dengan opsi **order by** secara default. Tampilan yang dihasilkan adalah data terurut mulai dari jumlah_pembelian terkecil menuju ke yang paling besar. Untuk mengurutkan dari besar ke kecil kita menggunakan tambahan opsi **desc**.

Command Prompt - mysql

```
mysql> select * from pesanan order by jumlah_pembelian;
```

id_pesanan	id_pembeli	jumlah_pembelian	tanggal_pembelian
4	2	55000.00	2007-08-10
1	2	65000.00	2007-04-12
2	1	75000.00	2007-07-22
5	1	120000.00	2007-09-15
3	3	160000.00	2007-07-25

5 rows in set (0.14 sec)

```
mysql> select * from pesanan order by jumlah_pembelian desc;
```

id_pesanan	id_pembeli	jumlah_pembelian	tanggal_pembelian
3	3	160000.00	2007-07-25
5	1	120000.00	2007-09-15
2	1	75000.00	2007-07-22
1	2	65000.00	2007-04-12
4	2	55000.00	2007-08-10

5 rows in set (0.00 sec)

Tampilan jumlah_pembelian urut dari kecil ke besar

Tampilan jumlah_pembelian urut dari besar ke kecil

Gambar 18.22. Mengurutkan tampilan dengan opsi order by.

- **Menggunakan pernyataan min, max, sum, avg**

SQL juga menyediakan pernyataan-pernyataan yang berhubungan dengan operasi matematika, meskipun terbatas. Pernyataan min berguna untuk mencari nilai data numeric yang paling kecil. Sedangkan pernyataan max sebaliknya. Pernyataan sum digunakan untuk mencari total nilai numerik dari kolom yang dipilih. Sedangkan pernyataan avg untuk mencari nilai rata-ratanya. Gambar 18.23 dan Gambar 18.24 memperlihatkan contoh bagaimana menggunakan pernyataan-pernyataan tersebut.

Command Prompt - mysql

```
mysql> select * from pesanan;
```

id_pesanan	id_pembeli	jumlah_pembelian	tanggal_pembelian
1	2	65000.00	2007-04-12
2	1	75000.00	2007-07-22
3	3	160000.00	2007-07-25
4	2	55000.00	2007-08-10
5	1	120000.00	2007-09-15

5 rows in set (0.00 sec)

```
mysql> select min(jumlah_pembelian), max(jumlah_pembelian) from pesanan;
```

min(jumlah_pembelian)	max(jumlah_pembelian)
55000.00	160000.00

1 row in set (0.00 sec)

Gambar 18.23. Penggunaan pernyataan min dan max.

```
mysql> select sum(jumlah_pembelian) from pesanan;
```

sum(jumlah_pembelian)
475000.00

```
1 row in set (0.00 sec)
```

```
mysql> select avg(jumlah_pembelian) from pesanan;
```

avg(jumlah_pembelian)
95000.000000

```
1 row in set (0.00 sec)
```

Gambar 18.24. Penggunaan pernyataan sum dan avg.

- **Menghitung banyaknya record.**

Kadang-kadang kita tidak mengetahui berapa banyak record data dari suatu table atau suatu hasil query yang lain. Padahal kita diminta oleh seseorang untuk menginformasikan hal tersebut sesegera mungkin. Kalau kita menghitung satu persatu mungkin akan memakan waktu panjang. Pada SQL disediakan pernyataan `count` untuk menangani hal ini. Perhatikan contoh pada Gambar 18.25.

```
mysql> select * from item_pesanan;
```

id_pesanan	isbn	jumlah
1	0-672-31667-9	1
2	0-672-31667-8	1
3	0-672-31637-9	1
3	0-672-31337-3	1
4	0-672-31637-9	1
5	0-672-31637-9	1
5	0-672-31667-9	1

```
7 rows in set (0.00 sec)
```

```
mysql> select count(*) from item_pesanan;
```

count(*)
7

```
1 row in set (0.00 sec)
```

Gambar 18.25. Penggunaan pernyataan count.

- **Mengelompokkan data**

Data hasil query biasanya ditampilkan satu-per-satu record. Namun suatu saat kita mungkin ingin menampilkan data tersebut dalam kelompok-kelompok berdasarkan kriteria tertentu. SQL menyediakan pernyataan `group by`. Pada Gambar 18.26 diperlihatkan bagaimana menggunakan perintah `group by`.

mysql> select pengarang from buku;

pengarang
Arsene Wenger
Harry Redknapp
Harry Redknapp
Juande Ramos

4 rows in set (0.00 sec)

mysql> select count(pengarang) from buku;

count(pengarang)
4

1 row in set (0.00 sec)

mysql> select pengarang, count(*) from buku group by pengarang;

pengarang	count(*)
Arsene Wenger	1
Harry Redknapp	2
Juande Ramos	1

3 rows in set (0.00 sec)

Gambar 18.26. Penggunaan pernyataan group by.

Dasar-Dasar Pemrograman Web

TUJUAN

Setelah anda mempelajari bab ini, diharapkan anda akan mampu :

- Menjelaskan pengertian intranet, extranet dan internet.
 - Menjelaskan pengertian web dan pemrograman web.
 - Memahami pengertian client-side, server-side, web statis dan web dinamis.
 - Menyebutkan jenis-jenis perangkat lunak yang dibutuhkan dalam pemrograman web.
 - Menyebutkan jenis-jenis bahasa pemrograman berbasis web.
-

19.1 INTRANET, EKSTRANET DAN INTERNET

Internet atau *International Networking* :

Dua komputer atau lebih yang saling berhubungan membentuk jaringan komputer hingga meliputi jutaan komputer di dunia (internasional), yang saling berinteraksi dan bertukar informasi dengan menggunakan standar protocol tertentu.

Dari segi ilmu pengetahuan, Internet merupakan sebuah perpustakaan besar yang didalamnya terdapat jutaan (bahkan milyaran) informasi atau data yang dapat berupa text, graphic, audio maupun animasi, dan lain-lain dalam bentuk media elektronik. Orang bisa "berkunjung" ke perpustakaan tersebut kapan saja dan dari mana saja. Dari segi komunikasi, Internet adalah sarana yang sangat efisien dan efektif untuk melakukan pertukaran informasi jarak jauh, maupun di dalam lingkungan perkantoran.

Intranet :

Jaringan komputer terbatas yang menggunakan protokol internet yang digunakan secara internal dalam suatu organisasi.

Extranet :

Jaringan komputer yang juga menggunakan protokol internet namun lebih luas jangkauannya dari pada intranet.

Kadang-kadang orang mengartikan intranet sebagai website internal. Sebagai contoh jaringan intranet pada suatu sekolah. Pada extranet selain akses informasi di dalam internal organisasi, juga dapat dilakukan akses informasi antar organisasi yang sudah terhubung. Sebagai contoh jaringan ekstranet antara perusahaan dengan suppliernya.

19.2 WEB DAN PEMROGRAMAN WEB

World Wide Web (WWW) :

Kumpulan dokumen multimedia (gambar/teks/link) yang menyediakan informasi, jasa dan situs dengan dukungan internet.

World Wide Web secara luas lebih dikenal dengan istilah Web. Web pertama kali diperkenalkan pada tahun 1992. Hal ini sebagai hasil usaha pengembangan yang dilakukan CERN di Swiss.

Dengan demikian, internet dan web adalah dua hal yang berbeda. Internet lebih merupakan perangkat keras, sedangkan web adalah perangkat lunak. Selain itu, protokol yang dipakai oleh keduanya juga berbeda. Internet menggunakan TCP/IP sebagai protokol operasionalnya, sedangkan web menggunakan HTTP (Hyper Text Transfer Protocol).

Pemrograman web :

Merupakan usaha untuk membuat halaman web dengan menggunakan bahasa pemrograman web (script).

Pemrograman web (*web programming*) dikenal juga dengan istilah pengembangan web (*web development*). Istilah lain yang mungkin juga cukup

terkenal adalah web design. Web design lebih memfokuskan bagaimana merancang tampilan halaman-halaman web menjadi menarik bila dilihat. Atau boleh dikatakan web design lebih pada aspek visualnya sedangkan web programming lebih fokus pada aspek logika proses yang terjadi di dalam halaman-halaman web. Seorang web programmer mungkin juga seorang web designer, namun biasanya pekerjaan ini dipisahkan orangnya karena perbedaan fokus penekanannya.

19.3 CLIENT-SIDE DAN SERVER-SIDE

Untuk membuat interaksi yang baik, dibutuhkan beberapa model pemrograman web. Model pemrograman yang umum digunakan adalah *client-side* dan *server-side*. Client dan server dalam kasus ini menunjukkan dua tempat yang berbeda.

Server :

Komputer yang bertindak sebagai pihak yang melayani permintaan data atau informasi.

Client :

Komputer pengguna yang hendak mengakses program ke server untuk meminta data atau informasi dengan menggunakan alamat yang unik.

Biasanya perangkat komputer yang bertindak sebagai server akan berjalan terus-menerus tanpa henti dan berperan sebagai tempat dimana file/program dari aplikasi web ditempatkan. Secara umum, untuk dapat mengakses layanan ke server dibutuhkan *browser* seperti Internet Explorer (IE), Opera, Mozilla dan Netscape.

Pada server-side, segala proses akan dilakukan di web-server sedangkan client hanya akan menerima hasil olahan dari web-server yaitu berupa halaman-halaman dalam format HTML. Bahasa pemrograman yang digunakan dalam membuat aplikasi pada sisi server biasa disebut sebagai server-side script.

Ada beberapa keuntungan pada penggunaan server-side script, yaitu :

- *Keamanan kode*

Script yang kita buat tidak akan dapat dibaca oleh user karena seluruhnya akan disimpan dan dijalankan di web server. Yang akan dikirim ke client (pengguna) adalah hasil pengolahan yang berupa dokumen dalam format HTML saja.

- *Koneksi dengan database*

Kemampuan koneksi dengan database merupakan keuntungan terbesar dari server-side script. Dengan kemampuan ini informasi yang dihasilkan oleh web server menjadi sangat dinamis bukan lagi halaman-halaman statis. Informasi yang disampaikan ke user tergantung dari apa yang diinginkan oleh user tersebut.

- *Dapat melakukan tracking pengguna*

Dengan server-side script, dapat diketahui siapakah user yang sedang akses ke aplikasi web dengan menggunakan fasilitas session.

Contoh penggunaan server-side :

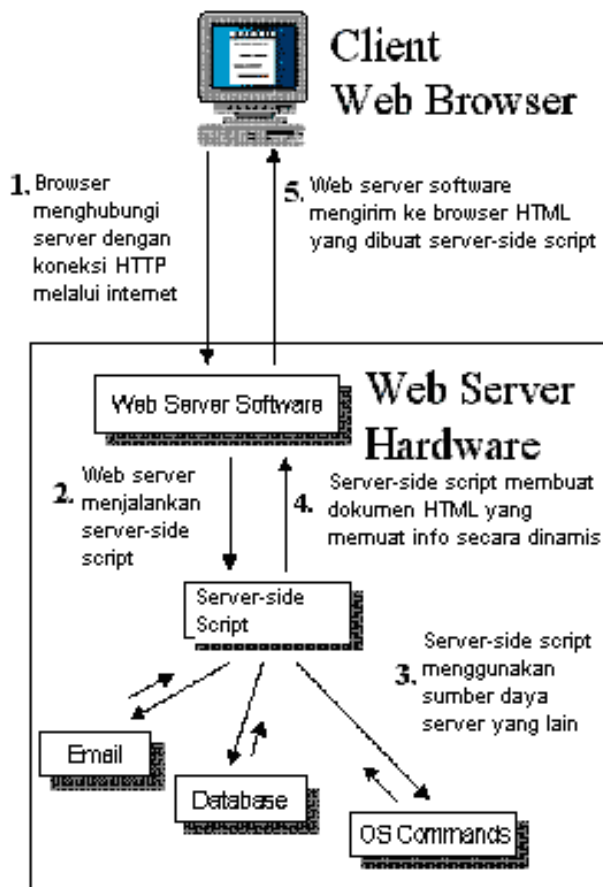
- Server dapat menentukan time-out (batas waktu) web jika pengguna telah login pada suatu aplikasi tetapi web tidak diakses.
- Pengguna meminta data dari database melalui aplikasi web.

Pada client-side, proses akan dilakukan sepenuhnya pada komputer client. Biasanya client-side digunakan pada hal-hal yang membutuhkan banyak interaksi dan menggunakan jenis informasi yang pasti dan beragam. Bahasa pemrograman yang digunakan dalam membuat aplikasi pada sisi client biasa disebut sebagai client-side script. Client-side script yang umum digunakan adalah HTML, JavaScript, dan VBScript. Keuntungan utama dari client-side script adalah waktu prosesnya yang jauh lebih cepat dibanding server-side. Hal ini karena seluruh permintaan pengguna akan diproses pada komputer pengguna sendiri. Namun masalah terbesar dari aplikasi dengan client-side script adalah keamanan kode dan data. Hal karena pengguna dapat dengan mudah membuka dan melihat kode program.

Contoh penggunaan client-side script adalah :

- aplikasi web untuk kalkulator, tanggal atau permainan
- pemeriksaan event pada browser, jika mouse diklik kanan maka akan muncul keterangan yang dibutuhkan
- validasi isi form yang diinputkan oleh pengguna, sebelum isi form tersebut dikirim ke server

Server-side dan client-side tampaknya seperti dua sisi yang berlawanan, namun sesungguhnya kedua hal ini bersifat saling melengkapi. Penggunaan dua tipe script ini harus disesuaikan dengan aplikasi yang dibangun. Artinya, mana proses yang harus dikerjakan di sisi server dan mana yang di sisi client harus ditentukan dengan hati-hati. Karena apabila tidak, tujuan dari pengembangan aplikasi web mungkin tidak akan tercapai. Secara umum interaksi dan pertukaran data antara client dan server dalam internet tampak pada gambar 19.1.



Gambar 19.1. Client side dan server side.

19.4 PENGERTIAN WEB STATIS DAN DINAMIS

Pengertian web statis dan web dinamis seringkali mengundang perdebatan. Sebagian pengguna internet menyatakan jika pada halaman-halaman web dilengkapi dengan animasi yang bergerak maka disebut web dinamis sedangkan jika halaman-halaman web tersebut hanya berisi teks dan gambar yang tidak bergerak maka disebut web statis. Namun berdasarkan kesepakatan maka pengertian statis dan dinamis tidak ditentukan oleh ada atau tidaknya animasi bergerak pada halaman-halaman web, tetapi ditentukan oleh isi atau informasi yang ada pada halaman-halaman tersebut.

Web statis :

Halaman-halaman web yang isi atau informasinya tidak berubah-ubah.

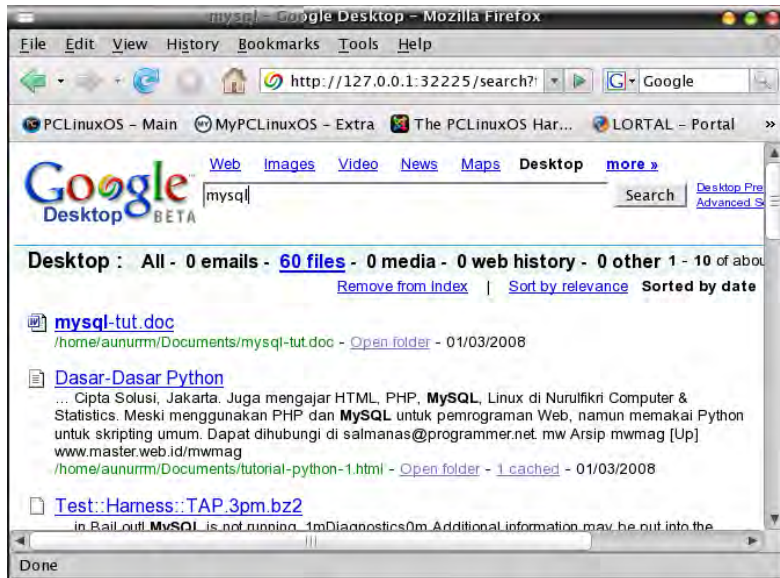
Web dinamis :

Halaman-halaman web yang isi atau informasinya berubah-ubah bergantung input atau permintaan pengguna.

Dengan melihat definisi di atas maka halaman-halaman web yang tidak dilengkapi animasi bergerak tetapi isi atau informasinya mampu berubah-ubah sesuai keinginan pengguna digolongkan sebagai web dinamis demikian pula sebaliknya. Contoh paling mudah adalah bila kita membuka situs Google, maka halaman awal (lihat Gambar 19.2) adalah statis karena kita tidak melihat perubahan isi atau informasi. Namun begitu kita memasukkan kata pada textbox yang tersedia kemudian menekan tombol search maka kita sedang berinteraksi dengan web server Google. Tampilan halaman berikutnya akan sangat tergantung pada apa yang kita inputkan (lihat Gambar 19.3). Pada bagian ini kita berhadapan dengan halaman-halaman web dinamis. Biasanya halaman-halaman web yang statis dibuat hanya dengan menggunakan client side script (sebagian besar hanya dengan script HTML), sedangkan web dinamis membutuhkan server side script.



Gambar 19.2. Halaman awal situs Google.



Gambar 19.3. Halaman web dinamis.

19.5 PERANGKAT LUNAK UNTUK PEMROGRAMAN WEB

Selain perangkat keras yaitu komputer dan perangkat jaringan maka pemrograman web membutuhkan dukungan perangkat lunak yang sesuai. Secara umum ada lima kelompok perangkat lunak yang dibutuhkan yaitu sistem operasi, web server, perangkat pengembang web, sistem manajemen basis data dan web browser.

19.5.1 Sistem Operasi

Sistem operasi pada server memegang peranan penting dalam pemrograman web karena pada sistem operasi itu akan ditanamkan (diinstal) web server, perangkat pengembang, sistem manajemen basis data dan bahasa pemrograman. Artinya pilihan pada sistem operasi tertentu akan menentukan pula pilihan web server, perangkat pengembang dan bahasa pemrograman yang akan digunakan. Hal ini dikarenakan adanya masalah kompatibilitas antar perangkat lunak. Sebagai contoh, apabila kita memilih menggunakan sistem operasi Linux maka kita tidak dapat menginstal IIS sebagai web server.

Selain masalah kompatibilitas, hal lain yang juga perlu dipertimbangkan dalam penentuan sistem operasi yang akan kita gunakan pada server adalah masalah keamanan, stabilitas, kemudahan konfigurasi. Keamanan berhubungan dengan kemampuan sistem operasi untuk melindungi diri dari serangan virus, spam, atau kode-kode jahat yang sengaja

disusupkan. Kemampuan ini sangat penting diperhatikan karena lalu lintas data dalam internet sangat rentan terhadap gangguan virus, spam, dan pengganggu lainnya. Stabilitas berhubungan dengan kemampuan system operasi untuk bekerja terus-menerus untuk merespon permintaan client. Kemudahan konfigurasi berhubungan mudah tidaknya konfigurasi dilakukan terhadap system operasi dalam perannya sebagai server.

Saat ini ada dua pemain besar dalam sistem operasi untuk server yaitu keluarga Unix/Linux dan Windows. Linux dikenal relative lebih aman dan stabil dari pada Windows. Hal ini karena Linux dan keluarganya mewarisi kekuatan Unix dalam jaringan (TCP/IP). Namun Windows memiliki kelebihan dari sisi kemudahan konfigurasi. Biasanya pembuat sistem operasi akan menyediakan versi yang memang khusus diperuntukkan bagi server. Versi ini biasanya memiliki tingkat keamanan dan stabilitas yang lebih tinggi dari versi desktop-nya. Sebagai contoh, pada Linux ada distro Ubuntu versi server, SuSe menyediakan versi Enterprise Server, dan demikian juga distro-distro lainnya. Sedangkan pada Windows tersedia Windows NT, Windows Server 2000, Windows Server 2003.

19.5.2 Web Server

Seperti telah disebutkan sebelumnya, web server adalah perangkat lunak yang bertindak melayani permintaan-permintaan client terhadap halaman-halaman web tertentu. Saat ini ada dua pemain utama pada web server, yaitu Apache dan IIS (Internet Information Service). Sampai dengan Desember 2007, wikipedia mencatat Apache berada dalam posisi pertama sebagai web server yang paling banyak digunakan, disusul IIS.

- Apache

Penggunaan nama Apache sebenarnya memiliki dua alasan. Alasan yang pertama sebagai penghormatan pada salah satu suku asli Indian Amerika. Suku ini dikenal dengan ketangguhan dan keahliannya dalam berperang. Diharapkan web server Apache akan berkarakter seperti itu, yaitu tangguh, handal, dan ahli dalam perannya sebagai web server. Alasan yang kedua, sebenarnya asal muasal proyek Apache adalah membuat sekumpulan patch (perbaikan) bagi kode NCSA HTTPd 1.3 agar menjadi 'a patchy' server. Untuk lebih mudah, pembacaan 'a patchy' kemudian dieja menjadi Apache.

Apache dikembangkan oleh komunitas pengembang software terbuka di bawah organisasi yang disebut Apache Software Foundation. Apache disebarkan sebagai open source software. Saat ini, Apache telah tersedia untuk hampir seluruh sistem operasi, termasuk Microsoft Windows, Unix, Novel Netware, Mac OS X dan keluarga Linux. Bahkan pada Mac OS X dan keluarga Linux, Apache adalah web server default. Selain itu pada perangkat-perangkat pengembang web seperti IBM

WebSphere, Borland Delphi, dan Oracle Database, Apache termasuk dalam paket perangkat lunak tersebut.

Apache dapat digunakan baik untuk web statis maupun web dinamis dan mensupport banyak bahasa pemrograman, antara lain Perl, Python dan tentu saja PHP. Fungsi-fungsi keamanan web juga dikendalikan dengan sangat baik. Dukungan pada koneksi dengan berbagai basis data, seperti MySQL, SQL Lite, PostgreSQL, Oracle, DB2 dan lain-lain dapat dilakukan dengan baik.

