

Bab 7

Tahap Desain

Bagaimana cara kerjanya

7.1 INTRODUCTION

Untuk para pembaca yang bekerja dibidang tehnik dan bagi yang kurang tertarik membicarakan tentang semua kegiatan perencanaan sebelumnya. Kita sampai pada pada satu hal yang paling menarik dari kegiatan tehnik-perencanaan sistem. Kita dapat turut menyelesaikan masalah-masalah sebenarnya dengan penyelesaian yang nyata.

Kegiatan utama tahap desain adalah membuat tingkat tertinggi dan tingkat menengah dari desain sistem dan mendokumentasikannya dalam perincian desain. Kegiatan kedua dalam tahapan ini akan mulai menerima rencana pengujian atau *Acceptance Test Plan* (ATP). ATP adalah sebuah dokumen laporan pengujian yang akan digunakan untuk mendemonstrasikan semua fungsi sistem kepada pemakai pada tahap penerimaan.

Hal penting utama adalah saat perincian desain berjalan (ditinjau) dan dinyatakan bebas dari kesalahan. Hal penting lainnya adalah pemakai mengakhiri ATP meskipun hal ini mungkin tidak akan terjadi sampai selanjutnya.

Catatan tingkatan dari usaha terdapat dalam gambar 1.1. Usaha manajer berkurang, tapi total usaha dan biaya untuk itu meningkat sejak beberapa perancang dan personalia walk-through yang terlibat.

Mendesain sebuah sistem perangkat lunak terdiri dari dua langkah utama: Pertama, bagilah sistem menjadi komponen-komponen fungsional, dan kedua, hubungkan antar komponen-komponen ini. Banyak macam metode desain diterbitkan. Beberapa yang bagus diantaranya adalah Warnier (keterangan 6), Orr (keterangan 7), Nassi-Shneiderman (keterangan 9), dan Lately Object Oriented Design menjadi terkenal (keterangan 8). Catatan dari keterangan 10 adalah sebuah judul karangan "A Survey of Software design Techniques." Kita tidak akan merinci beberapa metode desain khusus dalam bagian ini kecuali untuk hal sederhana, hirarki fungsional lepas. Hal ini tidak masalah dengan metode apa yang anda gunakan, selama anda menggunakan suatu metode standar untuk setiap orang. Kita mampu , walau bagaimanapun, fokuskan pada bagaimana melakukan desain file karena cara yang anda lakukan untuk membuat masukan dan keluaran akan merusak tampilan sistem anda dengan drastis.

Maka dari itu seseorang yang bukan orang teknis dapat menemukan pada bagian ini sedikitnya juga terlibat. Jika anda memfokuskannya pada bagian manajemen dan mempunyai para perancang yang bagus yang bekerja pada anda maka lewati bagian ini. Namun bagaimanapun juga anda pasti ingin mengetahui bagaimana mereka bekerja, atau seandainya anda seorang manajer teknis yang ingin tahu yang ingin mengetahui tentang semua hal yang

berkenaan pada proyek (bukan untuk proyek kecil ataupun menengah). Pada beberapa kasus contohnya War Stoies (cerita ini menyenangkan) dan kesimpulannya.

Kisah Tentang Perang Desain (Design War Story)

Cerita ini melibatkan dua orang pemuda yang menjadi sahabat karib ketika mengambil kuliah Ilmu Komputer pada sebuah Universitas terkemuka di Ontario, Kanada. Setelah lulus yang satu menjadi seorang Manajer Pengolahan Data pada sebuah perusahaan penerbitan terkenal, dan yang satu menjadi seorang perancang sistem pada sebuah Famous Minicomputer Manufacturing Company (FMMC). Beberapa tahun yang lalu perusahaan penerbitan memutuskan untuk memasang sebuah komputerisasi baru untuk sistem pengolahan. Manajer DP (Data Processing) tidak mampu menangani beban pekerjaan itu sendiri, maka ia menyewa temannya dari (FMMC) untuk membantu mengerjakan desain dan program.