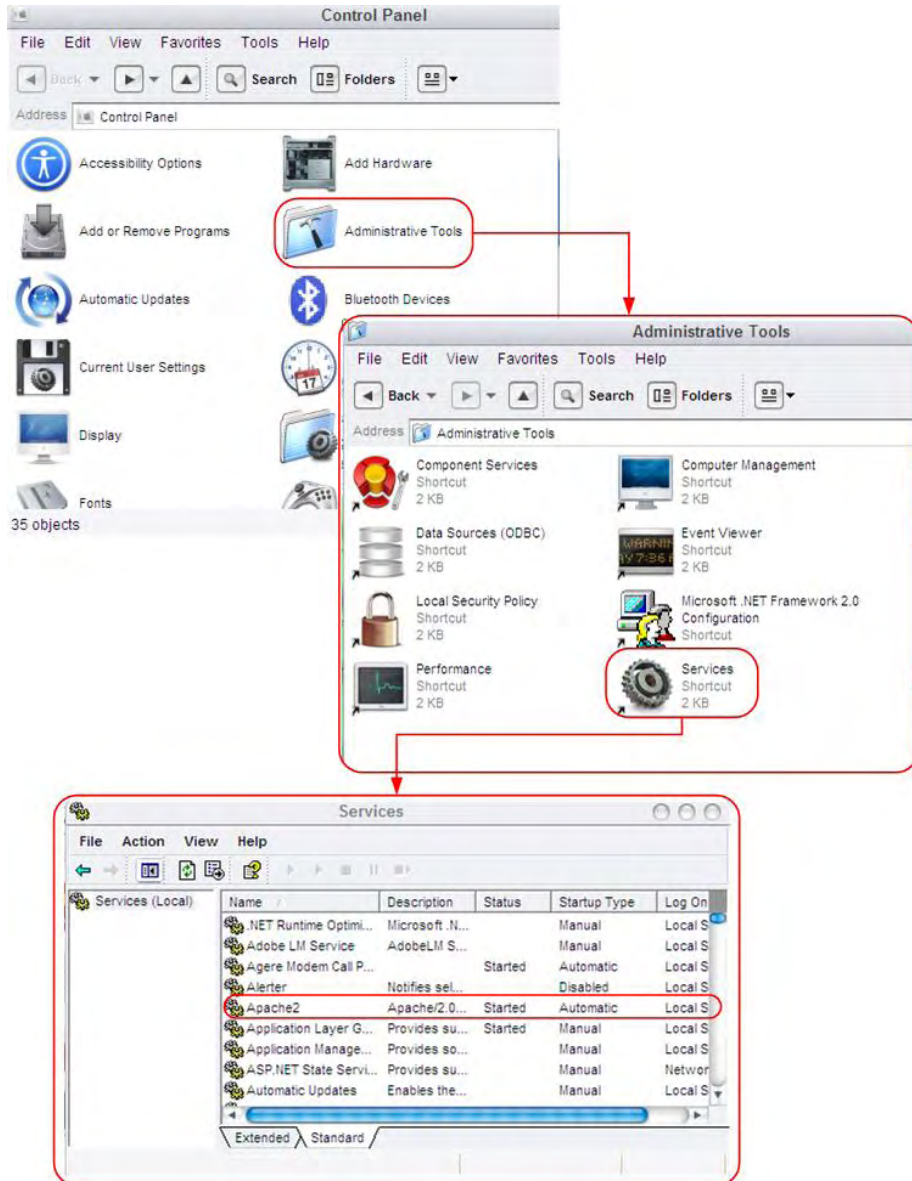
Untuk dapat menggunakan Apache maka kita harus memastikan bahwa Apache sudah terinstall dengan baik. Pada semua sistem operasi yang didukung, biasanya Apache akan dijalankan sebagai salah satu service. Kita dapat memeriksa apakah ada service bernama Apache atau Httpd yang dijalankan oleh sistem operasi tersebut pada daftar service. Pada keluarga Linux kita dapat memeriksa baik menggunakan mode CLI maupun GUI. Gambar 19.4 menunjukkan bagaimana kita memeriksa service-service yang dijalankan oleh Linux (distro Mandriva), termasuk Httpd.



Gambar 19.4. Memeriksa service Apache pada Linux.

Pada Microsoft Windows, kita dapat memeriksa dengan cara sebagai berikut: buka Control Panel kemudian double click pada icon Administrative Tools (Gambar 19.5.). Setelah jendela Administrative Tools terbuka, double click pada icon Services. Pada jendela Services, cari apakah ada nama Apache pada daftar service. Jika sudah ada dan

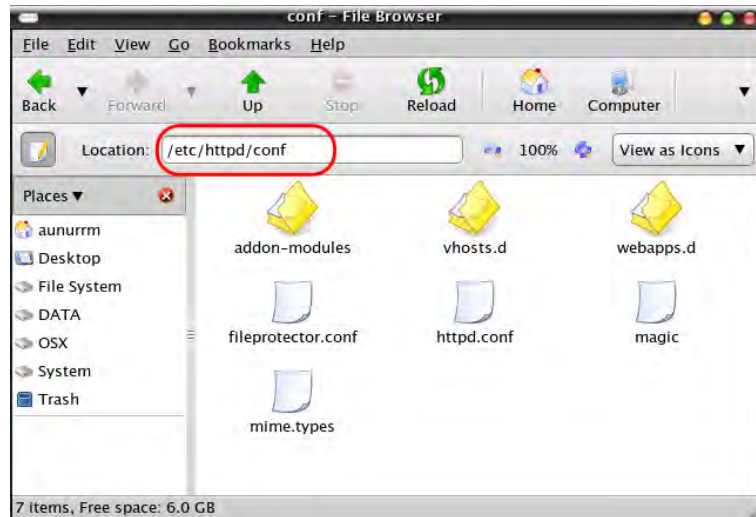
statusnya Started, berarti Apache telah terinstall dengan benar dan dapat kita gunakan. Jika belum ada, maka kita harus menginstallnya.



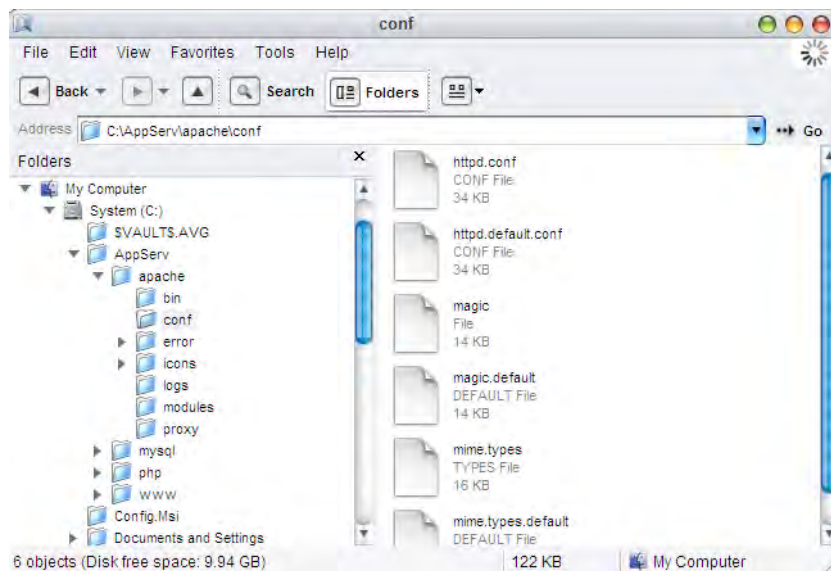
Gambar 19.5. Memeriksa service Apache pada Microsoft Windows.

Konfigurasi Apache diatur dengan menggunakan file-file teks biasa. Biasanya bernama `httpd.conf` atau `apache.conf`. Pada Linux biasanya file ini terletak pada direktori `/etc/httpd/` (lihat Gambar 19.6), sedangkan pada Microsoft Windows tergantung pada instalasinya. Pada Gambar 19.7,

terlihat lokasi dimana file konfigurasi httpd.conf berada. Untuk membuka file konfigurasi ini kita dapat menggunakan teks editor seperti Notepad (Microsoft Windows), Nano, Pico, Joe, atau Vi (pada keluarga Linux).



Gambar 19.6. Lokasi file konfigurasi Apache pada Linux (distro Mandriva).



Gambar 19.7. Lokasi file konfigurasi Apache pada Microsoft Windows (menggunakan AppServ).

Ada banyak hal yang bisa kita atur namun secara umum ada beberapa bagian penting yang harus diatur, yaitu:

Section 1. Global Configuration. Bagian ini berisi konfigurasi umum Apache, seperti, lokasi *ServerRoot*, penentuan port dimana Apache akan dijalankan, dan modul-modul tambahan yang dijalankan.

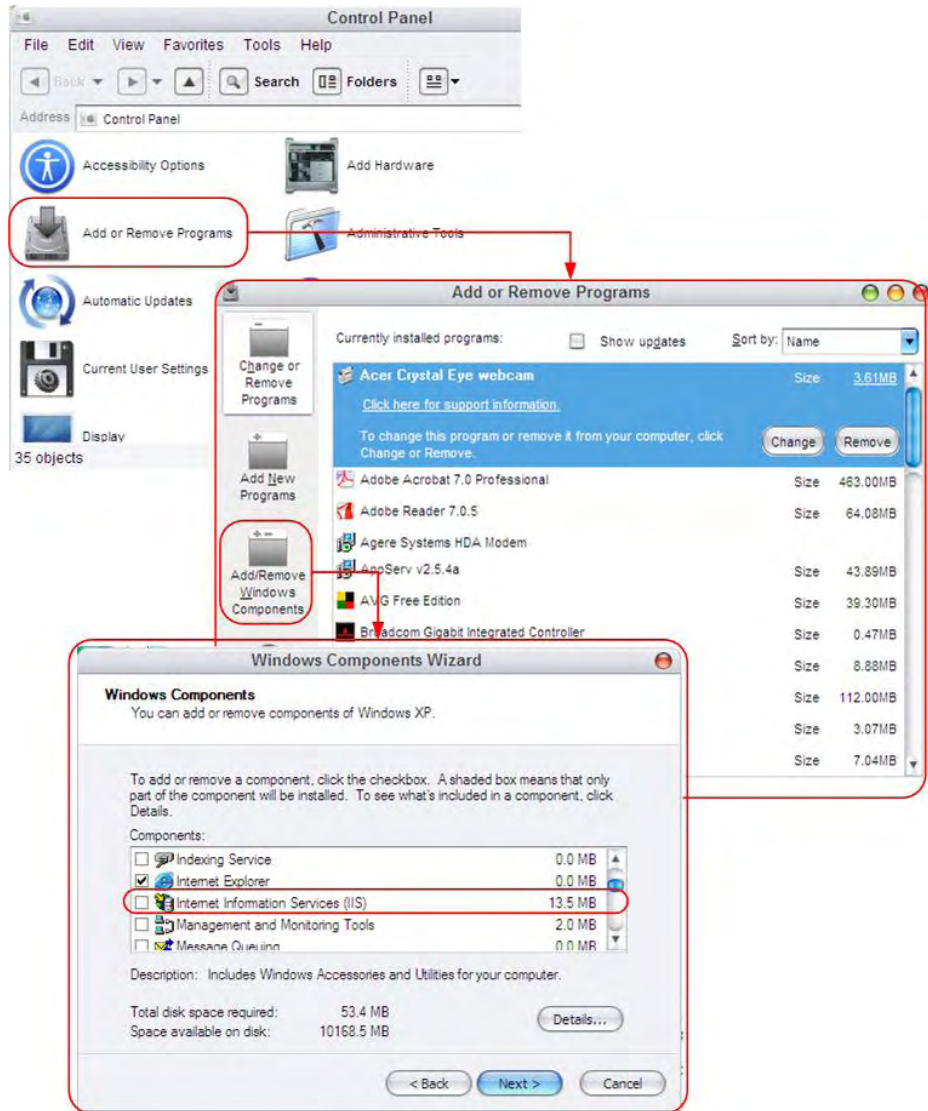
Section 2. Main Server Configuration. Bagian ini berisi konfigurasi utama server Apache, seperti, nama server, lokasi direktori dokumen yang dapat diakses oleh Apache, dokumen index direktori dan pengaturan lain yang berhubungan dengan karakteristik dokumen.

Section 2. Virtual Host. Bagian ini berisi konfigurasi yang dibutuhkan untuk mendefinisikan domain atau hostname bila ada lebih dari satu *domain/hostname*. Apache mempunyai kemampuan untuk membuat banyak *domain/hostname* pada satu web server.

- IIS

IIS adalah web server keluaran Microsoft. Sebutan web server bagi IIS mungkin tidak terlalu tepat, karena selain web server, IIS juga memberikan fasilitas *file server*, *email server* dan layanan lain berbasis internet. Oleh karena itu istilah yang tepat mungkin adalah *internet based-service*.

Perangkat lunak ini dibundel dalam sistem operasi Microsoft Windows. Namun secara default tidak langsung diinstall. Sehingga kalau kita mau menggunakannya kita harus menginstall lebih dulu. Untuk menginstall IIS pada Microsoft Windows, buka *Control Panel* kemudian pilih *Add or Remove Program*. Pada jendela *Add or Remove Program* pilih *Add/Remove Windows Component*. Setelah jendela *Windows Components Wizard* terbuka, cari *Internet Information Services (IIS)* di kotak *Components*. Untuk lebih jelas dapat dilihat pada Gambar 19.8. Jika IIS sudah terpilih berarti sudah terinstall, jika belum maka pilih IIS kemudian click *Next*. Untuk instalasi IIS ini kita membutuhkan CD Microsoft Windows.



Gambar 19.8. Memeriksa dan menginstal IIS.

19.5.3 Perangkat Pengembang Web

Perangkat pengembang web yang dimaksud disini adalah perangkat lunak yang digunakan untuk membuat halaman-halaman web, baik yang bersifat statis maupun dinamis. Di pasar perangkat lunak, saat ini tersedia banyak sekali jenis perangkat pengembang web, mulai dari yang sederhana sampai yang canggih dan complex. Namun sebenarnya untuk membuat halaman web baik statis maupun dinamis kita dapat menggunakan teks editor biasa. Hanya saja teks editor tidak menyediakan fasilitas-fasilitas yang

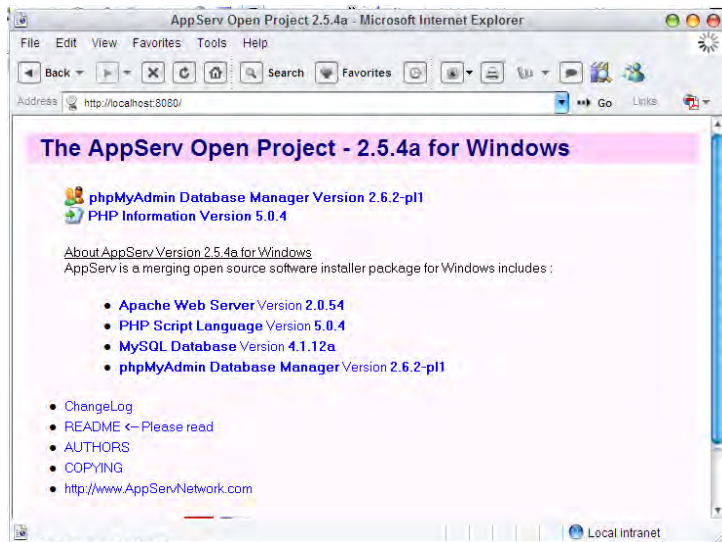
memudahkan kita dalam membuat halaman web. Pada perangkat pengembang web yang lebih kompleks seperti Adobe Dreamweaver (dulu Macromedia Dreamweaver), Microsoft Visual Studio .Net, dan beberapa yang lainnya, kita akan mendapati fasilitas yang sangat membantu mempercepat pembuatan halaman web, antara lain: tampilan berbasis GUI, *automatic code completion* (melengkapi kode secara otomatis), WYSIWYG (*What You See Is What You Get*) HTML Editor, koneksi ke basis data yang lebih mudah, dan banyak lagi fasilitas. Tentu saja perangkat pengembang ini berharga relative mahal. Penggunaan teks editor untuk membuat halaman-halaman web dapat dilihat pada Bab 20.

19.5.4 Sistem Manajemen Basis Data

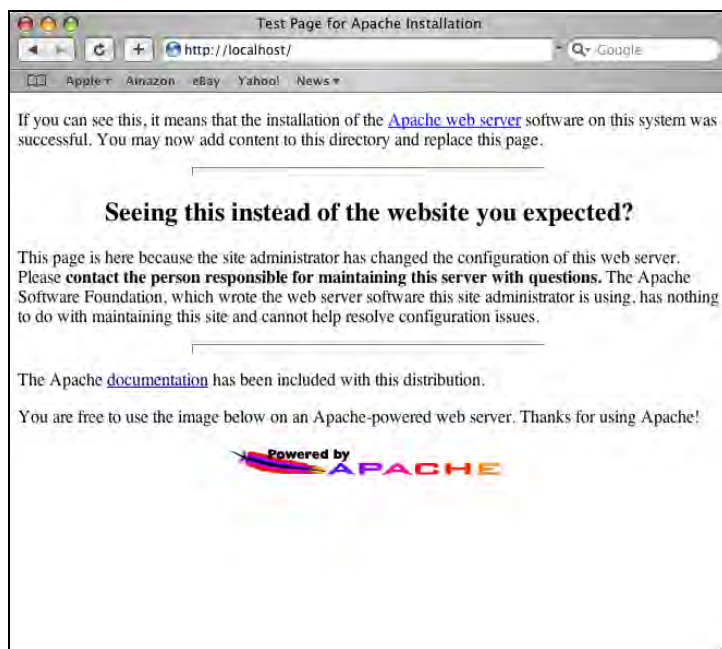
Sistem Manajemen Basis Data atau Database Management System (DBMS) seperti dijelaskan sebelumnya adalah perangkat lunak yang berfungsi mengelola basis data. Pada pemrograman web, DBMS merupakan bagian penting terutama pada web dinamis. Hal ini karena biasanya data atau informasi yang membuat halaman web menjadi dinamis disimpan dalam basis data. Pada Gambar 19.1. terlihat bagaimana hubungan antara basis data dengan server side script. Kedua bagian ini berada pada sisi server. Artinya pengguna atau client tidak perlu tahu bagaimana mekanisme basis data digunakan. Hal ini karena basis data hanya akan berhubungan dengan web server secara langsung. Beberapa DBMS yang umum digunakan dalam pemrograman web telah dijelaskan pada Bab 14.

19.5.5 Web Browser

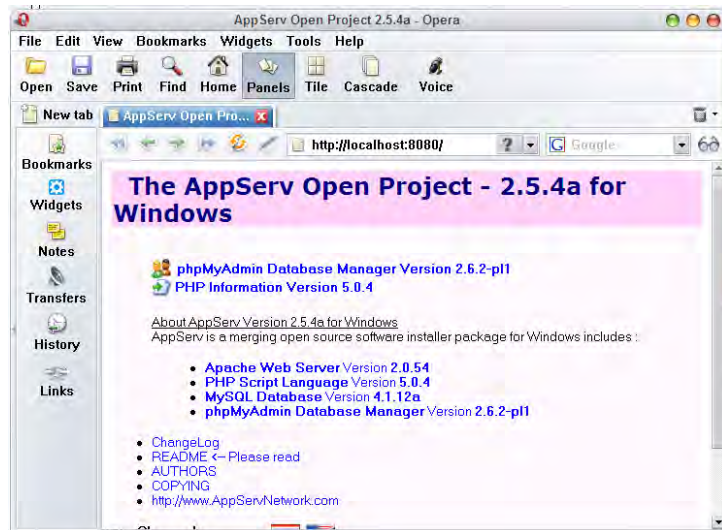
Web browser berfungsi menerjemahkan kode-kode HTML menjadi tampilan yang kita kehendaki. Ada banyak Web Browser tersedia di internet. Hampir semuanya dapat kita download secara bebas. Beberapa nama yang cukup terkenal antara lain Microsoft Internet Explorer, Firefox, Opera atau Safari. Microsoft Internet Explorer adalah default web browser pada system operasi Microsoft Windows (lihat Gambar 19.9). Firefox adalah default web browser pada sebagian besar system operasi Linux. Safari adalah default web browser pada system operasi Mac OS X (lihat Gambar 19.10). Sedangkan Opera adalah web browser keluaran Opera Software yang dapat berjalan pada berbagai platform system operasi (Gambar 19.11).



Gambar 19.9. Microsoft Internet Explorer.



Gambar 19.10. Safari.



Gambar 19.11. Opera.

19.6 BAHASA PEMROGRAMAN BERBASIS WEB

Pengertian bahasa pemrograman berbasis web pada buku ini akan dibatasi bahasa pemrograman yang digunakan dalam server side script. HTML tidak masuk dalam kelompok ini karena sebenarnya HTML bukanlah bahasa pemrograman. HTML hanyalah bahasa untuk mengatur bagaimana suatu dokumen web ditampilkan dalam web browser. Saat ini ada tiga teknologi utama yang digunakan dalam server side script, yaitu ASP (termasuk ASP.Net), PHP, dan JSP. Masing-masing memiliki kekuatan dan kelemahan.

19.6.1 ASP dan ASP.net

ASP sebenarnya bukan bahasa pemrograman karena ASP adalah mesin (engine) untuk server side script yang ditanamkan pada IIS. Bahasa pemrograman yang dipakai adalah VBScript. Namun orang lebih sering menyebut ASP saja. Secara default ini adalah server side script pada web server Microsoft Windows. Seperti halnya bahasa pemrograman web lainnya, untuk menandai bahwa suatu kode dalam halaman web merupakan VBScript maka digunakan tag sebagai penanda. Berikut ini suatu contoh VBScript:

```
<html>
<body>
<% Response.Write("Hello World!") %>
</body>
</html>
```

Atau sebagai berikut:

```
<html>
<body>
<%= "Hello World!" %>
</body>
</html>
```

Pada kode di atas bagian yang ditandai dengan <% dan %> merupakan bagian yang memuat kode VBScript. VBScript merupakan bahasa yang sangat mirip dengan Visual Basic. Namun telah disesuaikan dengan lingkungan web. Dokumen berisi kode ini dapat disimpan dalam bentuk html atau dengan ekstensi .asp. Namun yang paling penting adalah dokumen tersebut harus diletakkan pada directory yang dapat dibaca dan dieksekusi oleh web server IIS.

ASP.Net merupakan versi lanjutan dari ASP, namun dengan teknologi yang sangat berbeda. ASP.Net dibangun berdasarkan pada teknologi .Net yang dikembangkan oleh Microsoft. Bahasa yang digunakan tidak lagi VBScript namun telah menggunakan kemampuan penuh dari salah satu dari bahasa-bahasa dalam kelompok .Net yaitu Visual Basic.NET, C#, Visual J. Dengan teknologi .Net memungkinkan tag-tag HTML digantikan secara penuh oleh script-script yang ditulis dengan bahasa-bahasa .Net.

19.6.2 PHP

PHP adalah bahasa pemrograman yang didesain khusus untuk membuat halaman web. PHP adalah singkatan dari PHP: PHP Hypertext Preprocessor. Singkatan yang agak aneh. Awalnya, PHP adalah singkatan dari Personal Home Page yang pertama kali diciptakan oleh Rasmus Lerdorf. PHP diciptakan pertama kali untuk keperluan mencatat jumlah pengunjung hompagenya. Perkembangan php saat ini dapat dilihat pada www.php.net.

PHP adalah salah satu bahasa server-side yang paling populer. Kepopulerannya disebabkan kelebihan-kelebihannya dibanding bahasa sejenis, seperti Perl dan CGI. PHP mampu menutupi kekurangan pada bahasa pemrograman web pada umumnya. PHP mudah dibuat dan cepat dijalankan. PHP dapat berjalan dalam web server yang berbeda, seperti Apache, PWS, IIS dan sebagainya. PHP juga dapat berjalan dalam sistem operasi yang berbeda pula, seperti UNIX, Windows, Mac OS X dan Linux. PHP diterbitkan secara gratis. Source code PHP dapat di-download tanpa perlu mengeluarkan uang. PHP juga termasuk bahasa yang embedded (bisa diletakkan di dalam tag HTML). Pada Bab 21 akan dibahas secara detail penggunaan PHP dalam pembuatan halaman web dinamis.