Saya tidak tahu pasti rincian cerita ini, tapi saya menyimpulkan selanjutnya: Setelah mereka mengerjakan desain yang paling tinggi bersama, mereka membagi modul utama, yang masing-masing mengerjakan desain tingkat menengah dari modul-modul yang spesifik. Perancang dari FMMC mendesain modul yang pertama. Untuk menunjukkan kehebatan desainnya pada manajer DP temannya, Ia membuat rancangan itu dengan sangat "bagus" untuk menghemat sedikit bytes dari penyimpanan atau sedikit nanodetik dari waktu CPU. Ia bahkan menambahkan sedikit keistimewaan tersendiri: sesuatu yang tidak seorangpun meminta atau mengerti atau pernah memakai.

Manajer DP menanggapi dengan mendesain modul yang kedua. Ketika dia melihat rancangan hebat milik temannya, ia mungkin berpikir "saya dapat membuat yang lebih bagus dari itu !" Lalu ia membuat modulnya sedikit lebih bagus dari modul temannya, dan menambah sedikit 'suara bel-bel dan melodi-melodi' pada modulnya. Ia mengerjakan lebih lama dari jadwal yang ditentukan dan modul menjadi lebih besar dari rencana. Perancang , tertantang untuk membuat modul yang ketiga yang lebih 'sempurna' dengan menambah lebih banyak lagi suara bel-bel dan melodi-melodi sampai modul itu menjadi dua kali lebih besar dari seharusnya. Dan perlombaan terus berlanjut sampai mendesain sebagus mungkin tahap-tahap pemrograman. Sistem dibuat untuk 16 orang pemakai. Ketika sistem akhirnya berjalan (50% terlambat) sistem bekerja baik sampai 4 pemakai. Ketika enam orang pemakai bekerja sistem menjadi sangat lambat, dengan delapan pemakai, sistem berantakan- program sangat besar untuk dapat ditangani oleh sistem.

Komentar: Orang teknis dapat terbawa dengan tantangan teknis. Selalu banyak jalan yang bagus untuk menyelesaikan masalah, tapi tantangan terutama harus menghasilkan tujuan dari waktu dan biaya.

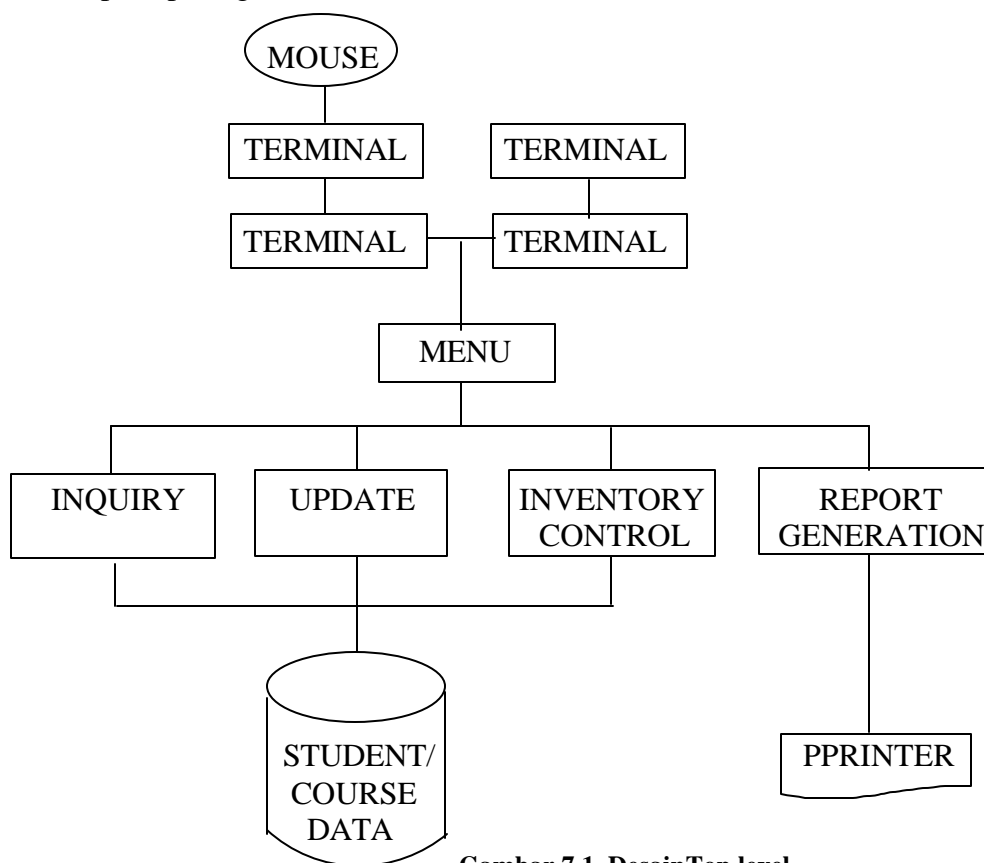
Epilog: FMMC memberi CPU yang lebih besar tanpa biaya.

7.2 DESAIN YANG TERSTRUKTUR

Tujuan utama dari desain yang terstruktur adalah untuk memecah sistem menjadi lebih kecil, dapat diatur, dapat dibentuk bagian-bagiannya. Beberapa metode hebat mempunyai dokumentasi untuk mengerjakannya (keterangan 10, 11). Pendekatan yang akan kita peroleh lebih mendasar dari metode-metode itu: kita akan mudah memecah sistem menjadi lebih kecil dan komponen fungsional yang lebih kecil sampai cukup sistem cukup terpecah untuk para pembuat program membuat kode.

Desain Top Down

Desain Top Down dimulai dengan Top Level Design (TLD) contohnya satu desain dibangun untuk sistem ABC pada tahap analisis (bagian 6.3), dibangun seperti pada gambar 7.1.



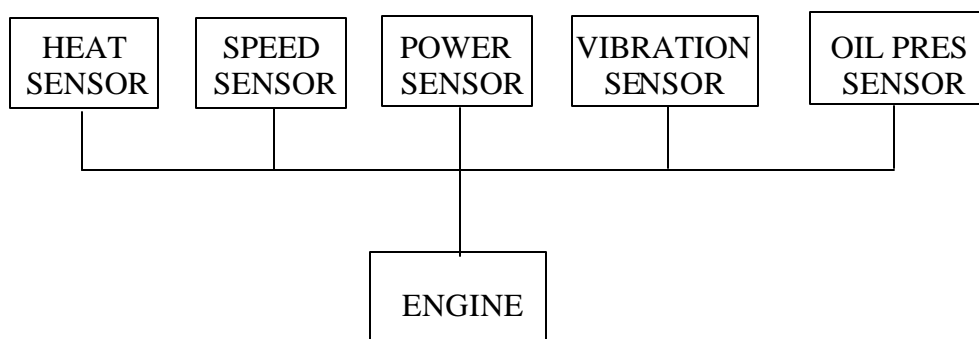
Gambar 7.1 Desain Top level

Sebagian komponen utama, atau kotak dalam TLD yang kemudian dipecah menjadi sub bagian-sub bagian, dimulai dengan tingkatan yang paling tinggi, menurun ke tingkat selanjutnya dan seterusnya. Dalam kasus ini kita akan mulai dengan MENU dan merancang ini sebelum turun ke INQUIRY,

UPDATE dan REPORT GENERATION yang akan diikuti dengan tingkat selanjutnya, jika ada.