TUJUAN

Setelah anda mempelajari bab ini diharapkan anda akan mampu :

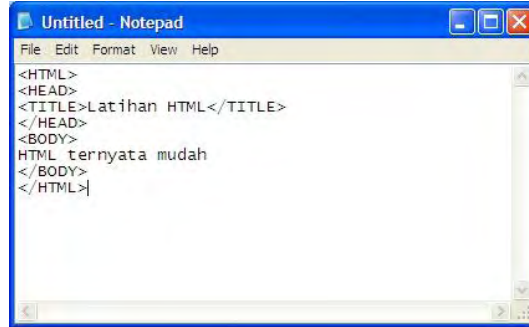
- Menjalankan perangkat lunak DBMS (Microsoft Access)
 - Membuat basis data, table, dan atribut dengan DBMS (Microsoft Access)
 - Membuat query dengan DBMS (Microsoft Access)
 - Membuat form dengan DBMS (Microsoft Access)
 - Membuat report dengan DBMS (Microsoft Access)
-

20.1. PENGERTIAN TENTANG HTML

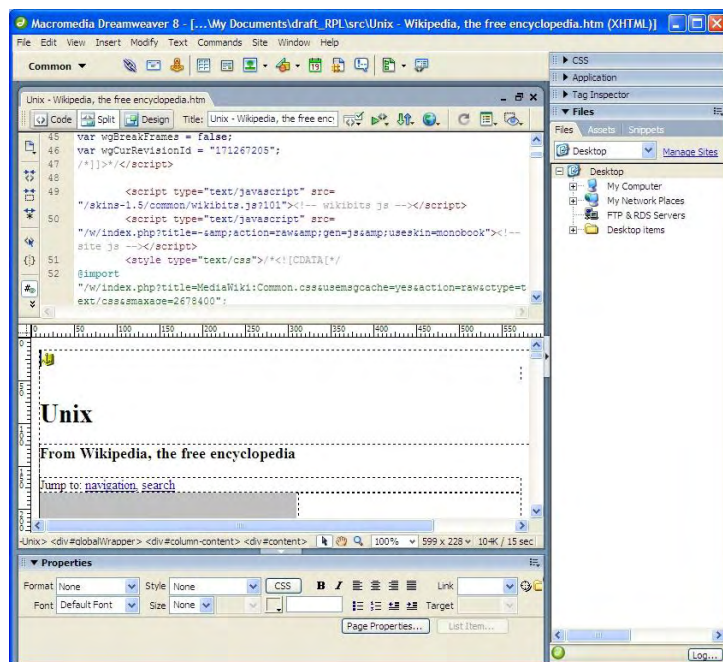
HTML (*Hypertext Markup Language*) adalah bahasa yang digunakan untuk menulis halaman web. HTML merupakan pengembangan dari standar pemformatan dokumen teks yaitu *Standard Generalized Markup Language* (SGML). HTML sebenarnya adalah dokumen ASCII atau teks biasa yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu.

HTML dibuat oleh Tim Berners-Lee ketika masih bekerja untuk CERN. HTML dipopulerkan pertama kali oleh browser Mosaic. Selama awal tahun 90'an, HTML mengalami perkembangan yang sangat pesat. Setiap pengembangan HTML pasti akan menambahkan kemampuan dan fasilitas yang lebih baik daripada versi sebelumnya. Perkembangan yang pesat tersebut tidak sampai merubah cara kerja HTML.

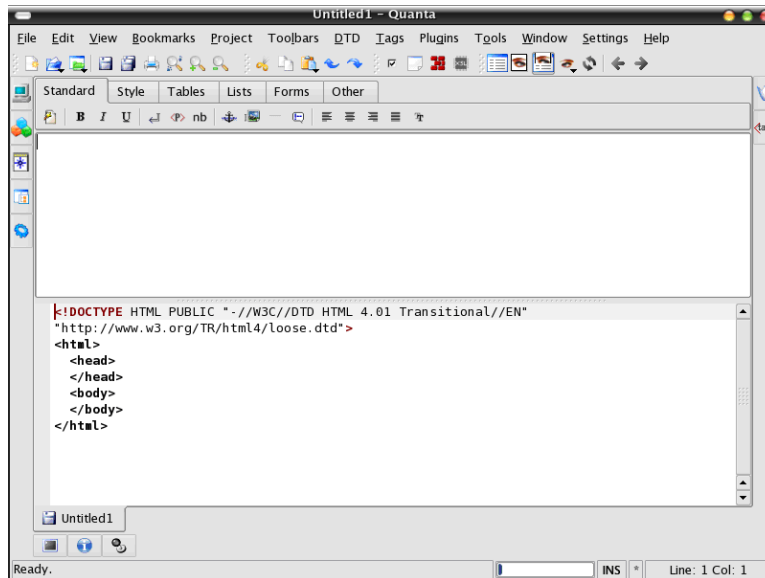
Sebuah dokumen atau file HTML agar dapat dibaca langsung oleh browser disimpan dalam ekstensi .htm atau .html. Untuk menulis HTML tidak dibutuhkan perangkat lunak yang spesifik, cukup dengan text editor sederhana seperti *Notepad* (pada Microsoft Windows) atau beragam text editor yang ada di platform Linux dan Apple Mac OS, seperti *vi*, *nano*, *joe*, *gedit*, *leafpad* dan lain-lain. Beberapa editor menyediakan fitur-fitur tambahan seperti *syntax coloring* (memberi warna pada kode-kode HTML) dan *code completion* (melengkapi secara otomatis kode yang akan dituliskan). Saat ini telah banyak perangkat lunak berbasis GUI yang sangat membantu dalam pembuatan halaman-halaman HTML. *Macromedia Dreamweaver* dan *Microsoft Frontpage* merupakan dua nama yang cukup populer di platform Microsoft Windows. Sedangkan di Linux tersedia *Quanta+*, *Bluefish* dan *Nvu*.



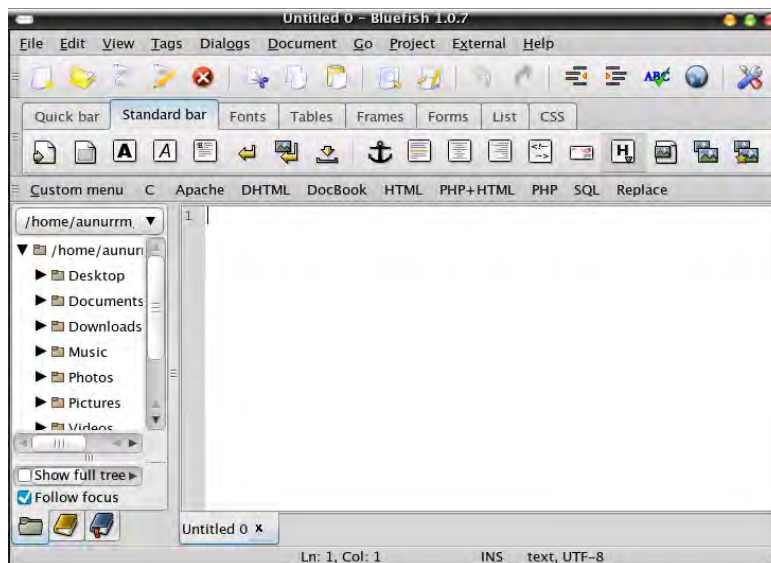
Gambar 20.1. Teks editor Notepad.



Gambar 20.2. Macromedia Dreamweaver.



Gambar 20.3. Quanta pada system operasi Linux



Gambar 20.4. Bluefish pada system operasi Linux

20.2. SINTAKS DAN STRUKTUR UMUM HTML

HTML adalah bahasa yang disisipkan (embedded language) pada dokumen dengan memberi tanda tertentu yang disebut tag. Tag merupakan aturan penulisan kode yang ditulis dengan diawali tanda lebih kecil dan di

akhiri dengan tanda lebih besar (<tag>). Browser akan menentukan tampilan teks atau dokumen berdasarkan tag yang digunakan.

Sintaks penulisan tag mengikuti aturan-aturan umum berikut ini:

- a) Setiap tag mempunyai nama yang spesifik. Kadang-kadang diikuti opsi-opsi yang disebut attribute. Baik nama maupun opsi harus berada dalam tanda <...>.

Contoh:

```
<a href="/wiki/PHP" title="PHP">PHP</a>
```

Pada contoh ini tagnya memiliki nama <a> sedangkan attribute untuk tag <a> adalah href dan title. Sehingga baik nama tag dan attributenya harus berada di dalam tanda <...> seperti pada contoh.

- b) Sebagian besar tag berpasangan. Penulisan untuk tag yang berpasangan adalah sebagai berikut : <namatag>...</namatag>

Contoh:

```
<TITLE>Paragraf</TITLE>  
<strong>Cetak Tebal</strong>
```

Pada tag yang berpasangan seperti pada contoh ini, <TITLE> adalah tag awal dan </TITLE> adalah tag akhir. Perhatikan tanda / pada tag akhir.

- c) Nama tag dan attribute-nya tidak bersifat case sensitive. Penulisan Cetak Tebal memberikan hasil yang sama dengan Cetak Tebal.
- d) Penulisan attribute suatu tag diletakkan setelah nama tag. Jika ada lebih dari satu attribute maka digunakan spasi untuk memisahkan. Urutan attribute tidak penting.

Contoh:

```
<FONT SIZE=3>Teks Baru</FONT>  
<FONT SIZE=5 FACE="verdana">Teks Baru </FONT>
```

- e) Nilai dari attribute ditulis setelah tanda sama dengan (=). Pada contoh sebelumnya (lihat bagian d) terlihat bahwa attribute SIZE dari tag FONT memiliki nilai 5 sedangkan attribute FACE memiliki nilai "verdana".
- f) Jika nilai dari attribute hanya tunggal, maka kita langsung menuliskan setelah tanda =. Jika lebih dari satu maka dapat digunakan tanda '...' atau "...". Pada contoh bagian d, tampak bahwa penulisan nilai untuk attribute SIZE tanpa menggunakan tanda "...", sedangkan pada attribute FACE tanda "..." untuk menandai kata **verdana**.

Dokumen HTML secara umum akan terdiri dari dua bagian yaitu Header dan Body (Gambar 20.5)


```

<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" /
>
<title>Untitled Document</title>
</head>

<body>
<strong>Cetak Tebal</strong>
</body>
</html>

```

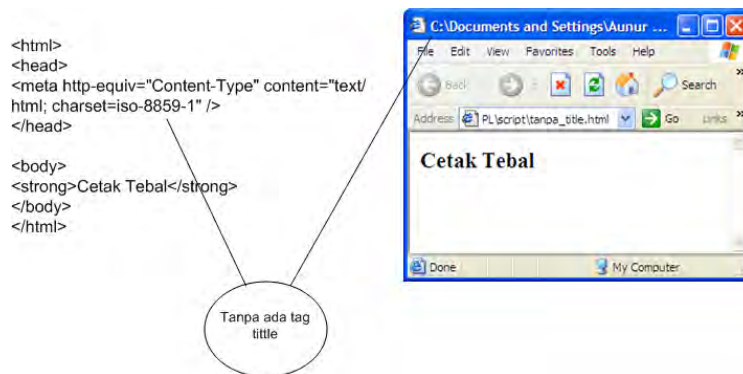
Header

Body

Gambar 20.5. Struktur umum dokumen HTML.

20.2.1. Header

Bagian ini biasanya berisi berbagai macam keterangan tentang dokumen termasuk title (judul dokumen), posisinya dalam sekumpulan halaman web dan hubungannya dengan dokumen lain. Bagian ini ditandai dengan tag `<head> ... </head>`. Tag ini tidak mempunyai attribute. Di dalam tag ini kita dapat meletakkan beberapa tag lain seperti tag title dan tag link. Lihat Gambar 20.6 dan 20.7.



Gambar 20.6. Header dokumen HTML tanpa tag title.



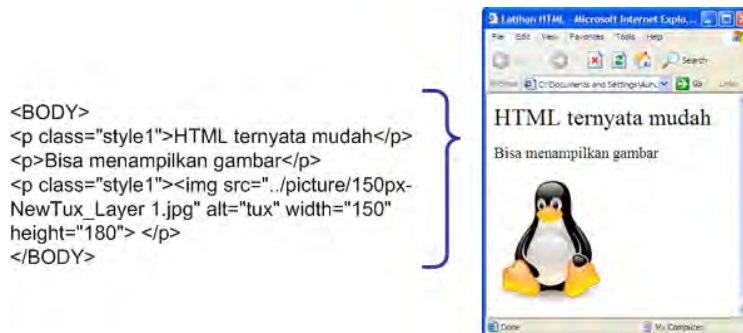
Gambar 20.7. Header dokumen HTML dengan tag title.

20.2.2. Body

Body adalah bagian dari dokumen HTML tempat dimana kita meletakkan isi dari dokumen. Bagian ini ditandai dengan tag `<body>` dan diakhiri dengan `</body>`. Apapun yang berada diantara dua tanda ini disebut sebagai *body content*. Dokumen HTML yang paling sederhana mungkin hanya berisi sebaris atau dua baris teks saja tanpa format apapun (Gambar 20.8). Pada dokumen yang lebih kompleks, body content bisa berisi teks yang terformat, gambar, table atau bahkan animasi yang rumit (Gambar 20.9).



Gambar 20.8. Dokumen HTML dengan body content sederhana.



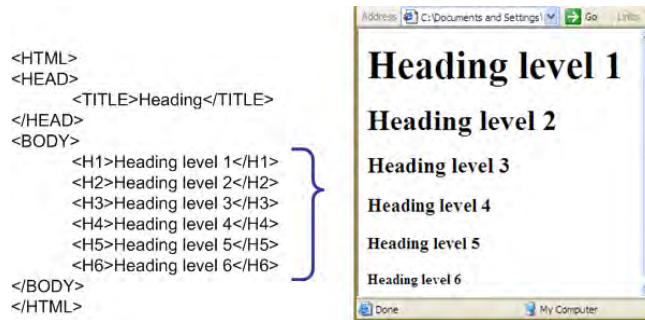
Gambar 20.9. Dokumen HTML dengan body content yang lebih kompleks.

20.3. FORMAT DOKUMEN

Ada banyak sekali tag HTML yang tersedia, baik itu berhubungan dengan deskripsi dokumen atau yang berhubungan dengan format tampilan dokumen. Tidak semuanya akan dibahas dalam buku ini. Pada bagian berikut ini akan dibahas beberapa tag yang biasa dipakai dalam format tampilan dokumen.

20.3.1. Heading

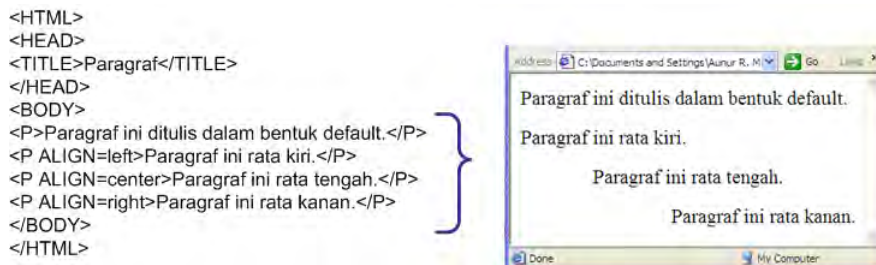
Heading adalah sekumpulan kata yang menjadi judul atau subjudul dalam sebuah dokumen HTML. Heading berbeda dengan tag `<TITLE>`. HTML menyediakan enam tingkatan heading, dimana heading level 1 (`<H1>`) adalah yang terbesar dan heading level 6 (`<H6>`) adalah yang terkecil. Gambar 20.10 menunjukkan bagaimana penggunaan heading.



Gambar 20.10. Penggunaan heading.

20.3.2. Paragraph

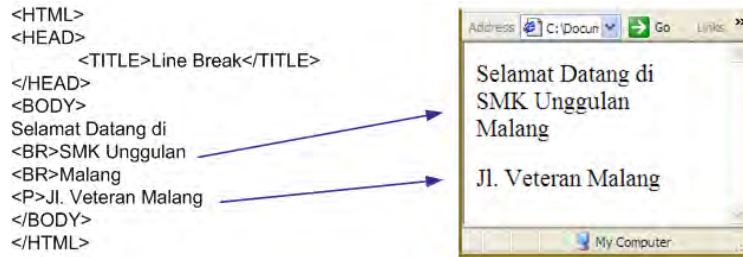
Paragraf dalam HTML dibuat dengan tag `<P>`. Tag ini akan membuat baris baru dengan menyisipi satu baris kosong. Penulisan isi paragraf diapit oleh `<P>` dan `</P>`. Pengaturan posisi paragraf dapat dilakukan dengan atribut **ALIGN** yang diikuti dengan posisi yang diinginkan, yaitu **left** untuk rata kiri, **center** untuk rata tengah horizontal dan **right** untuk rata kanan (Lihat Gambar 20.11).



Gambar 20.11. Penggunaan paragraph.

20.3.3. Line Break

Line break digunakan untuk menuliskan teks pada baris berikutnya. Line break dibuat dengan tag tunggal `<BR>`. Tag ini akan membuat baris baru tanpa memberi baris kosong sebagaimana pada tag `<P>` (Lihat Gambar 20.12).



Gambar 20.12. Tag `<BR>` dan `<P>`.

20.3.4. List

HTML menyediakan 3 cara untuk membuat daftar atau list, yaitu

- Ordered list

Ordered list digunakan untuk membuat daftar dimana setiap bagian memiliki nomor secara berurutan. Ordered list dimulai dengan tag `<OL>` dan diakhiri dengan tag `</OL>`, sedangkan setiap bagiannya digunakan tag `<LI>` tanpa tag penutup. Tag ini menggunakan angka sebagai urutan secara default, tapi bisa dirubah dengan merubah nilai attribute **TYPE**. Nilai-nilai attribute yang diijinkan dapat dilihat pada tabel 20.1. Contoh penggunaan Ordered list dapat dilihat pada Gambar 20.13.



Gambar 20.13. Penggunaan Ordered List

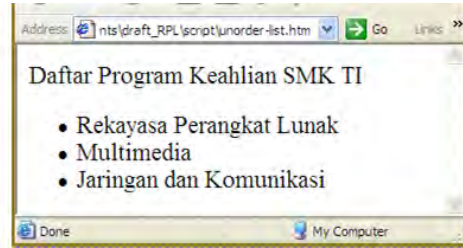
- Unordered list

Unordered list digunakan untuk membuat daftar yang disajikan tanpa nomor urut, melainkan dengan secara default dengan tanda bulatan utuh (bullet). Tanda ini bisa dirubah dengan merubah nilai attribute **TYPE** untuk unordered list. Nilai-nilai attribute yang diijinkan dapat dilihat pada tabel 20.1. Unordered list dimulai dengan tag `<UL>` dan diakhiri dengan tag `</UL>`, sedangkan setiap bagiannya digunakan tag `<LI>` tanpa tag penutup (lihat contoh pada Gambar 20.14).

```

<BODY>
Daftar Program Keahlian SMK TI
<UL>
<li>Rekayasa Perangkat Lunak
<li>Multimedia
<li>Jaringan dan Komunikasi
</UL>
</BODY>

```



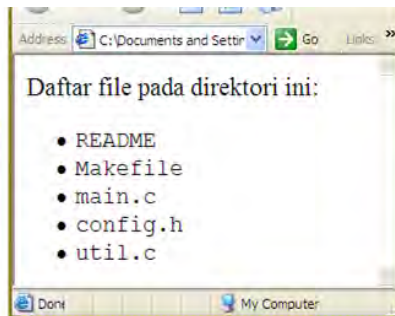
Gambar 20.14. Penggunaan Unordered List

Tipe lain dari list yang termasuk Unordered list adalah Directory list dan Menu list. Directory list merupakan daftar tak bernomor yang digunakan untuk menangani direktori. Tag yang digunakan adalah `<DIR> ... </DIR>`. Menu list umumnya digunakan untuk menu pilihan. Tag yang digunakan adalah `<MENU> ... </MENU>`. Baik Direktori list maupun Menu list menghasilkan tampilan yang sama dengan Unordered list (Gambar 20.15 dan Gambar 20.16).

```

<BODY>
Daftar file pada direktori ini:
<dir>
<li><code>README</code>
<li><code>Makefile</code>
<li><code>main.c</code>
<li><code>config.h</code>
<li><code>util.c</code>
</dir>
</BODY>

```



Gambar 20.15. Penggunaan Direktori List

```

<BODY>
Daftar Menu
<menu>
<li>Menu 1
<li>Menu 2
<li>Menu 3
<li>Menu 4
<li>Menu 5
</menu>
</BODY>

```



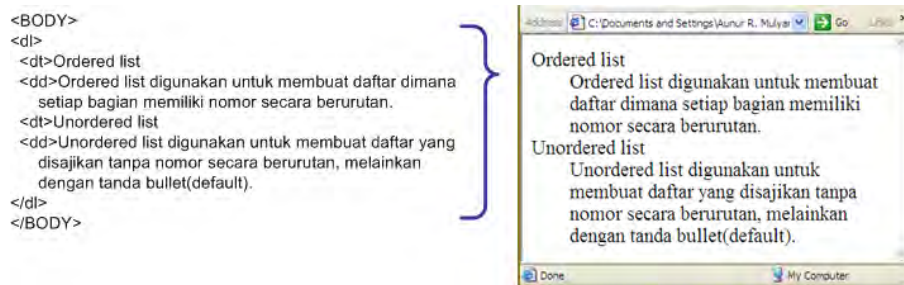
Gambar 20.16. Penggunaan Menu List

- Definition list

Definition list membuat daftar definisi mirip seperti tampilan pada kamus, dengan definisi suatu istilah agak menjorok ke kanan. Tiga buah pasang tag yang terkait dengan definition list adalah:

- ✓ **<DL> ... </DL>** untuk menyatakan tempat bagi daftar definisi.
- ✓ **<DT> ... </DT>** untuk menyatakan tempat bagi istilah yang akan didefinisikan.
- ✓ **<DD> ... </DD>** untuk menyatakan tempat bagi definisi dari istilah.

Contoh penggunaan Definition list dapat dilihat pada Gambar 20.17.



Gambar 20.17. Penggunaan Definition lis List

Tabel 20.1. Daftar attribute TYPE untuk Ordered list dan Unordered list.

Jenis List	Atribut	Fungsi
Ordered list	TYPE=1	Daftar berurutan dengan angka arab (1,2,3,...)
	TYPE=I	Daftar berurutan dengan angka romawi besar (I,II,III,...)
	TYPE=i	Daftar berurutan dengan angka romawi kecil (i,ii,iii,...)
	TYPE=A	Daftar berurutan dengan abjad besar (A,B,C,...)
	TYPE=a	Daftar berurutan dengan abjad kecil (a,b,c,...)
Unordered List	TYPE=circle	Daftar dengan tanda lingkaran
	TYPE=square	Daftar dengan tanda tanda kotak
	TYPE=disk	Daftar dengan tanda cakram

20.3.5. Font

HTML menyediakan fasilitas pengaturan huruf yang akan ditampilkan dalam dokumen. Pengaturan ini dilakukan dengan tag berpasangan **** dan ****. Tag ini memiliki beberapa atribut untuk mengatur ukuran, jenis dan warna huruf yang digunakan.

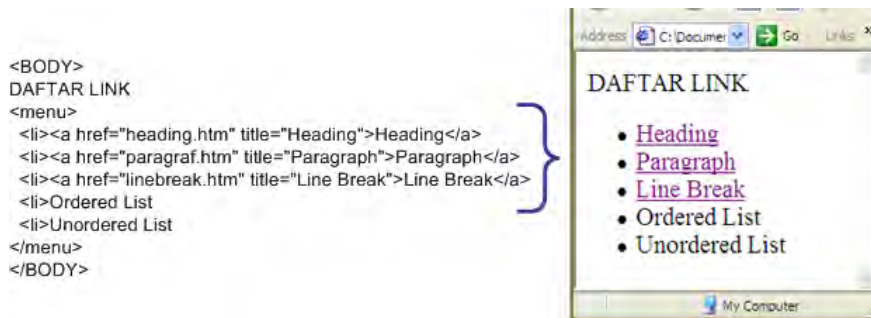
- Atribut **SIZE** untuk mengatur ukuran huruf, dimana nilai 1 untuk huruf terkecil dan nilai 7 untuk huruf terbesar.
- Atribut **FACE** untuk mengatur jenis huruf yang diinginkan, dimana nilainya berupa string nama font seperti Arial, Tahoma dan sebagainya.
- Atribut **COLOR** untuk mengatur warna teks yang dikehendaki, dimana nilainya dapat diisi dengan dua cara dengan menyebutkan nama warna dalam bahasa Inggris seperti red, blue dan green atau dengan menggunakan nilai RGB (Red Green Blue) seperti **FF0000** untuk merah.



Gambar 20.18. Penggunaan tag Font.

20.3.6. Link

Link merupakan pautan untuk membuka atau memanggil halaman web atau file tertentu. Link merupakan tag yang sangat penting dalam penggunaan HTML, karena disinilah letak perbedaan antara dokumen HTML dengan dokumen teks yang lain. Link dapat dibuat dengan memberi perintah tag anchor **<A>**. Anchor memiliki beberapa atribut, diantaranya **HREF** yang berfungsi untuk membuat link ke dokumen HTML tertentu dan **NAME** yang berfungsi untuk memberi tanda/nama titik tertentu pada dokumen HTML yang sama. Contoh penggunaan tag anchor dapat dilihat pada Gambar 20.19.



Gambar 20.19. Penggunaan tag anchor.

Attribute **HREF** dapat digunakan untuk memanggil halaman web pada system yang sama (pada satu computer) seperti ditunjukkan pada Gambar 20.19. Pada kasus ini kita tinggal menuliskan lokasi dimana halaman yang akan kita panggil berada. **HREF** dapat juga kita gunakan untuk memanggil halaman lain diluar system kita atau memanggil situs-situs lain di internet. Caranya dengan mengetikkan alamat URL situs yang akan kita panggil. Misalnya :

DAFTAR ALAMAT MESIN Pencari

```
<menu>
    <li><a href="http://www.google.com/">Google</a>
    <li><a href="http://www.yahoo.com/">Yahoo</a>
    <li><a href="http://www.altavista.com/">Altavista</a>
</menu>
```

20.3.7. Horizontal Line

Untuk mempercantik tampilan halaman HTML, kita dapat menambahkan garis horizontal dengan tag **<HR>**. Tag **<HR>** mempunyai attribute **SIZE** untuk menentukan ketebalan garis, attribute **WIDTH** untuk menentukan lebar garis, Attribute **ALIGN** untuk menentukan letak teks dalam garis, dan attribute **NOSHADE** untuk mengatur agar garis tidak disertai bayangan. Gambar 20.20 menunjukkan bagaimana tag **<HR>** digunakan.

```
<BODY>
<div align="center">DAFTAR LINK</div>
<hr align="center" width="80%" size="2">
<menu>
<li><a href="heading.htm" title="Heading">Heading</a>
<li><a href="paragraf.htm" title="Paragraph">Paragraph</a>
<li><a href="linebreak.htm" title="Line Break">Line Break</a>
</menu>
<hr align="center" width="60%" size="5" noshade>
</BODY>
```



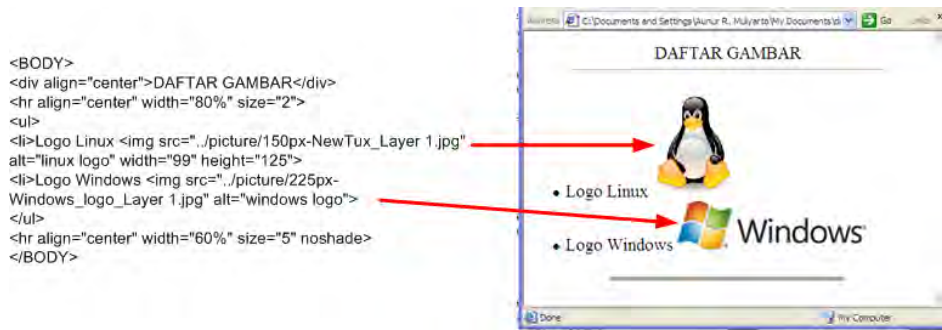
Gambar 20.20. Penggunaan garis.

20.3.8. Image

Dokumen HTML dapat diperindah dengan menyertakan gambar pada halaman web yang dibuat. Tag **** dapat digunakan untuk memanggil dan menampilkan gambar pada halaman web. Sintaks penulisan tag **** adalah:

```
<IMG SRC="file_gambar" ALT="nama_alternatif">
```

Atribut **SRC** digunakan untuk menentukan sumber file gambar yang akan ditampilkan. Atribut **ALT** berfungsi untuk memberi tulisan pengganti, apabila gambar tidak ditampilkan.

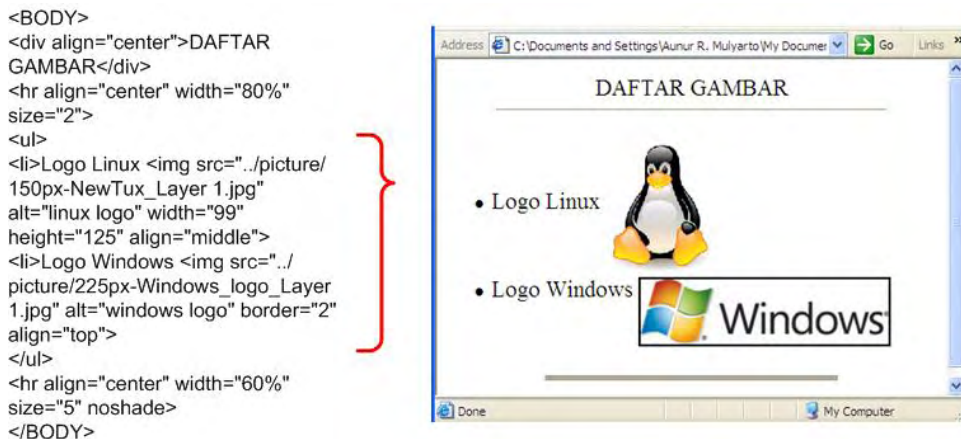


Gambar 20.21. Penggunaan tag image.

Untuk pengaturan gambar yang lebih baik, tag IMG menyediakan beberapa attribute, antara lain:

- Attribute **ALIGN** untuk mengatur penempatan teks pada gambar.
- Attribute **BORDER** untuk memberi bingkai pada gambar.
- Attribute **HEIGHT** dan **WIDTH** untuk mengatur tinggi dan lebar gambar.

Contoh berikut ini memperlihatkan penggunaan attribute-attribute tersebut.



Gambar 20.22. Penggunaan attribute-attribute tag IMG.

Pada Gambar 20.22 terlihat perbedaan tampilan dibandingkan dengan Gambar 20.21. Tulisan Logo Linux terletak ditengah gambar karena kita menggunakan attribute **align** dengan nilai **middle**. Sedangkan tulisan logo Windows terletak di bagian atas karena kita menggunakan **align** dengan nilai **top**. Gambar logo windows diberi garis bingkai dengan menggunakan attribute **border** dengan nilai 2.

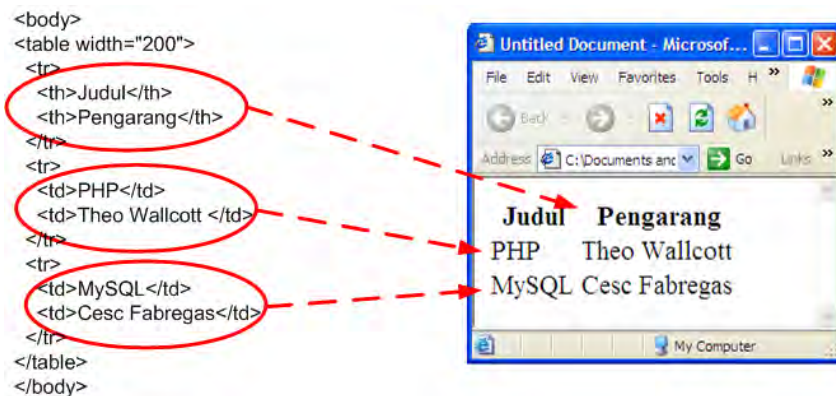
20.3.9. Tabel

Tabel dalam HTML dibuat dengan menggunakan tag awal **<TABLE>** dan tag penutup **</TABLE>**. Tag ini memiliki beberapa bagian penting, seperti dapat dilihat pada Tabel berikut ini.

Tabel 20.2. Bagian-bagian pada tag Table

Tag	Fungsi
<CAPTION>...</CAPTION>	Membentuk judul tabel
<TH>...</TH>	Membuat judul kolom
<TR>...</TR>	Membentuk baris pada suatu tabel
<TD> . . . </TD>	Membuat sebuah sel data

Contoh-contoh penggunaan table adalah sebagai berikut:



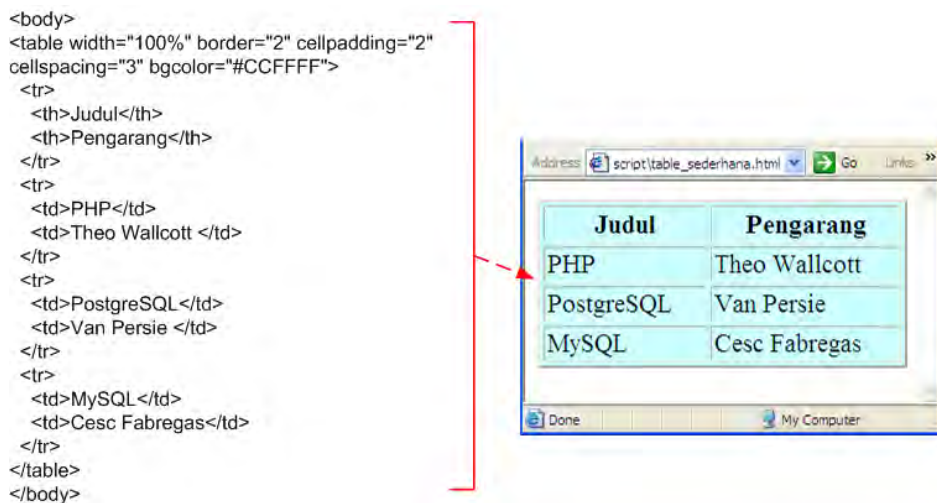
Gambar 20.23. Table sederhana.

Pada Gambar 20.23, table yang kita buat adalah table sederhana dengan dua buah kolom dan 3 buah baris (perhatikan ada 3 buah pasangan tag **<TR> ... </TR>**). Secara default table ditampilkan tanpa ada garis pada table tersebut. Kita dapat menambahkan garis dengan menggunakan attribute **border** pada table (lihat Gambar 20.24).

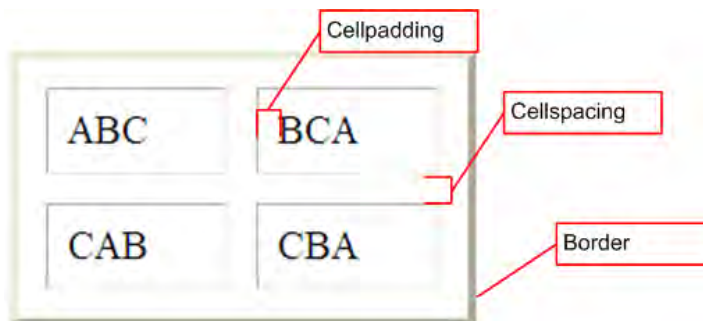
Pada Gambar 20.24, terlihat table yang tampilannya lebih baik daripada Gambar sebelumnya. Ada beberapa attribute yang kita tambahkan pada table yaitu :

- Attribute **WIDTH** untuk mengatur lebar table pada halaman. Kita dapat menggunakan satuan persen (%) atau pixel (px).

- Attribute **BORDER** untuk memberikan garis pada table. Nilai untuk attribute ini dari dimulai dari 0 yang berarti tidak ada garis. Semakin besar angka semakin tebal garis.
- Attribute **BGColor** untuk menambahkan warna latar belakang pada table.
- Attribute **CELLPADDING** untuk menentukan jarak antara teks dan tepi kiri sebuah sel (lihat gambar 20.25 untuk lebih jelasnya).
- Attribute **CELLSPACING** untuk menentukan jarak bagian sel terhadap tepi dalam bingkai table (lihat gambar 20.25 untuk lebih jelasnya).

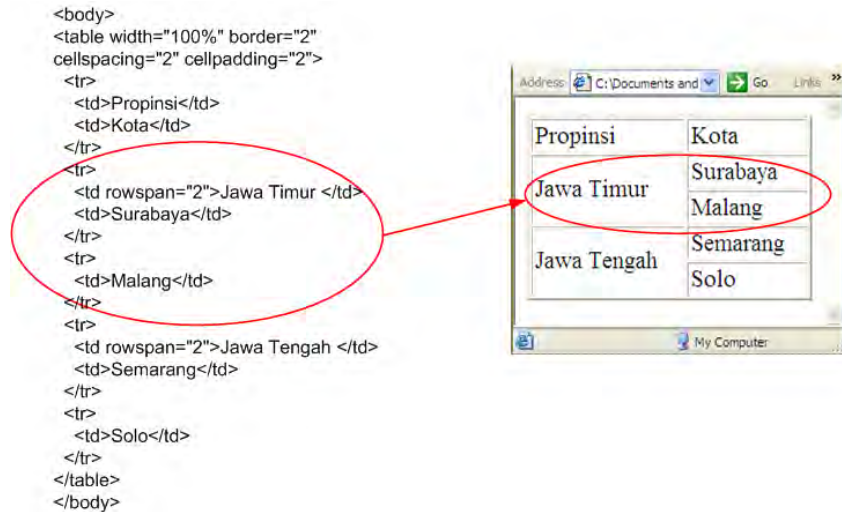


Gambar 20.24. Tabel dengan format yang lebih kompleks.

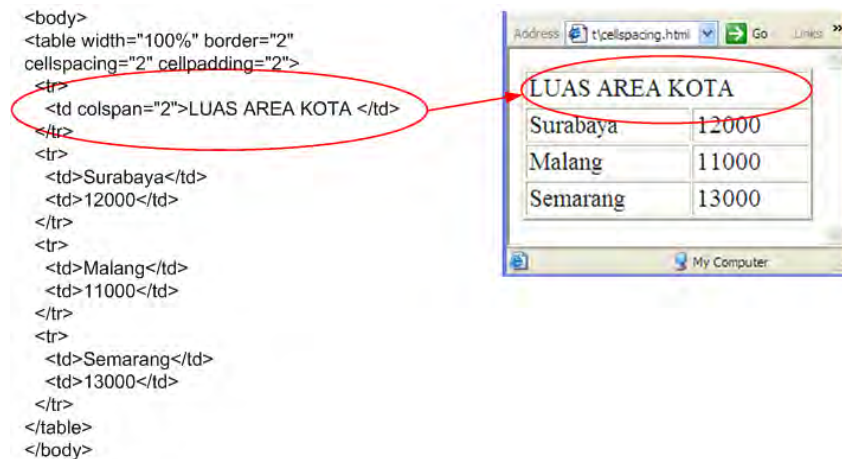


Gambar 20.25. Cellpadding, cellspacing dan border.

Seperti halnya perangkat lunak word-processor, pada HTML kita dapat menggabungkan dua atau lebih sel menjadi satu buah sel. Untuk menggabungkan baris dapat digunakan attribute **ROWSPAN** dan untuk menggabungkan kolom dapat digunakan attribute **COLSPAN**. Contoh penggunaannya dapat dilihat pada Gambar 20.26 dan 20.27.

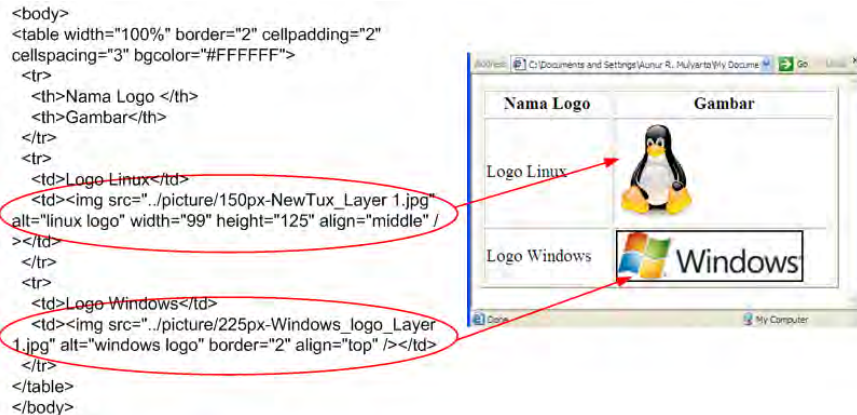


Gambar 20.26. Rowspan.



Gambar 20.27. Colspan.

Sel pada table tidak selalu harus berisi teks namun dapat juga berisi gambar seperti terlihat pada Gambar 20.28.



Gambar 20.28. Tabel dengan sel berisi gambar.

20.4. BEKERJA DENGAN FORM

Jika kita sering membuka-buka halaman web di internet, seringkali kita menjumpai bagian-bagian dimana kita harus memasukkan atau mengetikkan data-data tertentu. Misalnya sebeleum kita masuk ke inbox email, kita diminta mengetikkan user name dan password pada tempat yang telah disediakan. Fasilitas seperti ini biasa disebut sebagai form (formulir).

Formulir dibentuk dengan menggunakan pasangan tag **<FORM>** dan **</FORM>**. Dua attribute utama dari FORM yang sering digunakan adalah **ACTION** dan **METHOD**.

- **ACTION** menentukan alamat yang akan dijalankan dan menerima semua masukan pada FORM. Jika **ACTION** tidak disebutkan, informasi akan dikirim ke alamat yang sama dengan halaman FORM itu sendiri.
- **METHOD** digunakan untuk menentukan bagaimana informasi dikirim ke alamat yang disebutkan dalam **ACTION**. Nilai yang umum digunakan adalah **GET** dan **POST**. **POST** membuat informasi akan dikirim secara terpisah dengan alamat, sedangkan **GET** akan membuat informasi dikirim menjadi satu dengan alamat yang dituju.

20.4.1. Tipe Input

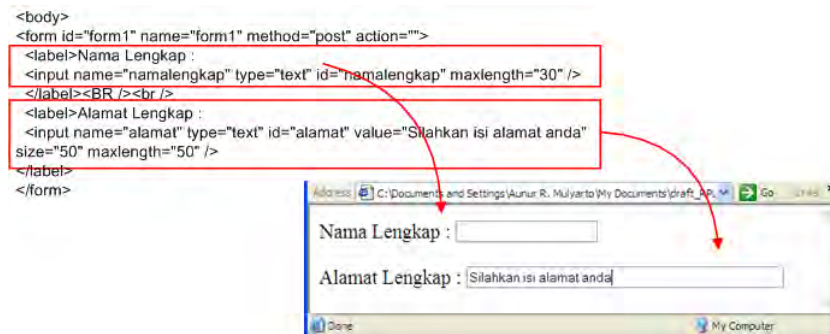
Ada beberapa tipe input yang diterima oleh FORM yaitu **TEXT**, **PASSWORD**, **CHECKBOX**, **RADIO**, **SUBMIT** dan **RESET**. Tipe-tipe input ini secara eksplisit menggunakan tag **<INPUT>**. Selain itu ada tipe input lain yang tidak menggunakan tag **<INPUT>** yaitu, **TEXTAREA** dan **SELECT**. Attribute untuk tag **<INPUT>** dapat dilihat pada table 20.3.

Tabel 20.3. Attribute-attribute tag **<INPUT>**

Attribute	Fungsi
TYPE	Menentukan tipe input
NAME	Menentukan nama data. Attribute ini diperlukan oleh semua jenis input, kecuali submit dan reset
MAXLENGTH	Menentukan jumlah maksimal karakter pada tipe input text dan password
VALUE	Menentukan nilai awal dari input
SIZE	Menentukan ukuran kotak input pada tipe input text dan password
CHECKED	Mengatur agar checkbox dalam keadaan terpilih pada keadaan awal

- Tipe TEXT

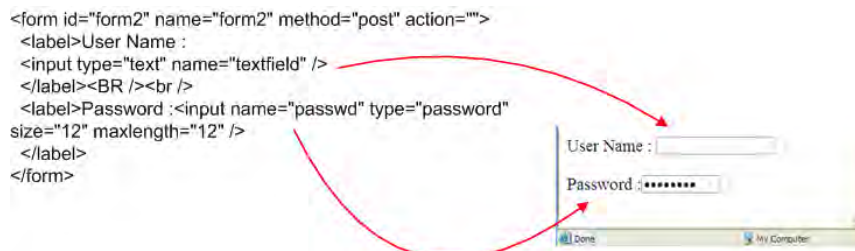
Tipe TEXT berguna untuk memasukkan nama orang, benda, alamat atau teks lain yang tidak terlalu panjang. Gambar berikut ini menunjukkan penggunaan tipe TEXT.



Gambar 20.29. Penggunaan input tipe TEXT

- Tipe PASSWORD

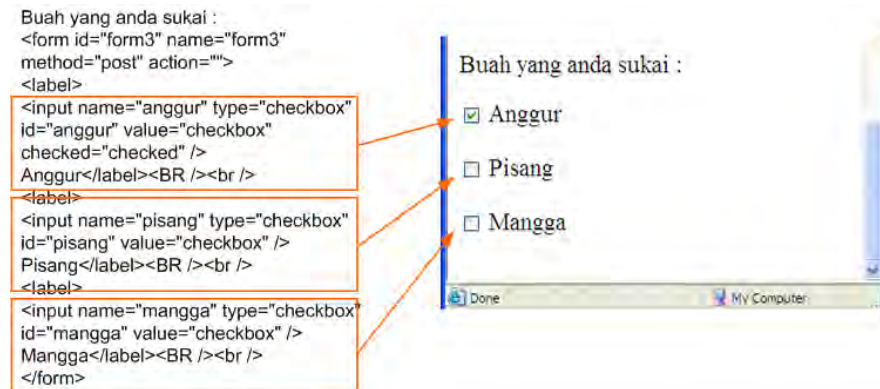
Tipe password sebenarnya sama dengan tipe text, hanya bedanya apabila mengetikkan sesuatu pada kotak password, maka karakter akan disembunyikan dan diganti misalnya dengan tanda *. Berikut ini contoh penggunaan tipe password.



Gambar 20.30. Penggunaan input tipe PASSWORD

- Tipe CHECKBOX

Tipe ini berguna untuk membuat kotak cek. Kotak cek adalah suatu bentuk masukan yang memungkinkan pengguna memilih dengan cara mengklik pada pilihan yang diinginkan. Pengguna dapat memilih satu atau lebih dari satu pilihan atau tidak memilih sama sekali. Attribute checked dapat digunakan untuk memberi nilai awal pada suatu pilihan. Berikut ini contoh checkbox.



Gambar 20.31. Penggunaan input tipe CHECKBOX

- Tipe RADIO

Berbeda dengan checkbox, radio digunakan dalam dua atau lebih tombol yang memungkinkan pengguna hanya memilih satu pilihan. Seperti halnya checkbox, attribute checked juga dapat digunakan untuk memberi nilai awal pilihan.

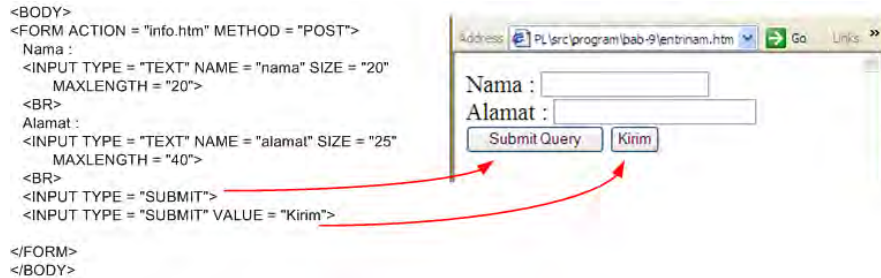


Gambar 20.32. Penggunaan input tipe RADIO

- Tipe SUBMIT

Tipe SUBMIT akan membentuk tombol submit. Apabila pengguna mengklik tombol tersebut maka akan menyebabkan alamat yang disebutkan pada ACTION pada tag <FORM> akan dimuat. Tipe SUBMIT

mempunyai atribut value yang bila diisi akan merubah kata yang muncul dalam tombol (Gambar 20.33).