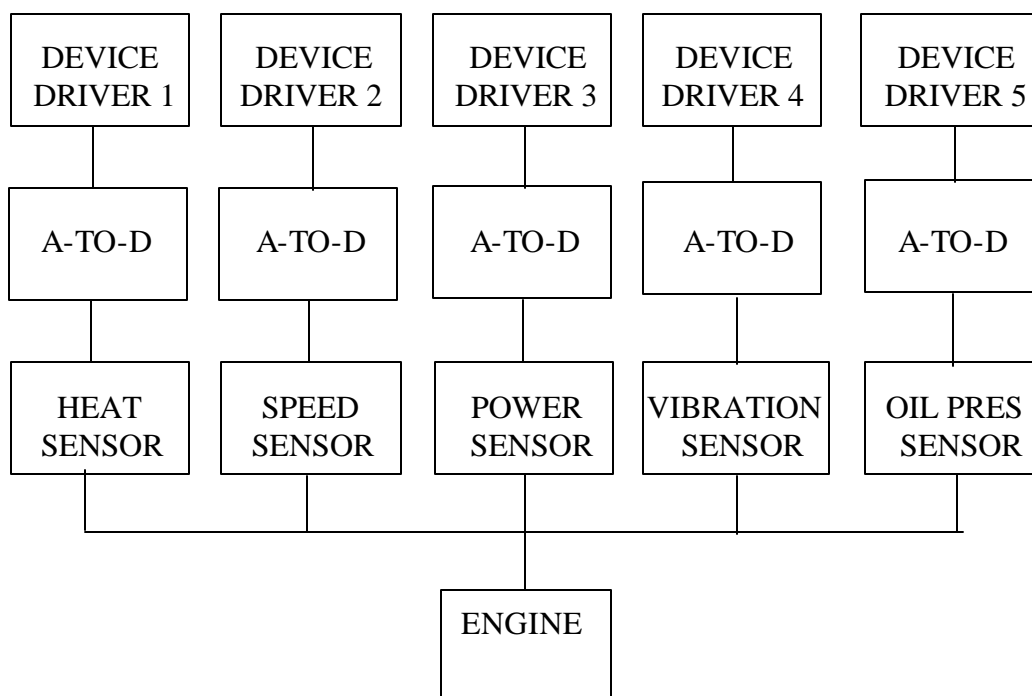
Desain Bottom Up

Dalam kasus tertentu mungkin akan lebih mudah bila dilakukan pendekatan desain dari tingkat yang terendah ke tingkat yang lebih tinggi. Hal ini sering ditemui pada kasus sistem kontrol proses dimana perangkat keras kontrol pada tingkat bawah menentukan bagaimana sistem tersebut disatukan (integrasi sistem). Sebagai contoh, kita akan mendesain sebuah sistem pengujian mesin kendaraan. Kita harus mulai dengan menentukan perangkat keras dasar atau komponen dasar yang terlibat--sensor mesin (gambar 7.2A).

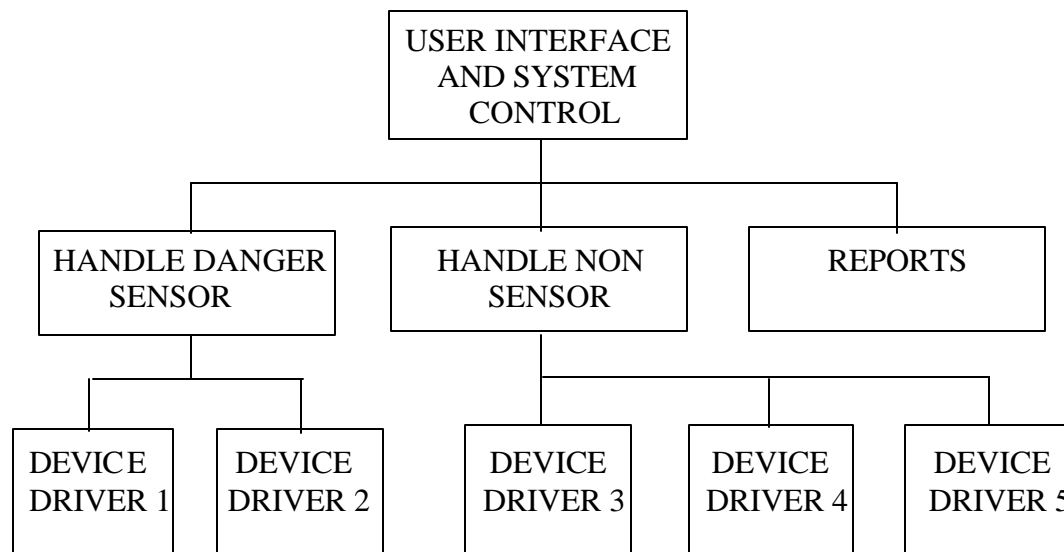


Gambar 7.2A Desain Bottom Up

Sensor umumnya dipasang pada alat digital atau analog, yang terpasang pada modul perangkat lunak pengendali alat yang unik (Gambar 7.2B)



Gambar 7.2B Desain Botton Up (lanjutan)



Gambar 7.2C Desain Bottom Up

Perangkat lunak yang digunakan untuk mengontrol pengendali alat kemudian didesain di atas pengendali-pengendali tersebut (gambar 7.2C).