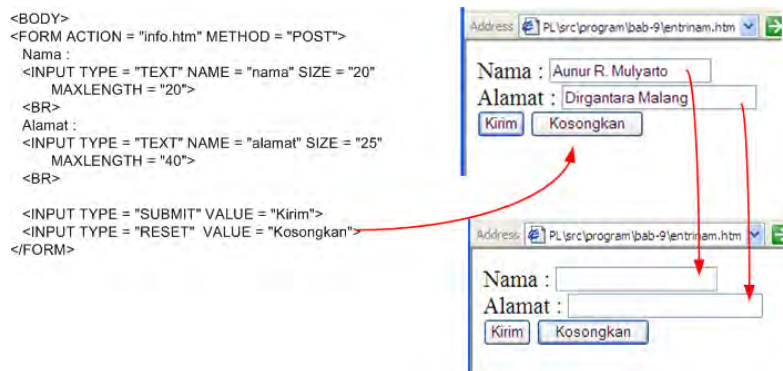


Gambar 20.33. Penggunaan input tipe SUBMIT

- Tipe RESET

Tipe RESET berfungsi untuk mengosongkan atau mengembalikan nilai field teks ke nilai asal. Seperti halnya SUBMIT, RESET juga memiliki atribut value yang bila diisi akan merubah kata dalam tombol.

Pada Gambar 20.34, tampak bahwa field nama telah diisi (gambar kanan atas) ketika tombol kosongkan (RESET) ditekan maka field-field tersebut akan kembali dikosongkan (gambar kanan bawah)



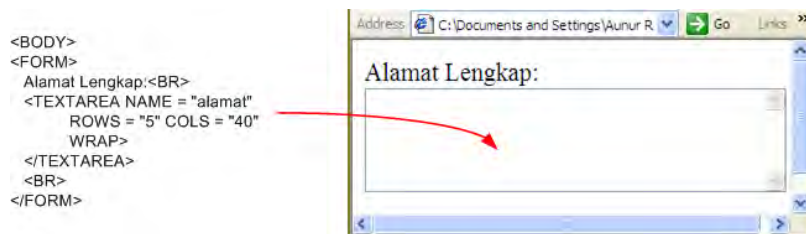
Gambar 20.34. Penggunaan input tipe RESET

- Tipe TEXTAREA

Pasangan tag `<TEXTAREA>` dan `</TEXTAREA>` berfungsi untuk membentuk suatu masukan dalam bentuk teks yang panjang dan bisa mencakup banyak baris. Atribut untuk TEXTAREA, yaitu:

- **NAME** untuk menentukan nama textarea

- **ROWS** menentukan jumlah baris yang akan ditampilkan
- **COLS** menentukan jumlah kolom yang akan ditampilkan
- **WRAP** menentukan apakah secara otomatis teks akan diturunkan ke baris berikutnya jika panjang baris lebih dari kolom textarea



Gambar 20.35. Penggunaan tipe TEXTAREA

- Tipe SELECT

SELECT digunakan untuk membuat combo box yang berisi daftar pilihan. Pengguna dapat melihat daftar pilihan dengan mengklik pada combo box dan memilih dengan cara mengkliknya pada pilihannya (Gambar 20.36).



Gambar 20.36. Penggunaan tipe SELECT.

20.5. STYLE SHEETS

Style Sheets merupakan feature yang sangat penting dalam membuat Dynamic HTML. Meskipun bukan merupakan suatu keharusan dalam membuat web, akan tetapi penggunaan style sheets merupakan kelebihan tersendiri. Suatu style sheet merupakan tempat dimana kita mengontrol dan manage style-style yang ada. Style sheet mendeskripsikan bagaimana tampilan document HTML di layar. Kita juga bias menyebutnya sebagai template dari documents HTML yang menggunakannya. Kita juga bisa membuat efek-efek spesial di web kita dengan menggunakan style sheet. Sebagai contoh kita bisa membuat style sheet yang mendefinisikan style untuk <H1> dengan style bold dan italic dan berwarna biru. Atau pada tag <P>

yang akan di tampilkan dengan warna kuning dan menggunakan font verdana dan masih banyak lagi yang bisa kita lakukan dengan style sheet.

Secara teoritis kita bisa menggunakan style sheet technology dengan HTML. Akan tetapi pada prakteknya hanya Cascading Style Sheet (CSS) technology yang support pada hampir semua web Browser. Karena CSS telah distandardkan oleh World Wide Web Consortium (W3C) untuk di gunakan di web browser.

- **Inline Styles**

Ada dua cara untuk merubah style dari web page kita yaitu:

- Merubah inline style
- Menulis script untuk merubah style kita.

Dengan menggunakan inline style kita dapat membuat dynamic style tanpa harus menambahkan script ke web kita. Inline styles merupakan style yang bisa kita pasang pada element web tertentu saja.

Contoh:

Jika kita ingin menambahkan style pada <H1> dengan warna merah, kita harus mengeset atribut STYLE dari tag <H1>.

```
<H1 STYLE="color:red">
```

jika kita ingin menggunakan script untuk memodifikasi inline style, kita dapat menggunakan Style Object. Style Object mensupport semua property yang di support CSS untuk style. Untuk menggunakan property pada script hilangkan tanda hubung "-" dari property CSS Style dan ganti huruf setelah tanda hubung menjadi Capital.

Contoh:

```
font-weight menjadi fontWeight  
text-align menjadi textAlign
```

- **Istilah-istilah dalam Style Sheet**

- a) Style rule

Cascading style sheet merupakan kumpulan aturan yang mendefinisikan style dari document. Sebagai contoh kita bisa membuat aturan style yang menentukan bahwa semua <H2> di tampilkan dengan warna orange.

- b) Style sheet

Style sheet dapat di embedded ke HTML document. Atau disebut embedded style sheet. Style sheet juga bisa dibuat sebagai

external file dan di link ke document HTML. Style role bisa di kenakan pada bagian tertentu dari web page. Sebagai contoh kita bisa menentukan paragraph tertentu di tampilkan dengan style bold dan italic sementara yang lain tetap seperti biasa.

c) Selector

```
selector { property1: value; property2:value, . . . }  
H1{ color:green; background-color:orange}
```

Style sheets terdiri dari dua bagian:

1. Selector

Bagian pertama sebelum tanda “{” disebut selector

2. Declaration

Terdiri dari property dan nilainya.

- **Komentar dalam Style Sheets**

Comments atau komentar biasanya di gunakan oleh programmer untuk memudahkan mengingat kembali script yang sudah di tulisnya, Comments di CSS hampir sama dengan comments di C atau C++ yaitu dengan menggunakan:

```
/* isi Comments */
```

Contoh:

```
H1 { color:blue;} /* H1 elements akan menjadi biru */  
Tags.H1.color = "blue"; /* H1 elements akan menjadi biru */
```

- **Pemakaian elemen style**

Mari kita mulai dengan mengatur warna huruf dan latar belakang. Kita dapat mengerjakan ini dengan menggunakan elemen style untuk mengatur karakter kode tag dokumen kita.

```
<style type="text/css">  
body { color: black; background: white; }  
</style>
```

Pernyataan yang ditulis antara kode tag <style> dan </style> menunjukkan perintah pengaturan style.

a) Link ke sheet lainnya

Apabila kita menginginkan style yang sama untuk halaman HTML kita yang lain, disarankan mempergunakan sheet-sheet terpisah namun satu dan lainnya terhubung dengan cara link. Kita dapat mengikuti cara berikut ini :

```
<link rel="stylesheet" href="style.css">
```

Kode tag untuk link ini ditempatkan di bagian "head" dokumen kita. Perintah rel perlu diatur dengan pernyataan "stylesheet" agar supaya

browser dapat menemukan perintah href sebagai penunjukan ke alamat Web (URL) sheet kita.

b) Mengatur tepi halaman (page margin)

Halaman Web akan tampil cantik bila dituliskan dalam margin yang lebih lebar. Kita dapat mengatur sisi kiri dan kanan memakai karakter "margin-left" dan "margin-right".

Contoh :

```
<style type="text/css">
body { margin-left: 10%; margin-right: 10%; }
</style>
```

Perintah di atas dituiskan dengan tujuan agar tampilan halaman Web di layar monitor memiliki batas halaman kiri 10% dari lebar layar monitor.

c) Mengatur inden kiri dan kanan

Agar halaman Web kita tampil lebih cantik bisa juga diberikan inden (spasi) dari margin kiri beberapa huruf sebelum menuliskan awal kalimat.

Contoh :

```
<style type="text/css">
body { margin-left: 10%; margin-right: 10%; }
h1 { margin-left: -8%;}
h2,h3,h4,h5,h6 { margin-left: -4%; }
</style>
```

d) Mengatur jarak penulisan dari tepi atas dan bawah halaman

Program Browser biasanya mengerjakan batas atas dan bawah, paragraf dan lain-lain dengan baik. Namun ketika kita ingin membuat ruang disebelah atas atau bawah halaman web kita, atau kita ingin membuat spasi yang khusus, barulah disini kita perlu memiliki cara mengaturnya.

Property "margin-top" menentukan ruang sebelah atas dan property "margin-bottom" menentukan ruang sebelah bawah halaman web kita. Bila kita hendak mengatur semuanya dengan heading h2, kita cukup menuliskan dengan perintah HTML sebagai berikut :

```
h2 { margin-top: 8em; margin-bottom: 3em; }
```

Kode em merupakan unit penting dalam mengatur ukuran tinggi font (huruf). Ini lebih mudah bila dibandingkan dengan pengaturan pixel atau titik (point). Unit ini akan sangat berguna pada pembuatan huruf besar. Satuan titik (Point) umumnya dipergunakan oleh program word processor misalnya dituliskan ukuran huruf 10 pt. Sayangnya untuk ukuran titik yang sama, menghasilkan ukuran huruf yang berbeda pada pemakaian program browser yang berbeda pula. Apa yang kita

kerjakan dengan baik dengan menggunakan sebuah program browser, bila dibaca dengan program browser yang lain belum tentu baik !. Pergunakanlah kode em untuk mengatasi hal ini.

Untuk mengatur ruang sebelah atas bagian heading halaman web, kita sebaiknya membuat nama style untuk heading tersebut. Dalam penulisan HTML-nya kita cukup menggunakan atribut class.

Contoh :

```
<h2 class="subsection">Getting started</h2>
```

Kemudian pengaturan ruangnya ditulis dengan perintah berikut :

```
h2.subsection { margin-top: 8em; margin-bottom: 3em; }
```

Pengaturan ini dimulai dengan nama tag, sebuah titik dan kemudian nilai dari atribut class. Hati-hati dalam menempatkan ruang sebelum atau sesudah titik tersebut. Bila pengaturan tersebut tidak memberikan hasil. Ada cara lain untuk mengatur style elemen tertentu. Tapi atribut class tetap bersifat fleksibel.

Pada saat sebuah "heading" diikuti dengan sebuah paragraf, nilai untuk batas bawah (margin-bottom) untuk heading tersebut tidak ditambahkan dengan nilai batas atas (margin-top) paragraf.

e) Inden pada baris pertama

Kadang-kadang kita ingin membuat inden pada baris pertama tiap paragraf. Cara berikut dapat kita lakukan.

```
p { text-indent: 2em; margin-top: 0; margin-bottom: 0; }
```

Cara tersebut akan membuat inden pada baris pertama paragraf sejauh 2 em dan memberikan jarak antar paragraf.

f) Mengatur Format Font.

➤ Model huruf

Model yang umum dipakai adalah teks miring (italik) atau tebal (bold). Umumnya program browser mempergunakan tag em untuk huruf italic dan tag strong untuk huruf tebal. Misalnya kita ingin menuliskan kode em agar huruf tampil berbentuk italic dan tebal dan menuliskan kode strong untuk huruf tebal uppercase, perintahnya dituliskan sebagai berikut :

```
em { font-style: italic; font-weight: bold; }  
strong { text-transform: uppercase; font-weight: bold; }
```

bila gagal dapat ditambahkan perintah ini :

```
h2 { text-transform: lowercase; }
```

➤ Mengatur ukuran huruf

Kebanyakan program browser mempergunakan huruf yang lebih besar untuk heading yang penting sifatnya. Bila kita menimpa ukuran defaultnya, kita menempuh resiko yaitu huruf menjadi tampak lebih kecil khususnya bila kita mempergunakan ukuran yang kita tambahkan tersebut dengan ukuran titik (point). Karenanya kita disarankan untuk melakukan pengaturan ukuran huruf dengan ukuran yang sama. Contoh berikut mengatur ukuran heading dalam persen relatif terhadap ukuran teks normal.

```
h1 { font-size: 200%; }  
h2 { font-size: 150%; }  
h3 { font-size: 100%; }
```

➤ Mengatur jenis huruf

Bisa saja jenis huruf favorit kita tidak bisa ditampilkan oleh berbagai jenis browser. Untuk mengatasi hal ini kita dapat menuliskan beberapa jenis huruf yang tidak dapat ditampilkan oleh hampir semua browser. Ada beberapa jenis huruf generik yang dijamin cocok, sehingga kita kami sarankan untuk mengakhiri daftar perintah HTML kita dengan salah satu jenis huruf berikut : serif, sans-serif, cursive, fantasy, atau monospace, contoh :

```
body { font-family: Verdana, sans-serif; }  
h1,h2 { font-family: Garamond, Times New Roman, serif;  
}
```

Dalam contoh ini heading penting akan ditampilkan dalam bentuk huruf Garamond, bila gagal maka akan ditampilkan dalam bentuk Times New Roman, dan bila juga masih tidak dapat tampil, maka akan tampil sebagai huruf serif. Teks paragraf akan ditampilkan dengan huruf Verdana atau bila masih tidak tampil juga, maka browser masih dapat menampilkannya dengan jenis huruf standar sans-serif.

➤ Cara menghindari masalah huruf dan batas tepi halaman web

Pertama pergunakan elemen p untuk mencegah teks pada body tulisan yang tidak dapat ditampilkan dengan baik.

Contoh :

```
<h2>Spring in Wiltshire</h2>  
Blossom on the trees, bird song and the sound of lambs  
bleating in the fields.
```

Teks yang ditulis mengikuti sebuah heading dapat menimbulkan akibat tampilan jenis huruf yang berbeda pada beberapa jenis

program browser. Kita disarankan segera menutup teks pada paragraf tersebut.

Contoh :

```
<h2>Spring in Wiltshire</h2>
<p>Blossom on the trees, bird song and the sound of
lambs bleating in the fields.</p>
```

Kedua penggunaan selalu elemen pre ketika kita menuliskan pengaturan jenis huruf yang kita gunakan.

```
pre { font-family: monospace; }
```

Ketiga penggunaan elemen p dan ul pada waktu mengatur jenis huruf untuk heading. Khususnya ketika kita melakukan pengaturan border atau warna halaman web kita dengan elemen div. Beberapa jenis program browser tidak dapat melakukan pengaturan huruf dengan baik dan cenderung lupa sehingga huruf heading kita tampak menjadi huruf standar saja. Untuk menghindari hal ini kita dapat menuliskan perintah HTML sebagai berikut :

```
h1,h2,h3,h4,h5,p,ul { font-family: sans-serif; }
```

g) Menambahkan border dan latar belakang.

Kita dapat dengan mudah menambahkan border disekitar heading, daftar (list), atau paragraf atau sekelompok heading, list dan paragraph secara tertutup dengan mempergunakan elemen div. Contoh :

```
div.box{ border: solid; border-width: thin; width: 100% }
```

Catatan : tanpa property "width" beberapa browser akan menempatkan tepi kanan terlalu jauh ke arah kanan. Untuk mencegah hal ini kita dapat menuliskan perintah HTML-nya sebagai berikut :

```
<div class="box"> The content within this DIV
element will be enclosed in a box with a thin line around
it. </div>
```

Ada sedikit jenis border yaitu : dotted, dashed, solid, double, groove, ridge, inset dan outset. Lebar border diatur dengan mempergunakan property "border-width". Nilai dari property ini yaitu thin, medium dan thick yang tampak setipis ukuran 0.1em. Property "border-color" memungkinkan kita mengatur warna.

Sebuah efek yang cantik dapat dilakukan dengan memberikan warna latar belakang kotak dengan warna tebal (solid color) atau dengan hamburan gambar ("tile image"). Untuk melakukan ini kita perlu mempergunakan property "background". Kita dapat mengikuti perintah berikut ini.

```
div.color {
background: rgb(204,204,255) ;
padding: 0.5em;
border: none;
}
```

Tanpa pengaturan property border, biasanya program browser hanya akan menampilkan warna standar saja. Property padding memberikan beberapa ruangan diantara tepi-tepi daerah berwarna dan teks yang ada didalamnya.

Kita dapat mengatur nilai property padding dengan menambahkan padding-left, padding-top, padding-right dan padding-bottom. Pengaturan ini dituliskan misalnya sebagai :

```
padding-left: 1em.
```

Anggaplah kita menginginkan border pada satu sisi halaman web saja. Kita dapat melakukan pengontrolan tiap sisi border dengan memberikan keterangan border-left, border-top, border-right dan border-bottom.

Contoh :

```
p.changed {
padding-left: 0.2em;
border-left: solid;
border-right: none;
border-top: none;
border-bottom: none;
border-left-width: thin;
border-color: red;
}
```

Susunan perintah diatas memberikan efek pada tampilan web kita dengan border berwarna merah disisi sebelah kiri.

h) Mengatur warna

Contoh berikut adalah perintah pengaturan warna.

```
body {
color: black;
background: white;
}
strong { color: red }
```

Model pengaturan diatas memberikan warna hitam teks (default) dan latar belakang putih, tetapi memiliki elemen strong pada warna merah. Ada 16 buah nama warna standar yang dijelaskan dalam uraian berikut. Selain itu kita dapat juga mempergunakan nilai desimal untuk warna merah, hijau dan biru, dan masing-masing memiliki interval antara 0 sampai 255. Misalnya rgb(255, 0, 0) akan memberikan warna merah di layar monitor. Kita juga dapat mempergunakan angka

hexadecimal yang dimulai karakter # yang diikuti enam angka hexadesimal sebagai pengaturan warna. Sebuah pengubah juga diberikan dibawah ini agar kita dapat melakukan pengubahan nilai dari RGB ke nilai hexadesimal.

➤ Mengatur warna link

Kita dapat mempergunakan CSS untuk mengatur warna hypertext link, dengan warna yang berbeda untuk link yang belum pernah kita akses, link yang pernah kita akses dan link yang akan kemudian kita akses serta link yang aktif. Kita bahkan dapat mengatur warnanya pada saat kursor mouse berada diatas daerah yang akan dilink. Hal ini dapat kita tuliskan dalam bentuk perintah berikut :

```
:link { color: rgb(0, 0, 153) } /* untuk warna link
yang belum
diakses */
:visited { color: rgb(153, 0, 153) } /* untuk warna
link yang telah diakses */
:active { color: rgb(255, 0, 102) } /* untuk warna
link ketika link diklik */
:hover { color: rgb(0, 96, 255) } /* untuk warna link
ketika mouse diatasnya*/
```

Kadang-kadang kita ingin memperlihatkan link hypertexts tanpa garis bawah. Kita dapat melakukan ini dengan memberikan property text-decoration atau none, contoh :

```
a.plain { text-decoration: none }
```

Contoh berikut juga menampilkan link yang tidak bergaris bawah.

```
This is <a class="plain" href="what.html">not
underlined</a>
```

Kebanyakan orang ketika mereka melihat garis bawah dibawah sebuah link selalu mengira itu adalah bagian teks yang diberi link. Umumnya orang memberikan warna biru pada teks yang diberi link ke halaman atau alamat internet lain. Kita disarankan untuk meninggalkan warna link ini apabila warna latar belakang menyebabkan teks kita yang diberi link jadi sulit terbaca.

21

Pemrograman Web

TUJUAN

Setelah anda mempelajari bab ini diharapkan anda akan mampu :

- Memahami aturan-aturan umum dalam pemrograman web (PHP).
- Menerapkan penggunaan variable, tipe data dan operator pada PHP.
- Menerapkan algoritma pemrograman pada pemrograman web.
- Membuat aplikasi berbasis web.

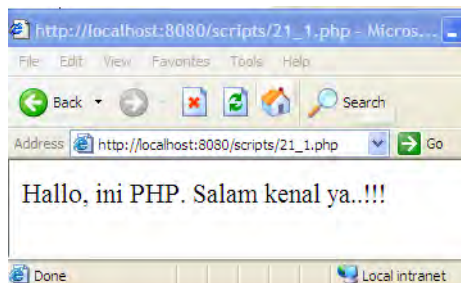
Pemrograman web pada buku ini dikhususkan pada pemrograman dengan menggunakan bahasa pemrograman PHP. Hal ini karena PHP merupakan bahasa pemrograman web yang paling populer dan banyak digunakan pada website di internet. Selain itu factor open source juga menjadi pertimbangan yang tidak dapat diabaikan.

21.1 PENGENALAN PADA PHP

PHP adalah bahasa scripting dijalankan pada *server side*. Artinya semua sintaks yang kita berikan akan sepenuhnya dijalankan pada server sedangkan yang dikirimkan ke browser hanya hasilnya saja.

Contoh 21.1:

```
<?php
    echo "Hallo, ini PHP. Salam kenal ya..!!!";
?>
```



Gambar 21.1. Hasil eksekusi contoh 21.1.

21.1.1 Aturan-aturan umum pada PHP

Beberapa hal yang perlu diperhatikan dalam PHP, antara lain:

- Suatu script (kode) PHP dapat dikenali bila diapit oleh salah satu diantara tiga bentuk berikut :

```
<?php
    // script PHP;
?>
```

Atau

```
<?
    // Script PHP;
?>
```

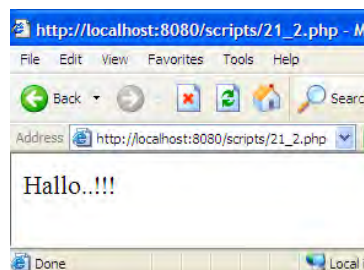
Atau

```
<SCRIPT LANGUAGE="php">
    // Script PHP;
</SCRIPT>
```

- Setiap baris kode PHP diakhiri dengan tanda titik koma/semicolon (;). Semicolon adalah pemisah dan digunakan untuk membedakan sekelompok instruksi dengan yang lain.
- Komentar dalam PHP dapat dilakukan dengan menggunakan:
 - Tanda // atau tanda # jika komentar dalam satu baris menggunakan.
 - Tanda /* ... */ jika komentar yang ingin ditulis lebih dari satu baris.

Contoh 21.2:

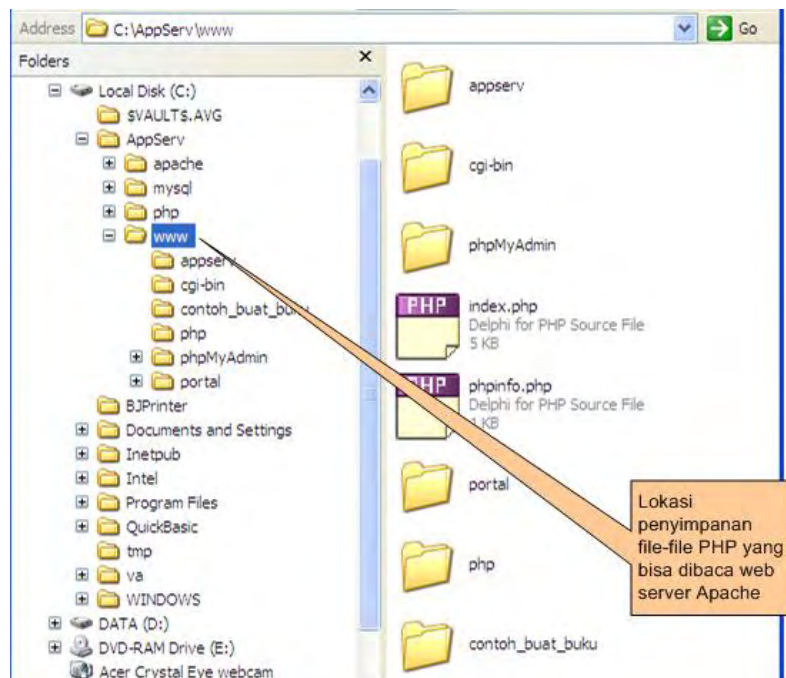
```
<?php
    echo "Hallo...!!!"; // ini komentar pendek
    /* ini komentar panjang
    Lebih dari satu baris lho */
?>
```



Gambar 21.2. Hasil eksekusi contoh 21.2.

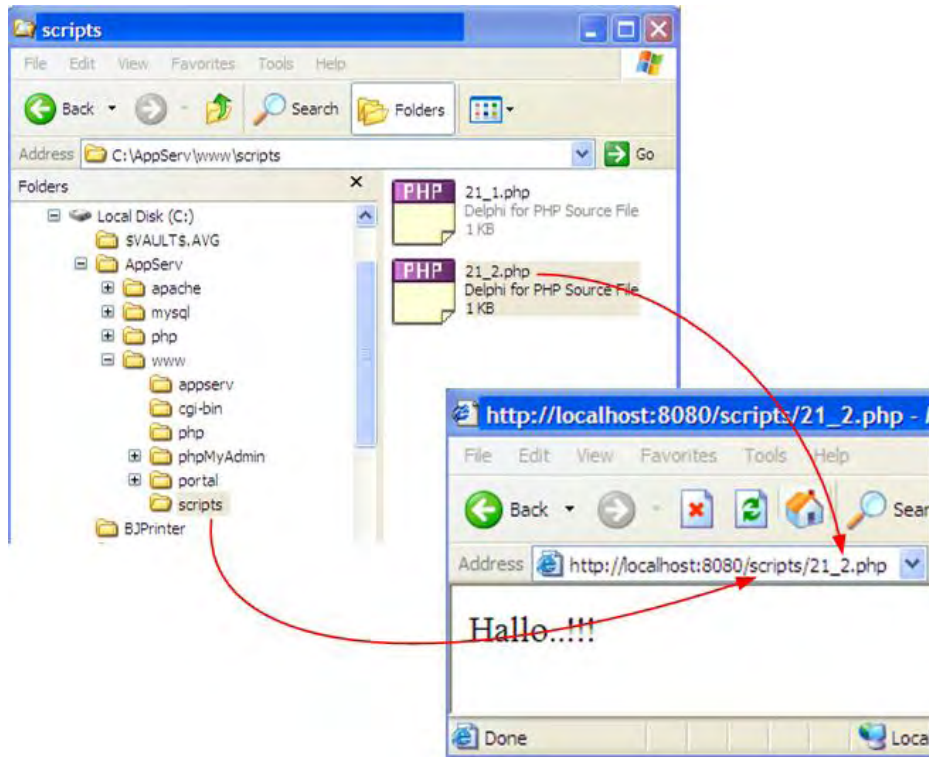
Pada Gambar 21.2 tampak bahwa komentar yang dibuat pada script contoh 21.2 tidak akan ditampilkan dalam browser. Karena semua yang ada dibelakang tanda // atau diantara tanda /* ... */ tidak akan diproses oleh web server.

- Disimpan dengan ekstensi .php dan diletakkan dalam direktori yang bisa dibaca oleh web server. Seperti disebutkan, script PHP adalah script yang berjalan di sisi server, sehingga lokasi penyimpanan file-file PHP menjadi sangat ditentukan dari setting web servernya. Sebagai contoh pada beberapa distribusi Linux file-file PHP disimpan pada direktori `/usr/var/www/` di beberapa distribusi yang lain disimpan di direktori `/svr/www/`. Pada Microsoft Windows demikian juga. Lokasi ini dapat diubah dengan cara mengedit file konfigurasi Apache Web Server (`httpd.conf`). Contoh lokasi penyimpanan file-file PHP dapat dilihat pada Gambar 21.3.



Gambar 21.3. Lokasi direktori yang bisa dibaca web server.

- Untuk menjalankan file-file berekstensi .php tidak dapat dilakukan dengan double klik pada nama file seperti pada file berekstensi .htm atau .html. Tetapi harus dipanggil menggunakan web browser dengan alamat <http://namaserver/namafile.php>. Perhatikan ilustrasi berikut:



Gambar 21.4. Cara menjalankan file dengan ekstensi .php.

Pada Gambar 21.4, nama file nya adalah 21_2.php dan disimpan pada direktori c:\appserv\www\scripts\. Maka untuk memanggilnya adalah dengan mengetikkan http://localhost:8080/scripts/21_2.php. Localhost adalah nama server nya dan 8080 adalah port dimana Apache dijalankan.

21.1.2 PHP dan HTML

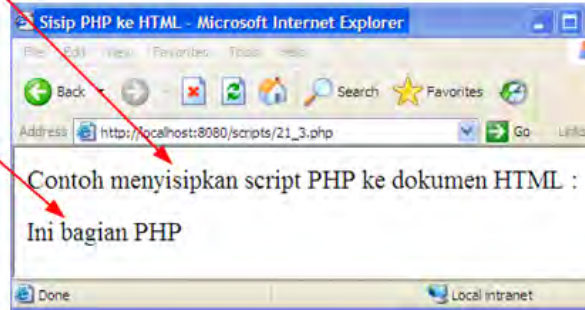
PHP tidak bisa melakukan format tampilan di browser seperti mengatur warna font atau background, mengatur jenis huruf dan ukurannya, atau membuat layout. HTML yang bisa melakukan hal-hal tersebut. Oleh karena itu untuk mendapatkan tampilan hasil yang baik, PHP akan selalu digabungkan dengan HTML dengan cara :

- menyisipkan (embed) skrip PHP pada dokumen HTML (Gambar 21.5)

```

<html>
<head>
<title>
Sisip PHP ke HTML
</title>
</head>
<body>
Contoh menyisipkan script PHP ke dokumen HTML :<P>
<?php
echo "Ini bagian PHP";
?>
</body>
</html>

```



Gambar 21.5. Menyisipkan skrip PHP pada dokumen HTML.

- menyisipkan tag HTML ke dokumen PHP (Gambar 21.6)

```

<?php
echo "<TITLE>Sisip HTML ke PHP</TITLE>";
echo "<H2>Pemrograman PHP Mengasyikkan</H2>";
print "<HR>";
?>

```



Gambar 21.6. Menyisipkan tag HTML ke dokumen PHP.

21.2 VARIABEL DAN TIPE DATA

21.2.1 Variabel

Ada beberapa aturan yang diikuti berkenaan dengan penggunaan nama variable, yaitu:

- Dimulai dengan tanda \$.

- Hanya ada tiga jenis karakter yang dapat digunakan dalam nama variable yaitu huruf, angka dan garis bawah.
- Karakter pertama harus huruf atau garis bawah (_), karakter berikutnya boleh huruf, angka, atau garis bawah.
- PHP bersifat case sensitive atau membedakan huruf besar dan huruf kecil. \$Nama dan \$NAMA merupakan dua variable yang berbeda.

Untuk dapat menggunakan variabel, ada dua langkah yang harus dilakukan, **deklarasi** dan **inisialisasi**.

- **Deklarasi variabel**

Deklarasi variabel bisa disebut juga memperkenalkan atau mendaftarkan variabel ke dalam program. Dalam php, deklarasi variabel seringkali digabung dengan inisialisasi.

Contoh 21.3:

```
$namaSiswa
$jumlahBarang
$totalHarga
```

- **Inisialisasi variabel**

Inisialisasi variabel adalah mengisi nilai untuk pertama kalinya ke dalam variabel.

Contoh 21.4:

```
$namaDepanSiswa = "Robert";
$namaBelakangSiswa = "Plant";
$jumlahBarang = 3;
$totalHarga = 1000;
```

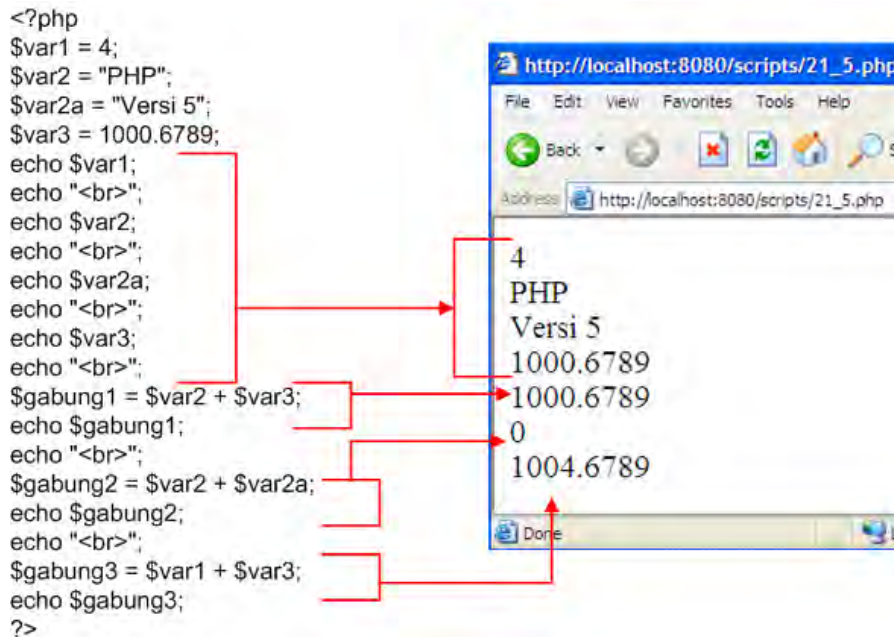
21.2.2 Tipe data

Dalam bahasa pemrograman yang lain, ada bermacam-macam tipe data, misalnya integer(bilangan bulat), float(bilangan pecahan), char(karakter angka dan huruf), string(kumpulan huruf atau kata), dan berbagai tipe lainnya. PHP hanya mengenal dua tipe data sederhana; numerik dan literal. Ditambah dengan dua tipe data yang tidak sederhana, yaitu array dan object.

- Tipe Numerik digunakan menyimpan data dalam bentuk angka.
 - data bilangan bulat (**Integer**) dengan jangkauan dari -2 milyar sampai +2 milyar. Contoh: 3, 7, 20.
 - bilangan pecahan (**Float**). Contoh: 3.50, 4.12, 100.29876.

- Tipe literal digunakan untuk menyimpan data berupa kumpulan huruf, kata, dan angka.
- Tipe boolean, yang dikenal dalam bahasa program yang lainnya, tidak ada dalam PHP. Untuk menguji benar salah (true false), kita menggunakan tipe data yang tersedia. FALSE dapat digantikan oleh integer 0, double 0.0 atau string kosong, yaitu "". Selain nilai itu, semua dianggap TRUE.

Perhatikan Gambar berikut ini.



Gambar 21.7. Contoh penggunaan variable.

Dari gambar 21.7 kita dapat melihat beberapa aturan yang berlaku dalam tipe data, yaitu:

- Tipe literal digunakan untuk menyimpan data berupa kumpulan huruf, kata, dan angka.
- Satu nama variabel dapat digunakan untuk menyimpan berbagai jenis data. Misalnya data numerik yang dapat dioperasikan secara matematika.

Contoh 21.5:

```

$jumlahBarang = 3;
$harga = 1000;
$pembayaran = $jumlahBarang * $harga;

```


pada contoh di atas, variabel pembayaran akan menyimpan nilai 3000.

- Sedangkan data non numerik (disebut juga data literal) tidak dapat dioperasikan secara matematika.

Contoh 21.6:

```
$nama = $namaDepan + $namaBelakang;
```

variabel nama akan menyimpan gabungan dari dua variabel, yaitu "Robert Plant".

Secara umum, data literal ditandai dengan pasangan `"` dan `"`. Data numerik tidak dikelilingi oleh `"` dan `"`. Tetapi biasanya PHP akan secara otomatis mengubah tipe data sesuai kebutuhan.

Contoh 21.7:

```
$jalan = "Gubeng Kertajaya";  
$noRumah = 29;  
$blok = 4c;  
$jumlahPengehuni = 3;  
$alamat = $jalan + $noRumah;  
$hasil = $noRumah + $jumlahPengehuni;  
$hasilAneh = $blok + $noRumah;
```

Pada sampel kode di atas, variabel alamat akan menyimpan nilai **Gubeng Kertajaya 29**. PHP secara otomatis mengubah tipe data variabel `noRumah` (numerik) menjadi literal. Variabel alamat akan bertipe literal. Variabel hasil akan menyimpan nilai 32, yaitu penjumlahan dari 29 dan 3.

Perhatikan, konversi otomatis ini kadang berjalan secara tidak semestinya. Ini dapat dilihat dari variabel `hasilAneh` yang akan menyimpan nilai 7. PHP mengambil nilai numerik dari variabel `blok`, yaitu 4, kemudian menambahkannya dengan isi variabel `jumlahPengehuni`. Hasil akhirnya adalah $4 + 3$, yaitu 7.

21.2.3 Operator

Operator digunakan untuk mengoperasikan nilai. PHP memiliki beberapa jenis operator, yaitu aritmatika, assignment, perbandingan dan logika. Macam-macam operator dapat dilihat dalam Tabel berikut ini.

Tabel 21.1 Jenis-jenis operator

Jenis Operator	Operator	Deskripsi
Aritmatika	+	Penambahan
	-	Pengurangan
	*	Perkalian
	/	Pembagian
	%	Modulus (sisanya bagi)
	++	Inkrementasi (ditambah 1)
	--	Dekrementasi (dikurang 1)
Assignment	=	$x=y \rightarrow x=y$
	+=	$x+=y \rightarrow x=x+y$
	-=	$x-=y \rightarrow x=x-y$
	*=	$x*=y \rightarrow x=x*y$
	/=	$x/=y \rightarrow x=x/y$
	%=	$x\%=y \rightarrow x=x\%y$
Perbandingan	=	Sama dengan
	!=	Tidak sama dengan
	>	Lebih besar dari
	<	Lebih kecil dari
	>=	Lebih besar sama dengan
	<=	Lebih kecil sama dengan
Logika	&&	Dan
		Atau
	!	Tidak

21.3 TRANSLASI ALGORITMA KE BAHASA PEMROGRAMAN WEB

Tidak jauh berbeda dengan bahasa-bahasa pemrograman lain, PHP juga menganut tiga struktur dasar algoritma yang strukturnya berurutan, struktur pemilihan dan struktur pengulangan.

21.3.1 Struktur berurutan

Struktur berurutan merupakan struktur paling mudah karena program akan mengalir baris-perbaris.

Contoh 21.8.:

```
<HTML>
<HEAD>
<TITLE>Contoh Konstanta Simbolis</TITLE>
</HEAD>
<BODY>
<?php
    define("PHI", 3.14);

    $jari_jari = 10;
```

```

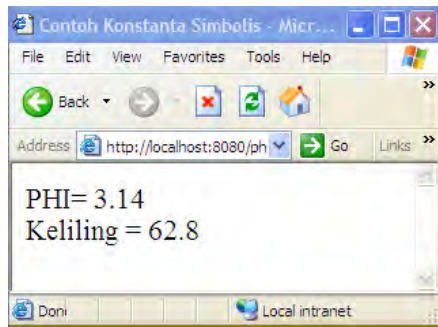
$keliling = 2 * PHI * $jari_jari;

printf("PHI= %s<BR>\n", PHI);
printf("Keliling = %s <BR>\n", $keliling);
?>

</BODY>
</HTML>

```

Hasil eksekusi kode program di atas adalah sebagai berikut:



21.3.2 Struktur pemilihan

Sama seperti pada Visual Basic struktur pemilihan menggunakan konstruksi IF untuk melakukan eksekusi suatu statement secara bersyarat. Cara penulisannya adalah sebagai berikut:

```

if (syarat)
{
    statement
}

```

atau:

```

if (syarat)
{
    statement
}
else
{
    statement lain
}

```

atau:

```

if (syarat pertama)
{
    statement pertama
}

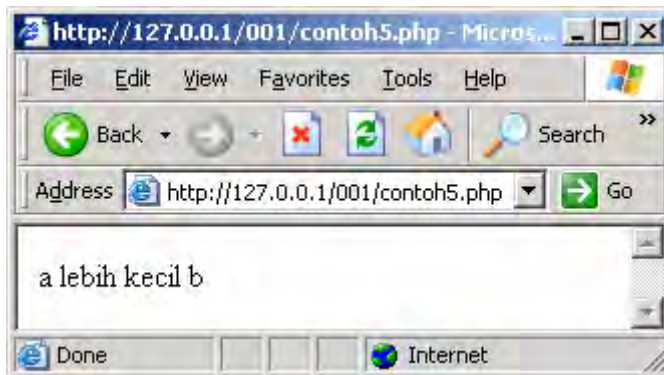
```

```
elseif (syarat kedua)
{
statement kedua
}
else
{
statement lain
}
```

Contoh 21.9.:

```
<?php
$a=4;
$b=9;
if ($a>$b)
{
echo("a lebih besar dari pada b");
}
elseif ($a<$b)
{
echo("a lebih kecil b");
}
else
{
echo("a sama dengan b");
}
?>
```

Hasilnya adalah:



Pernyataan switch dapat digunakan untuk memiliki salah satu dari beberapa kumpulan kode untuk dijalankan. Sintaksnya sebagai berikut:

```
Switch(kondisi) {  
Case konstanta 1:
```

```

        Pernyataan 1;
        Break;
Case konstanta 1:
        Pernyataan 1;
        Break;
Default:
        Pernyataan default;
}

```

Contoh 21.10.:

```

<?php
$a=2;
switch($a)
{
    case 1:
        echo("Nilai variable a adalah satu");
        break;
    case 2:
        echo("Nilai variable a adalah dua");
        break;
    case 3:
        echo("Nilai variable a adalah tiga");
        break;
}
?>

```

Hasilnya adalah:



21.3.3 Struktur pengulangan

Struktur digunakan untuk menjalankan berulang kelompok kode yang sama sejumlah tertentu. Php memiliki beberapa perintah looping, antara lain:

- While

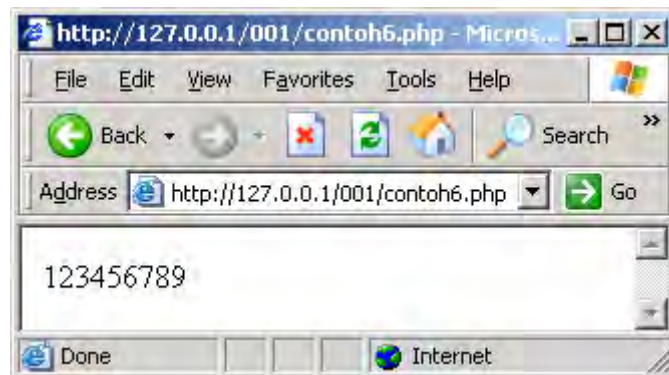
Perulangan dengan menjalankan kelompok kode jika dan hanya jika kondisi tertentu benar. Sintaksnya sebagai berikut:

```
For (nilai awal; nilai akhir; peningkatan/penurunan)  
{  
pernyataan yang dijalankan;  
}
```

Contoh 21.11:

```
<?php  
$a=1;  
while ($a<10)  
{  
    echo ($a) ;  
    $a++;  
}  
?>
```

Hasilnya adalah:



- Do ... while

Perulangan dengan menjalankan kelompok kode sekali, dan mengulanginya jika kondisi tertentu adalah benar. Sintaksnya sebagai berikut:

```
Do  
{  
pernyataan yang dijalankan;  
}while (kondisi)
```

Contoh 21.12.

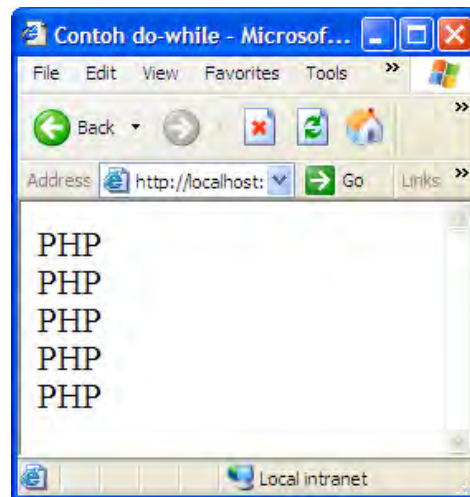
```
<HTML>  
<HEAD>
```

```

<TITLE>Contoh do-while</TITLE>
</HEAD>
<BODY>
<?php
    $bilangan = 0;
    do {
        print("PHP<BR>\n");
        $bilangan++;
    } while ($bilangan < 5);
?>
</BODY>
</HTML>

```

Hasil eksekusi adalah sebagai berikut:



- For

Perulangan dengan menjalankan kelompok kode sejumlah tertentu. Sintaksnya sebagai berikut:

```

For (nilai awal; nilai akhir; peningkatan/penurunan)
{
    pernyataan yang dijalankan;
}

```

Contoh 21.1`3:

```

<?php
    for ($a=0;$a<10;$a++)
    {
        echo("Nilai A = ");
    }

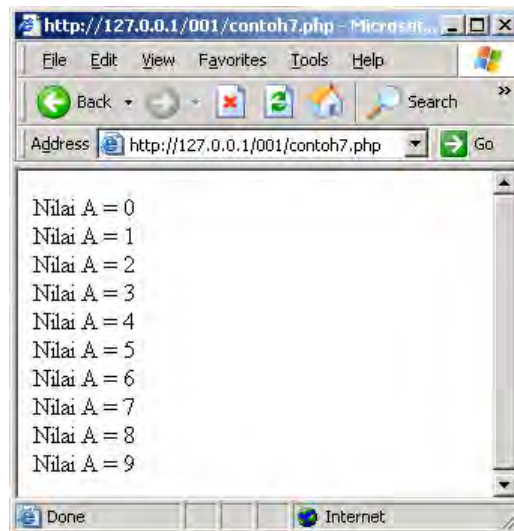
```

```

        echo (" $a" ) ;
        echo (" <br>" ) ;
    }
?>

```

Hasilnya adalah



- Foreach

Perulangan dengan menjalankan kelompok kode untuk setiap elemen dalam bentuk larik. Sintaksnya sebagai berikut;

```

Foreach (array sebagai nilai)
{
    pernyataan yang dijalankan;
}

```

21.4. REQUIRE DAN INCLUDE

Statement Require digunakan untuk membaca nilai variable dan fungsi-fungsi dari sebuah file lain. Cara penulisan statement Require adalah:

```
require (namafile);
```

Statement Require ini tidak dapat dimasukkan didalam suatu struktur looping misalnya while atau for. Karena hanya memperbolehkan pemanggilan file yang sama tersebut hanya sekali saja.