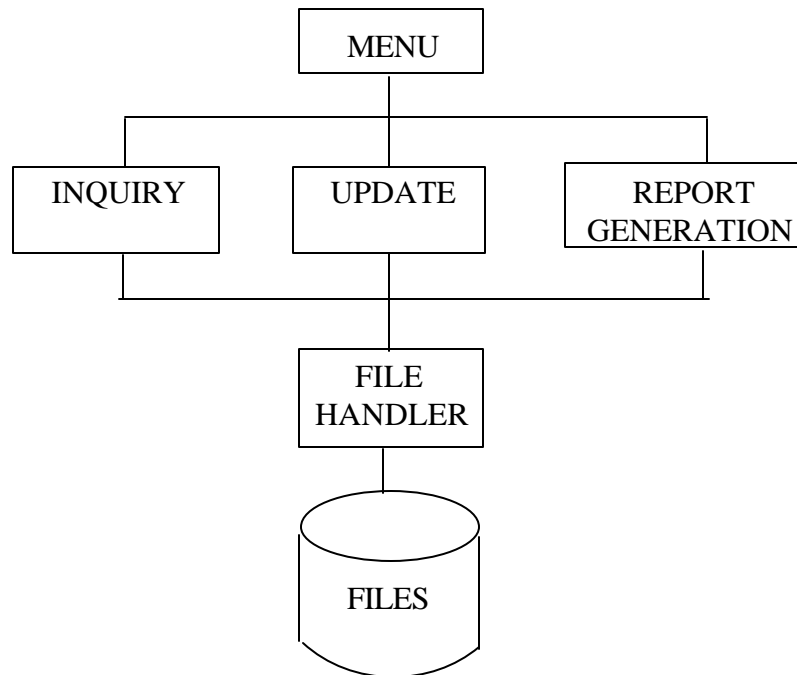
Demikianlah bagaimana sistem perangkat lunak tersebut didesain dari bawah ke atas. Desain Bottom Up juga sangat cocok digunakan pada kasus dimana komponen perangkat lunak yang ada digabungkan dan disatukan dengan modul baru untuk membangun sebuah sistem.

7.3 PERTUKARAN DESAIN TINGKAT ATAS

Umumnya banyak desain tingkat atas yang dapat mencapai atau memperoleh hasil yang sama dalam sebuah sistem perangkat lunak. Contohnya, desain tingkat atas (top level) pada gambar 7.1 hanya salah satu cara untuk memecahkan sistem ABC ke dalam komponen-komponen utama. Metode yang lain mungkin dengan menggunakan Sistem Manajemen Berbasis Data (Contohnya DATARIEVE, SQL, atau Fourth Generation System) untuk menggantikan porsi INQUIRY dan UPDATE, atau sebuah Forms Management System (FMS) untuk melakukan MENU, mungkin sebuah Report Generation System (RPG) untuk pelaporan, atau kombinasi dari komponen yang telah disebutkan di atas. Ini adalah ciri keputusan "membangun atau membeli", dan ada keuntungan dan kerugian pada setiap kombinasi dari item yang dibangun maupun yang dibeli. Semakin banyak paket yang anda beli, semakin berkurang pemrograman yang harus anda lakukan; tetapi paket

tersebut mahal harganya, dan umumnya kurang efisien dari program tertulis biasa yang sama.

Desain tingkat atas yang lain ada juga yang cocok. Salah satu masukan mungkin adalah menghilangkan porsi INQUIRY, UPDATE, dan REPORT GENERATION dan menggunakan rutin FILE HANDLER yang umum untuk melakukan semua kegiatan akses file. TLD (Top Level Desain) akses tersebut seperti gambar 7.3.