Contoh:

File contoh9.php:

```
<?php
$a="Saya sedang belajar PHP";

function tulistebal($teks)
{
echo("<b>$teks</b>");
}
?>
```

File contoh10.php:

```
<?php
require("contoh9.php");
tulistebal("Ini adalah tulisan tebal");
echo("<br>");
echo($a);
?>
```

Hasilnya adalah:



Statement Include akan menyertakan isi suatu file tertentu. Include dapat diletakkan didalam suatu looping misalkan dalam statement for atau while.

Contoh:

File contoh11.php:

```
<?php
echo("<div>-----</div>");
echo("PHP adalah bahasa scripting<br>");
```

```
echo ("-----<br>");  
echo ("<br>");  
?>
```

File contoh12.php:

```
<?php  
for ($b=1; $b<5; $b++)  
{  
    include ("contoh11.php");  
}  
?>
```

Hasilnya adalah:



21.5 FUNGSI

- **FUNGSI STRING**

Fungsi string digunakan memanipulasi string untuk berbagai macam kebutuhan. Disini akan dibahas beberapa fungsi string yang sering digunakan dalam membuat program aplikasi web.

AddSlashes

Digunakan untuk menambahkan karakter backslash (\) pada suatu string. Hal ini penting digunakan pada query string untuk database, misalkan pada MySQL. Beberapa karakter yang akan ditambahkan tanda backslashes adalah karakter tanda petik satu ('), karakter petik dua ("), backslash (\) dan karakter NULL.

Sintaks:

```
addslashes(string)
```

StripSlashes

Digunakan untuk menghilangkan karakter backslash (\) pada suatu string.

Sintaks:

```
string stripslashes(string)
```

Crypt

Digunakan untuk meng-encrypt dengan metode DES suatu string. Fungsi ini sering digunakan untuk mengacak string password sebelum disimpan dalam database. Dalam penggunaan fungsi crypt ini dapat ditambahkan parameter string 'salt'. Parameter 'salt' ini ditambahkan untuk menentukan basis pengacakan. 'Salt' string terdiri atas 2 karakter. Jika 'salt' string tidak ditambahkan pada fungsi crypt maka PHP akan menentukan sendiri 'salt' string tersebut secara acak.

Sintaks:

```
crypt(string [ , salt ] )
```

Echo

Digunakan untuk mencetak isi suatu string atau argumen.

Sintaks:

```
echo( string argumen1, string argumen2 , ... )
```

Explode

Digunakan untuk memecah-mecah suatu string berdasarkan tanda pemisah tertentu dan memasukkan hasilnya kedalam suatu variable array.

Sintaks:

```
explode(string pemisah , string [, int limit] )
```

Contoh:

```
$namahari = "minggu senin Selasa Rabu Kamis Jumat  
Sabtu";  
$hari = explode(" ", $namahari);
```

Implode

Kegunaan fungsi ini adalah kebalikan daripada fungsi explode. Fungsi implode digunakan untuk menghasilkan suatu string dari masing-masing elemen suatu array. String yang dihasilkan tersebut dipisahkan oleh suatu string telah yang ditentukan sebelumnya.

Sintaks:

```
implode(string pemisah , array)
```

Strip_Tags

Digunakan untuk menghilangkan kode-kode tag HTML pada suatu string.

Sintaks:

```
striptags(string [, string tags yang tidak  
dihilangkan] )
```

StrLen

Digunakan untuk menghitung jumlah karakter suatu string.

Sintaks:

```
strlen(string)
```

StrPos

Digunakan untuk mencari posisi pertama suatu sub string pada suatu string. Fungsi ini biasanya digunakan untuk mencari suatu sub string didalam suatu string.

Sintaks:

```
strlen(string , sub string)
```

Str_Repeat

Digunakan untuk mengulang isi suatu string.

Sintaks:

```
str_repeat(string , int jumlah perulangan)
```

StrToLower

Digunakan untuk merubah suatu string menjadi huruf kecil (*lowercase*).

Sintaks:

```
strtolower(string)
```

StrToUpper

Digunakan untuk merubah suatu string menjadi huruf besar (*uppercase*).

Sintaks:

```
strtoupper(string)
```

SubStr

Digunakan untuk mengambil suatu sub string dengan panjang tertentu dari suatu string pada posisi tertentu pula.

Sintaks:

substr(*string*, *int posisi* , *int posisi*)

Contoh:

```
substr("abcdefg",0,3); // menghasilkan string "abc"  
substr("abcdefg",3,2); // menghasilkan string "de"
```

SubStr_Count

Digunakan untuk menghitung jumlah sub string dalam suatu string

Sintaks:

substr_count(*string* , *string substring*)

Contoh:

```
substr_count("This is a test","is"); // menghasilkan  
nilai 2
```

UCFirst

Digunakan untuk mengganti karakter pertama pada suatu string menjadi huruf besar.

Sintaks:

ucfirst(*string*)

UCWords

Digunakan untuk mengganti karakter pertama pada setiap kata dalam suatu string menjadi huruf besar.

Sintaks:

ucwords(*string*)

- **FUNGSI DATE**

Digunakan untuk mengambil tanggal dan jam. Hasil dari fungsi ini adalah sebuah string yang berisi tanggal/jam sesuai dengan format yang diinginkan. Format yang dikenal dalam fungsi date ini adalah sebagai berikut:

- a - "am" or "pm"
- A - "AM" or "PM"
- B - Swatch Internet time
- d - day of the month, 2 digits with leading zeros; i.e. "01" to "31"
- D - day of the week, textual, 3 letters; i.e. "Fri"
- F - month, textual, long; i.e. "January"

- g - hour, 12-hour format without leading zeros; i.e. "1" to "12"
- G - hour, 24-hour format without leading zeros; i.e. "0" to "23"
- h - hour, 12-hour format; i.e. "01" to "12"
- H - hour, 24-hour format; i.e. "00" to "23"
- i - minutes; i.e. "00" to "59"
- I (capital i) - "1" if Daylight Savings Time, "0" otherwise.
- j - day of the month without leading zeros; i.e. "1" to "31"
- l (lowercase 'l') - day of the week, textual, long; i.e. "Friday"
- L - boolean for whether it is a leap year; i.e. "0" or "1"
- m - month; i.e. "01" to "12"
- M - month, textual, 3 letters; i.e. "Jan"
- n - month without leading zeros; i.e. "1" to "12"
- s - seconds; i.e. "00" to "59"
- S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd"
- t - number of days in the given month; i.e. "28" to "31"
- T - Timezone setting of this machine; i.e. "MDT"
- U - seconds since the epoch
- w - day of the week, numeric, i.e. "0" (Sunday) to "6" (Saturday)
- Y - year, 4 digits; i.e. "1999"
- y - year, 2 digits; i.e. "99"
- z - day of the year; i.e. "0" to "365"
- Z - timezone offset in seconds (i.e. "-43200" to "43200")

Sintaks:

date(*string format*)

Contoh:

```
date("Y-m-d"); // menghasilkan "2001-07-28"
date("l, j F Y"); // menghasilkan "Saturday, 28
July 2001"
date("H:i:s"); // menghasilkan "20:15:07"
```

• FUNGSI MAIL

Digunakan untuk mengirimkan e-mail ke alamat e-mail tertentu.

Sintaks:

mail(*string tujuan* , *string subject* , *string isi* [, *string header*]);

Contoh:

```
$pengirim = "From: saya@email.com";
$tujuan = "anonkuncoro@yahoo.com";
$subject = "Pemberitahuan";
```

```
$isi = "Ini adalah percobaan pengiriman e-mail  
dengan menggunakan PHP";  
mail($to,$subject,$isi,$pengirim);
```

21.6 BEKERJA DENGAN SISTEM MANAJEMEN BASIS DATA

Menyimpan data dalam file biasa memiliki banyak keterbatasan. Semakin besar ukuran file, pencarian data menjadi lebih sulit. File biasa juga tidak memiliki kemampuan untuk mengolah data, misalnya menghitung total nilai, rata-rata, dan lain sebagainya. Dan yang lebih penting, adanya keterbatasan untuk mengendalikan akses terhadap data. Kita tidak dapat menentukan siapa yang boleh dan siapa yang tidak boleh mengakses data. Siapa yang boleh membaca dan tidak boleh menulis, dan sebagainya. Karena itu, sekarang kita akan menggunakan media penyimpanan data yang lebih mutakhir, yaitu database.

Operasi data dalam database umumnya mengikuti pola yang sama, yaitu melalui rangkaian langkah sebagai berikut :

- Membuka sambungan (koneksi) dengan database server
- Memilih dan membuka database yang diinginkan
- Mengirim perintah untuk mengambil/mengubah/menghapus data
- Mengakses hasil pengambilan data
- Mengakhiri sambungan

21.6.1 Koneksi

Untuk membuka sambungan, kita membutuhkan alamat server serta username dan password untuk database. Setelah itu, kita membuka sambungan dengan perintah sebagai berikut :

```
$dbServer = "localhost";  
$dbUser = "username";  
$dbPass = "passwd";  
$dbConn = mysql_pconnect($dbServer, $dbUser, $dbPass);
```

sambungan dengan database disimpan dalam variabel \$dbConn untuk digunakan pada langkah-langkah selanjutnya.

Setelah sambungan ke server basisdata terbuka, maka kita dapat memilih basis data mana yang akan kita pilih dengan menggunakan perintah berikut:

```
$dbName = "nama_basis_data";  
mysql_select_db($dbName);
```

sekarang database telah siap untuk digunakan. Kita dapat melangkah ke tahap selanjutnya, yaitu memasukkan data.

21.6.2 Manipulasi data

- Memasukkan/menambah data

Perintah untuk memasukkan data dibuat dengan menggunakan aturan SQL, dijelaskan lebih lengkap pada bagian selanjutnya. Query (perintah) untuk memasukkan data adalah :

```
INSERT INTO userTable VALUES (
1,
'Robert',
'inipaswod',
'Robert Plant'
);
```

perintah tersebut dieksekusi di dalam script PHP sebagai berikut :

```
$query = "
INSERT INTO userTable VALUES (
1,
'Robert',
'inipaswod',
'Robert Plan'
);
";
$hasil = mysql_query($query);
if($hasil){
echo(mysql_affected_rows()." orang ditambahkan ke dalam
sistem");
}
```

Script kita harus mengambil data dari HTML Form (register.htm) dan memasukkannya ke dalam database. Untuk itu, kita akan membuat file register.php yang berisi kode sebagai berikut :

```
<?
$dbServer = "localhost";
$dbUser = "endy";
$dbPass = "test";
$dbConn = mysql_pconnect($dbServer, $dbUser, $dbPass);
$dbName = "phpTutorial";
mysql_select_db($dbName);
$query = "
INSERT INTO userTable
(UserName, Password, NamaLengkap)
VALUES (
```



```

'$user',
'$pass',
'$namaLengkap'
);
";
$hasil = mysql_query($query);
if($hasil){
echo(mysql_affected_rows()." orang ditambahkan ke dalam
sistem");
}
?>

```

- Mengakses hasil query

Setelah register.htm dieksekusi dengan sukses, kita telah memiliki satu user dalam database. Dengan demikian, kita dapat menguji halaman login yang telah kita buat. Untuk itu, kita perlu mengakses data dalam database dan mencocokkannya dengan input yang diberikan user. Langkah-langkah untuk mengakses data dalam database adalah :

Membuat sambungan database

- Memilih database
- Membuat query
- Menjalankan query
- Mengambil hasilnya
- Memproses hasil

Query yang akan digunakan adalah :

```
SELECT * FROM UserTable WHERE userName='input dari user'
```

Query ini akan dimasukkan ke dalam script menjadi :

```
$query = SELECT * FROM UserTable WHERE userName='$login'
```

Dan dieksekusi dengan kode :

```
$hasil = mysql_query($query);
```

Bila userName yang diinputkan user tidak ada dalam database, query akan mengembalikan 0 (nol) baris hasil. Bila ada, query akan menghasilkan array yang berisi data pada masing-masing kolom dalam database.

Untuk mengetahui jumlah hasil yang didapat dari query, digunakan kode sebagai berikut:

```
$jumlahHasil = mysql_num_rows($hasil);
```

Apabila hasilnya tidak nol (berarti username telah terdaftar dalam sistem), kita akan mengambil data password. Untuk itu, kita gunakan perintah:

```
$data = mysql_fetch_array($hasil);
```

data password ada dalam array, diakses dengan cara :
`$passDb = $data["Password"];`

Dengan demikian, keseluruhan script login.php akan menjadi :

```
<?
// variabel yang diperlukan untuk akses database
$user = "endy";
$pass = "test";
$db = "ContentManager";
$server = "localhost";
// membuat koneksi
$koneksi = mysql_connect($server, $user, $pass);
// memeriksa koneksi
if(!$koneksi){
echo("Koneksi ke database gagal");
exit;
}
// membuka database
mysql_select_db($db);
// membuat query
$query = "SELECT * FROM userTable WHERE
userName='".$login.'"";
// mengeksekusi query
$hasil = mysql_query($query);
// mengakses hasil
$jumlahHasil = mysql_num_rows($hasil);
if($jumlahHasil < 1){
echo("User $login tidak ada dalam database <br>");
}else{
echo("User $login ada dalam database<br>");
$data = mysql_fetch_array($hasil);
$passwordAsli = $data["password"];
if($password == $passwordAsli){
echo("Password untuk user $login benar<br>");
}else{
echo("Password untuk user $login salah<br>");
}
}
}
```

21.7 CONTOH-CONTOH APLIKASI PHP

- Aplikasi counter

Aplikasi web yang paling sederhana yang akan kita bahas adalah teknik pembuatan counter dengan menggunakan PHP. Counter yang kita buat ini adalah untuk menghitung berapa kali suatu halaman situs web telah ditampilkan. Untuk menyederhanakannya maka counter ditampilkan dalam bentuk teks bukan grafik.

Algoritma:

1. Bila suatu halaman web ditampilkan maka terlebih dulu dibaca isi file tertentu dan dibaca nilainya.
2. Tampilkan nilainya di layar browser
3. Tambahkan nilainya dengan 1
4. Simpan nilainya yang baru di file
5. Selesai

File counter.txt

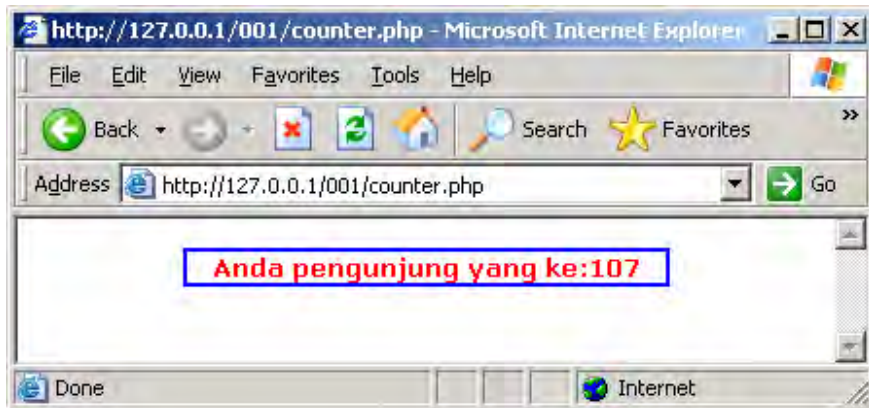
0

File counter.php:

```
<?
$filecounter="counter.txt";
$fl=fopen($filecounter,"r+");
$hit=fread($fl,filesize($filecounter));

echo("<table width=250 align=center border=1
cellspacing=0 cellpadding=0
bordercolor=#0000FF><tr>");
echo("<td width=250 valign=middle align=center>");
echo("<font face=verdana size=2 color=#FF0000><b>");
echo("Anda pengunjung yang ke:");
echo($hit);
echo("</b></font>");
echo("</td>");
echo("</tr></table>");
fclose($fl);

$fl=fopen($filecounter,"w+");
$hit=$hit+1;
fwrite($fl,$hit,strlen($hit));
fclose($fl);
?>
```



- Formulir Online

Salah satu bagian penting dari sebuah website adalah fasilitas pengisian formulir online, baik untuk keperluan pemesanan, kontak ke pemilik website atau untuk pendaftaran secara online. Pada bagian ini akan dibahas pembuatan formulir online yang data isian dikirimkan ke alamat e-mail tertentu.

Algoritma:

1. Membuat formulir dengan menggunakan tag FORM pada HTML. Dengan data yang dimasukkan adalah nama, e-mail, alamat, kota, telepon dan pesan.
2. Semua data masukkan disimpan dalam beberapa variable dan dikirimkan ke sebuah file PHP lain yang fungsinya untuk menerima variable yang dikirimkan oleh formulir.
3. Mengirimkan ke alamat e-mail tertentu yang isinya sesuai yang diisikan pada formulir.
4. Selesai

File formulir1.php:

```
<html>
<head>
<title>
Formulir Online
</title>
</head>

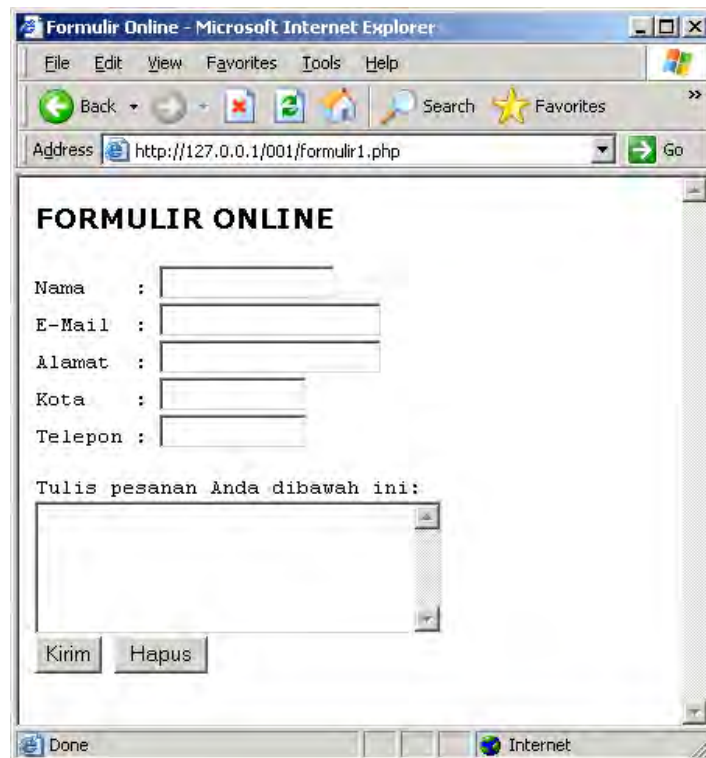
<body bgcolor="#FFFFFF">
<form action=formulir2.php method=POST>
```

```

<font face=verdana size=2 color="#000000">
<h3 align=left>
FORMULIR ONLINE
</h3>
</font>
<pre>
Nama      : <input type=text name=nama size=15>
E-Mail    : <input type=text name=email size=20>
Alamat    : <input type=text name=alamat size=20>
Kota      : <input type=text name=kota size=12>
Telepon   : <input type=text name=telepon size=12>

Tulis pesanan Anda dibawah ini:
<textarea name=pesanan rows=5 cols=30>
</textarea>
<input type=submit value="Kirim"> <input type=reset
value="Hapus">
</pre>
</form>
</body>
</html>

```



File formulir2.php:

```
<html>
<head>
<title>
Formulir Online
</title>
</head>

<body bgcolor="#FFFFFF">
<font face=verdana size=2 color="#000000">
<h3 align=left>
FORMULIR ONLINE
</h3>
</font>
<pre>
<?php
// ----- buat header email
$to="anonkuncoro@yahoo.com";
$subject="Formulir Online";
$from="From: $nama <$email>";
// ----- isi email
$content="";
$content.="Formulir Online\n";
$content.="Nama      : $nama\n";
$content.="E-Mail    : $email\n";
$content.="Alamat    : $alamat\n";
$content.="Kota      : $kota\n";
$content.="Telepon   : $telepon\n";
$content.="Pesanan Anda:\n";
$content.=" $pesanan\n";
// ----- kirim ke email
mail($to,$subject,$content,$from);

// ----- tampilkan pesan di browser
echo("Terima kasih, data yang Anda kirimkan sebagai
berikut:\n");
echo("Nama      : $nama\n");
echo("E-Mail    : $email\n");
echo("Alamat    : $alamat\n");
echo("Kota      : $kota\n");
echo("Telepon   : $telepon\n");
echo("\n");
echo("Pesanan Anda:\n");
```

```

?>
</pre>
<?php
echo (" $pesanan" );
?>
</body>
</html>

```

- Guestbook

Kali ini kita akan membahas cara pembuatan *guestbook* (buku tamu). Langkah-langkah yang kita lakukan dalam pembuatan buku tamu ini adalah:

1. Membuat table MySQL yang akan menyimpan isi buku tamu
2. Membuat form pengisian buku tamu
3. Membuat program untuk menerima masukan data dari form yang telah kita buat sebelumnya
4. Membuat program untuk menampilkan isi buku tamu.

Membuat table

Struktur table untuk menyimpan buku tamu adalah sebagai berikut:

Kolom/Field	Tipe data	Keterangan
id	int(10) unsigned auto_increment not null primary key	angka dengan panjang maksimal 10, sebagai <i>primary key</i> , tidak boleh kosong, nilai secara otomatis bertambah
posted	date	tanggal
name	varchar(80)	teks dengan panjang maksimal 80 karakter
email	varchar(128)	teks dengan panjang maksimal 128 karakter
address	varchar(128)	teks dengan panjang maksimal 128 karakter
city	varchar(80)	teks dengan panjang maksimal 80 karakter
msg	longblob	teks panjang

Adapun perintah MySQL untuk membuat table *guestbook* dengan struktur seperti diatas adalah:

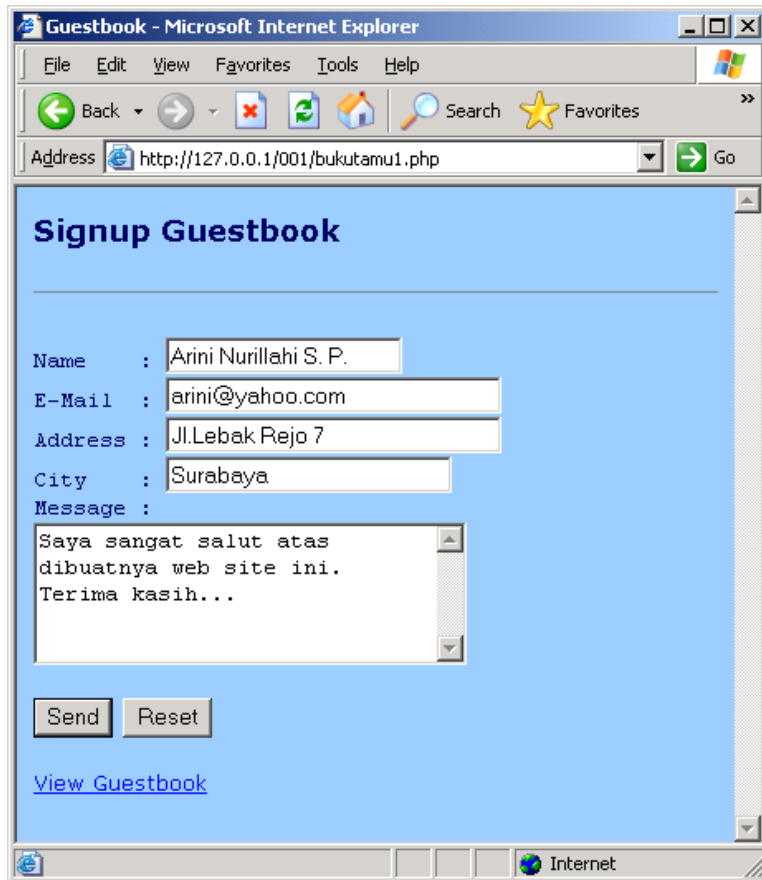
```
CREATE TABLE guestbook (
    id int(10) unsigned NOT NULL auto_increment,
    posted date NOT NULL,
    name varchar(80),
    email varchar(128),
    address varchar(128),
    city varchar(80),
    msg longblob,
    PRIMARY KEY (id)
);
```

Membuat form pengisian buku tamu

File `bukutamu1.php`:

```
<html>
<head>
<title>
Guestbook
</title>
</head>

<body bgcolor="#99CCFF">
<font color="#000066" face=verdana>
<h3 align=left>
Signup Guestbook
</h3>
<form action="bukutamu2.php" method=POST>
<hr size=1 width=100% align=left>
<pre>
Name      : <input type=text name=nama size=20>
E-Mail    : <input type=text name=email size=30>
Address   : <input type=text name=alamat size=30>
City      : <input type=text name=kota size=25>
Message   :
<textarea name=pesan rows=5 cols=30>
</textarea>
</pre>
<input type=submit value="Send"> <input type=reset
value="Reset">
</form>
</font>
<font color="#000066" face=verdana size=2>
<a href="bukutamu3.php">View Guestbook</a>
</font>
</body>
</html>
```

Program untuk menerima masukan data dari form

File `bukutamu2.php`:

```
<html>
<head>
<title>
Guestbook
</title>
</head>

<body bgcolor="#99CCFF">
<font color="#000066" face=verdana>
<h3 align=left>
Signup Guestbook
</h3>

<?php
require("utama.php");
```

```

$link=open_connection();
$tablename="guestbook";
$tgl=date("Y-m-d");
$pesan=addslashes($pesan);
$sqlstr="INSERT INTO $tablename
VALUES('',' $tgl',' $nama',' $email',' $alamat',' $kota',' $pesan
')";
if (!mysql_query ($sqlstr))
{
    echo("Invalid Query!<br>Please register again...<br>");
    exit;
}
mysql_close($link);
echo("Thank you for signing up Guestbook...<p>");

?>

</font>
<font color="#000066" face=verdana size=2>
<a href="bukutamu3.php">View Guestbook</a> <a
href="bukutamu1.php">Sign Up</a>

</font>
</body>
</html>

```



Program untuk menampilkan isi buku tamu

File bukutamu3.php:

```

<html>
<head>
<title>

```

```

Guestbook
</title>
</head>

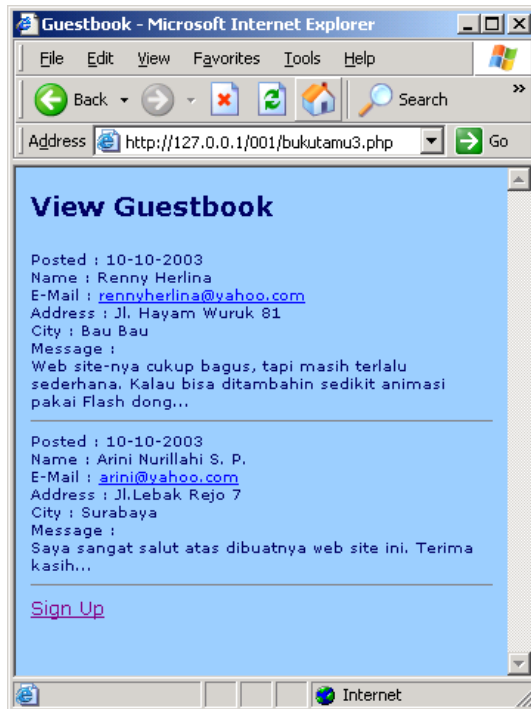
<body bgcolor="#99CCFF">
<font color="#000066" face=verdana size=1>
<h3 align=left>
View Guestbook
</h3>

<?php
require("utama.php");
$link=open_connection();
$tablename="guestbook";
$sqlstr="SELECT * FROM $tablename ORDER BY id DESC";
$result = mysql_query ($sqlstr) or die ("Invalid query");
while ($row = mysql_fetch_object ($result))
{
$name=$row->name;
$posted=substr($row->posted,8,2)."-" .substr($row->
    posted,5,2)."-" .substr($row->posted,0,4);
$email=$row->email;
$address=$row->address;
$city=$row->city;
$msg=stripslashes($row->msg);
echo("Posted : $posted<br>");
echo("Name : $name<br>");
echo("E-Mail : <a href=mailto:$email>$email</a><br>");
echo("Address : $address<br>");
echo("City : $city<br>");
echo("Message :<br>");
echo("$msg<br>");
echo("<hr size=1 width=100%>");
}
mysql_close($link);

?>

</font>
<font color="#000066" face=verdana size=2>
<a href="bukutamul.php">Sign Up</a>
</font>
</body>
</html>

```



Daftar Pustaka

- Bass, L., P. Clements, and R. Kazman. 2003. Software Architecture in Practice. 2nd Edition. Addison-Wesley.
- Cormen, T.H. 2001. Introduction to Algorithm: Second Edition. The MIT Press.
- Dobson, R. 1999. Programming Microsoft Access 2000: The Developer's Guide to Harnessing the Power of Access. Microsoft Press.
- Felleisen, M, R.B. Findler, M. Flatt, and S. Krishnamurthi. 2001. How to Design Programs; An Introduction to Computing and Programming. The MIT Press.
- Kak, A.C. 2003. Programming With Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java. John Wiley & Sons, Inc.
- Kaisler, S.H. 2005. Software Paradigm. John Wiley & Sons, Inc.
- Lafore, R. 1998. Data Structures & Algorithm in Java. Waite Group Press.
- Laurie, B and P. Laurie. 2001. Apache: The Definition Guide. 2nd Edition. O'Reilly and Associates, Inc.
- Leffingwell, D. and D. Widrig. 2003. Managing Software Requirements: A Use Case Approach. 2nd Edition. Addison-Wesley.
- Lischner, R. 2000. Delphi in a Nutshell. O'Reilly and Associates, Inc.
- Luckey, T. and J. Phillips. 2006. Software Project Management for Dummies. Wiley Publishing, Inc.
- McConnel, S. 2003. Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers. Addison-Wesley.
- Meyer, B. 2000. Object Oriented Software Construction. 2nd Edition. ISE, Inc.

- Musciano, C. and B. Kennedy. 2002. HTML and XHTML: The Definition Guide. 4th Edition. O'Reilly and Associates, Inc.
- Navarro, A. 2001. Effective Web Design. 2nd Edition. SYBEX, Inc.
- Powell, G. 2006. Beginning Database Design. Wiley Publishing, Inc.
- Riordan, R.M. 2005. Designing Effective Database Systems. Addison Wesley Professional.
- Suehring, S. 2002. MySQL Bible. Wiley Publishing, Inc.
- Taylor, D.A. 1998. Object Technology: A Manager's Guide. Addison-Wesley.
- Van Roy, P and S. Haridi. 2004. Concepts, Techniques, and Models of Computer Programming. The MIT Press.

Daftar Istilah / Glosari

Basis data (database)

Kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya

Constraint

Batasan-batasan dari masalah

Control

Aktivitas monitoring dan evaluasi terhadap feedback untuk menentukan apakah system telah bekerja dengan baik atau tidak

Database Management System (DBMS)

Perangkat Lunak yang khusus / spesifik ditujukan untuk pengelolaan basis data

Feedback

Data tentang kinerja system

Gejala

Signal atau tanda terjadinya suatu masalah

Givens

Representasi dari fakta-fakta spesifik yang harus diidentifikasi sebelum proses pemecah masalah dilakukan

Goals

Sesuatu yang ingin kita tuju atau selesaikan

Ilmu komputer

Suatu studi sistematis pada proses-proses algoritma yang menjelaskan dan mentransfor-masikan informasi

Input

Elemen-elemen yang masuk ke dalam system

Masalah (problem)

Perbedaan antara situasi aktual dan situasi yang diharapkan atau perbedaan antara kondisi sekarang dengan target atau tujuan yang diinginkan

Model

Penyederhanaan dari suatu system atau Tiruan dari suatu system dengan sedikit atau banyak penyederhanaan

Output

Perpindahan elemen-elemen yang dihasilkan dari proses perubahan ke tujuan yang diinginkan

Pemecahan masalah

Sebuah proses dimana suatu situasi dianalisa kemudian solusi-solusi dibuat bila ditemukan ada masalah dengan cara pendefinisian, pengurangan atau penghilangan, atau pencegahan masalah

Pendekatan sistem

Pendekatan yang berorientasi pada system untuk mendefinisikan masalah dan kesempatan dan mengembangkan solusinya

Perangkat lunak

Seluruh instruksi yang digunakan untuk memproses informasi

Prosedur

Instruksi yang dibutuhkan oleh pengguna dalam memproses informasi

Proses

Perubahan atau transformasi input menjadi output

Prototyping

Salah satu pendekatan dalam pengembangan perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan

Rekayasa Perangkat Lunak

suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan

Sistem

Kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai tujuan tertentu

Sistem basis data

Kumpulan elemen-elemen seperti basis data, perangkat lunak, perangkat keras, dan manusia yang saling berinteraksi untuk mencapai tujuan yaitu pengorganisasian data.

Software

Lihat Perangkat Lunak

Software Engineering

Lihat Rekayasa Perangkat Lunak

Solusi

Bagian akhir atau output dari proses pemecahan masalah.

Strategi pemecahan masalah

Metode atau pendekatan yang digunakan seseorang ketika menghadapi masalah

Unknowns

Sesuatu yang harus kita dapatkan setelah proses pemecahan masalah dilakukan

Lampiran 1.

Daftar Alamat Situs

Berikut ini daftar alamat situs-situs internet yang penting dan digunakan sebagai rujukan dalam buku ini.

Alamat	Keterangan
http://www.apache.org	Situs resmi web server Apache. Situs ini menyediakan kode sumber Apache dan file-file binary Apache yang siap diinstall di berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi Apache yang lengkap.
http://www.borland.com	Situs resmi Borland. Borland merupakan perusahaan perangkat lunak yang memproduksi Borland Delphi, Borland JBuilder, Turbo Pascal, Turbo Delphi, Borland C++ dan lain-lain.
http://www.debian.org	Situs resmi distribusi linux Debian.
http://www.google.com	Situs resmi search engine Google.
http://www.ilmukomputer.com	Situs berbahasa Indonesia yang menyediakan dokumen-dokumen untuk belajar berbagai sub bidang dalam ilmu computer.
http://www.javasoft.com	Situs resmi yang diluncurkan Sun Microsystem dan berisi dokumentasi dan informasi online tentang bahasa pemrograman Java.
http://www.kambing.vlsm.org	Situs dengan server local di Indonesia. Situs ini menyediakan file-file iso dari berbagai jenis distribusi linux dan dapat didownload secara bebas. Selain itu situs ini juga sebagai mirror dari berbagai distribusi linux dan aplikasi yang berjalan di linux.

http://www.linuxdoc.org	Situs yang berisi dokumentasi bebas tentang linux. Sumber informasi online yang sangat bagus untuk mempelajari linux
http://www.microsoft.com	Situs resmi Microsoft. Microsoft merupakan perusahaan perangkat lunak yang memproduksi system operasi keluarga Windows, IDE Microsoft Visual Studio, Microsoft Office, Microsoft SQL Server, dan lain-lain.
http://www.mysql.com	Situs resmi MySQL Database Software. Situs ini menyediakan file-file instalasi MySQL untuk berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi MySQL yang lengkap.
http://www.php.net	Situs resmi bahasa pemrograman dan interpreter PHP. Situs ini menyediakan kode sumber dan file-file instalasi PHP untuk berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi PHP yang lengkap.
http://www.w3.org	Situs resmi The World Wide Web Consortium (W3C). W3C adalah konsorsium yang menetapkan standar dalam teknologi internet, terutama tentang HTML, XML, CSS, XHTML dan teknologi lain. Dokumentasi tentang teknologi tersebut dapat dijumpai di situs ini.

Lampiran 2.

Fungsi Built-in Pada Visual Basic

IsNumeric(ekspresi)

Mengembalikan nilai boolean yang menunjukkan suatu ekspresi dapat dievaluasi sebagai numeric

IsEmpty(ekspresi)

Mengembalikan nilai boolean yang menunjukkan suatu variabel telah diinitaliasi, biasanya fungsi ini digunakan untuk memeriksa variabel yang dideklarasikan sebagai Variant.

Contoh :

```
Dim A as Variant
If IsEmpty(A) Then
    MsgBox "Variabel A Empty"           'Message
    akan muncul karena A belum terinisialisai
End If
A = 1
If IsEmpty(A) Then
    MsgBox "Variabel A masih Empty"     'Tidak
    akan muncul lagi karena A telah terinisialisasi
    sebagai Integer
End If
Dim B as Integer
If IsEmpty(B) Then
    MsgBox "Variabel B Empty"           'Tidak
    akan muncul karena B adalah Integer
End If
```

IsNull(ekspresi)

Mengembalikan nilai boolean yang menunjukkan suatu ekspresi mengandung data yang tidak valid, biasanya digunakan untuk memeriksa isi field recordset.

IsArray(varname)

Mengembalikan suatu nilai boolean yang menunjukkan suatu variabel adalah suatu array.

IsDate(ekspresi)

Mengembalikan suatu nilai boolean yang menunjukkan suatu ekspresi dapat dikonversi ke date.

IsError(ekspresi)

Mengembalikan suatu nilai boolean yang menunjukkan suatu ekspresi adalah nilai error

IsObject(ekspresi)

Mengembalikan suatu nilai boolean yang menunjukkan suatu ekspresi mengacu pada suatu OLE Automation object.

IsMissing(argname)

Mengembalikan suatu nilai boolean yang menunjukkan suatu argumen optional pada procedure ada dilewatkan atau tidak

CBool(ekspresi)

Konversi suatu ekspresi ke Boolean

Contoh :

```
A = 5: B = 5           ' Inisialisasi variabel
Check = CBool(A = B)   ' Check berisi nilai
True

A = 0
Check = CBool(A)       ' Check berisi False.
```

CByte(ekspresi)

Konversi ekspresi ke Byte

Contoh :

```
MyDouble = 125.5678    ' MyDouble
adalah suatu nilai Double

MyByte = CByte(MyDouble) ' MyByte
berisi 126.
```

CCur(ekspresi)

Konversi suatu ekspresi ke Currency

Contoh :

```
MyDouble = 543.214588           ' MyDouble
adalah suatu nilai Double.
MyCurr = CCur(MyDouble * 2)     ' Konversi
hasil MyDouble * 2 (1086.429176) menjadi suatu
Currency (1086.4292).
```

CDate(date)

Konversi suatu ekspresi ke date

Contoh :

```
MyDate = "February 12, 1969"    ' Definisi
waktu dalam string
MyShortDate = CDate(MyDate)     ' Konversi ke
type data Date
MyTime = "4:35:47 PM"           ' Definisi
waktu dalam string
MyShortTime = CDate(MyTime)     ' Konversi ke
type data Date
```

CDBl(ekspresi)

Konversi suatu ekspresi ke Double

Contoh :

```
MyCurr = CCur(234.456784)      ' MyCurr
adalah nilai Currency.
MyDouble = CDBl(MyCurr * 8.2 * 0.01) ' Konversi
hasil ke suatu Double.
```

CInt(ekspresi)

Konversi suatu ekspresi ke Integer

Contoh :

```
MyDouble = 2345.5678           ' MyDouble
adalah nilai Double.
MyInt = CInt(MyDouble)         ' MyInt
berisi nilai 2346.
```

CLng(ekspresi)

Konversi suatu ekspresi ke Long

Contoh :

```
MyVal1 = 25427.45: MyVal2 = 25427.55 ' MyVal1,  
MyVal2 adalah Double  
  
MyLong1 = CLng(MyVal1) ' MyLong1  
berisi nilai 25427.  
  
MyLong2 = CLng(MyVal2) ' MyLong2  
berisi nilai 25428.
```

CSng(ekspresi)

Konversi suatu ekspresi ke single

Contoh :

```
' MyDouble1, MyDouble2 adalah Double.  
MyDouble1 = 75.3421115: MyDouble2 = 75.3421555  
  
MySingle1 = CSng(MyDouble1) ' MySingle1  
berisi nilai 75.34211.  
  
MySingle2 = CSng(MyDouble2) ' MySingle2  
berisi nilai 75.34216.
```

CStr(ekspresi)

Konversi suatu ekspresi ke string

Contoh :

```
MyDouble = 437.324 ' MyDouble  
adalah Double  
  
MyString = CStr(MyDouble) ' MyString  
berisi nilai "437.324".
```

CVar(ekspresi)

Konversi suatu ekspresi ke Variant

Contoh :

```
MyInt = 4534 ' MyInt  
adalah suatu Integer.
```

```
MyVar = CVar(MyInt & "000")           ' MyVar
berisi nilai string 4534000.
```

Asc(string)

Mengembalikan kode character dari huruf pertama di suatu string.

Contoh :

```
MyNumber = Asc("A")                     '
Mengembalikan 65.

MyNumber = Asc("a")                     '
Mengembalikan 97.

MyNumber = Asc("Apple")                 '
Mengembalikan 65.
```

Chr(charcode)

Mengembalikan karakter dari suatu kode karakter

Contoh :

```
MyChar = Chr(65)                        '
Mengembalikan A.

MyChar = Chr(97)                        '
Mengembalikan a.

MyChar = Chr(62)                        '
Mengembalikan >.

MyChar = Chr(37)                        '
Mengembalikan %.
```

Format(ekspresi[, format[, hariPertamaDariMinggu[, mingguPertamaDariTahun]]])

Memformat suatu ekspresi berdasarkan ekspresi format

Contoh :

```
MyTime = #17:04:23#
MyDate = #January 27, 1993#

' Mengembalikan waktu sistem dalam format long
time
MyStr = Format(Time, "Long Time")
```



```
' Mengembalikan tanggal sistem dalam format long
date
```

```
MyStr = Format(Date, "Long Date")
```

```
MyStr = Format(MyTime, "h:m:s")           '
Mengembalikan "17:4:23".
```

```
MyStr = Format(MyTime, "hh:mm:ss AMPM")   '
Mengembalikan "05:04:23 PM".
```

```
MyStr = Format(MyDate, "dddd, mmm d yyyy") '
Mengembalikan "Wednesday, Jan 27 1993".
```

```
'Jika format tidak tersedia, suatu string
dikembalikan.
```

```
MyStr = Format(23) ' Mengembalikan "23".
```

```
' Format User-defined
```

```
MyStr = Format(5459.4, "##,##0.00")       '
Mengembalikan "5,459.40".
```

```
MyStr = Format(334.9, "###0.00")          '
Mengembalikan "334.90".
```

```
MyStr = Format(5, "0.00%")                '
Mengembalikan "500.00%".
```

```
MyStr = Format("HELLO", "<")              '
Mengembalikan "hello".
```

```
MyStr = Format("This is it", ">")         '
Mengembalikan "THIS IS IT".
```

Hex(number) dan Oct(number)

Mengembalikan string yang mewakili Octal atau Hexa dari suatu bilangan

Contoh :

```
MyHex = Hex(5)                             '
Mengembalikan 5.
```

```
MyHex = Hex(10)                            '
Mengembalikan A.
```

```
MyHex = Hex(459)                           '
Mengembalikan 1CB.
```

```

MyOct = Oct(4)           '
Mengembalikan 4.

MyOct = Oct(8)           '
Mengembalikan 10.

MyOct = Oct(459)         '
Mengembalikan 713.

```

Str(number)

Mengembalikan string yang mewakili suatu angka.

Contoh :

```

MyString = Str(459)       '
Mengembalikan " 459".

MyString = Str(-459.65)   '
Mengembalikan "-459.65".

MyString = Str(459.001)   '
Mengembalikan " 459.001".

```

Val(string)

Mengembalikan angka yang terkandung dalam suatu string.

Contoh :

```

MyVal =Val(" 1615 198th Street N.E.") '
Mengembalikan 1615198

MyVal =Val("&HFFFF") '
Mengembalikan -1

MyVal =Val("2457") '
Mengembalikan 2457.

MyVal =Val(" 2 45 7") '
Mengembalikan 2457.

MyVal =Val("24 and 57") '
Mengembalikan 24.

```

Fungsi Bantu Waktu

Now

Mengembalikan suatu Variant (Date) yang menunjukkan tanggal dan waktu berdasarkan sistem komputer.

Time

Mengembalikan waktu sistem sekarang

Timer

Mengembalikan suatu bilangan yang menunjukkan jumlah detik sejak tengah malam

Date

Mengembalikan tanggal sistem sekarang

Time = Time dan Date = Date

Mengatur waktu atau tanggal sistem

Untuk sistem yang menjalankan Microsoft Windows 95, tanggal yang dibutuhkan harus berupa tanggal dari 1 Jan 1998 sampai 31 Des 2099. Untuk sistem yang menjalankan Microsoft Windows NT, tanggal yang dibutuhkan harus berupa tanggal dari 1 Jan 1980 sampai 31 Desember 2079.

Hour(time), Minute(time) dan Second(time)

Mengembalikan suatu Variant (Integer) berupa bilangan 0 s/d 23 untuk jam, 0 s/d 59 untuk menit, dan 0 s/d 59 untuk detik.

Day(date), Month(date), dan Year(date)

Mengembalikan suatu Variant (Integer) berupa bilangan 1 s/d 31 untuk bulan, 1 s/d 12 untuk bulan, dan tahun.

Biodata Penulis



Aunur Rofiq Mulyarto, penulis buku ini lahir di Blitar pada tanggal 25 Nopember 1970. Pada usia 4 tahun, keluarganya pindah ke Kota Malang, sehingga masa kecil dihabiskan di Kota Malang. Menyelesaikan sekolah dasar pada SDN Dinoyo 2 Malang dan sekolah menengah pertama pada MTs Negeri 1 Malang. Penulis kemudian melanjutkan pada SMA Negeri 2 di kota yang sama.

Setamat dari SMA pada tahun 1989, penulis melanjutkan ke jenjang perguruan tinggi di Institut Pertanian Bogor, melalui jalur PMDK (Penelusuran Minat dan Bakat). Jurusan yang diambil adalah Teknologi Industri Pertanian. Setelah berkuliah dengan bangku kuliah selama kurang lebih 5 tahun, pada tahun 1994 penulis berhasil memperoleh gelar Sarjana Teknologi Pertanian.

Setelah lulus, penulis sempat bekerja pada sebuah perusahaan sepatu di Sidoarjo, sebelum diterima menjadi staf pengajar pada Jurusan Teknologi Industri Pertanian, Fakultas Teknologi Pertanian, Universitas Brawijaya Malang pada tahun 1999. Pada tahun 2001 penulis memperoleh kesempatan tugas belajar pada Program Internasional Master of Science in Information Technology for Natural Resources Management yang diselenggarakan oleh IPB dan SEAMEO – BIOTROP. Gelar Master diperoleh pada tahun 2003 dengan predikat cumlaude.

Saat ini selain aktif sebagai staf pengajar Jurusan Teknologi Industri Pertanian, Fakultas Teknologi Pertanian, Universitas Brawijaya Malang, penulis juga terlibat dalam beberapa penelitian, pengabdian dan proyek-proyek pengembangan sistem informasi. Penulis cukup aktif menulis beberapa karya ilmiah, antara lain buku teks Teknik Pemrograman, buku Teks Sistem dan Teknologi Informasi, dan sejumlah tulisan jurnal ilmiah. Di sela-sela kesibukannya, penulis masih sempat menyalurkan hobi lamanya bermain music bersama rekan-rekan sejawatnya.