Gambar 7.3 Desain Top Level (lainnya)

Disini ada lima program yang harus dibuat dan sedikit digradasi kinerja akan terlihat oleh karena pemanggilan yang sering pada FILE HANDLER, tetapi sistem akan menjadi lebih kecil. Setiap pilihan TLD memiliki keuntungan dan kerugian dan melibatkan pertukaran dan kompromi.

Perioritas Desain

Pilihan TLD anda akan mempengaruhi hal-hal berikut :

- Biaya Sistem
- Waktu yang diperlukan untuk membangun sistem
- Sifat mudah dipakai
- Penampilan (kinerja)
- Ukuran Sistem
- Kehandalan
- Kemampuan modifikasi

Item-item ini harus menjadi prioritas, bersama dengan user pada waktu perencanaan sistem saat Pendefinisian dan analisis. Ini akan membuat pemilihan TLD jauh lebih mudah.

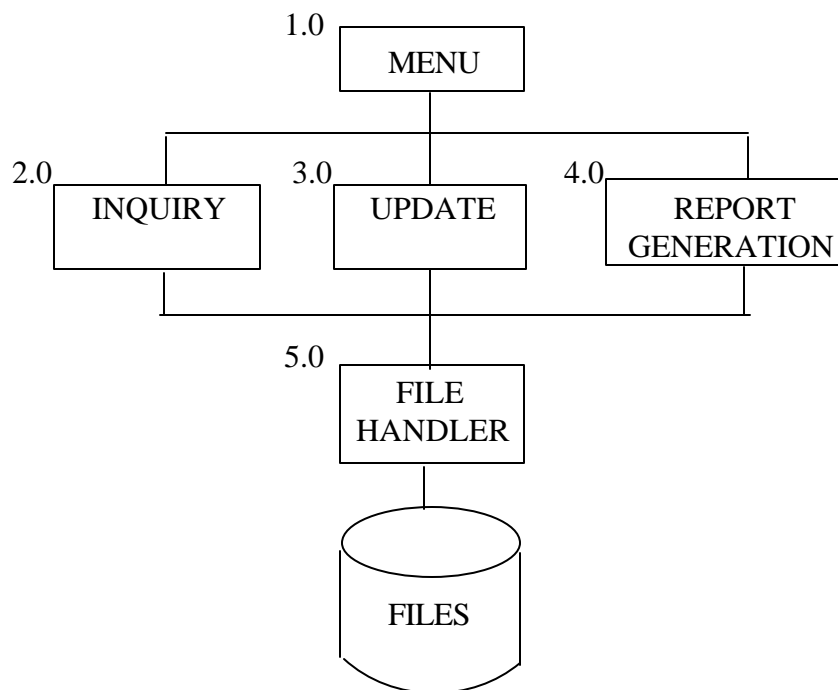
7.4 DESAIN WALK-THROUGHS

Ketika mengambil keputusan diantara beberapa pendekatan teknis terhadap suatu masalah, akan membuat pengambil keputusan lebih mudah dengan meminta pendapat orang lain. Adakan pertemuan dengan beberapa ahli untuk melakukan *desain walk-throughs* memberikan semua salinan FS dan semua TLD dengan daftar keuntungan dan kerugian dari setiap TLD kepada setiap peserta rapat. Beritahukan kepada mereka tujuan dari pertemuan adalah memilih TLD terbaik. Ini akan terlaksana dengan "walkin through", tahap demi tahap, setaip desain yagn diusulkan, pastikan bahwa daftar pertukaran adalah benar. Setiap orang harus dapat didorong untuk memberikan usulan desain alternatif, sama dengan pertukaran tambahan bahwa mungkin terlewatkan oleh penulis. Biarkan setiap orang yagn berada pada tim walk-through mengetahui bahwa mereka sebagai tim bertanggung jawab untuk menentukan desain terbaik.

TLD walk-throughs (dan kemudian ketika kita membahas desain tingkat bawah, dokumentasi dan program walk-throughs) dan bernilai tinggi bila aturan yang ada dipatuhi, antara lain : "Tinggalkan EGO anda diluar" (ini adalah tanda pada pintu studio ketika lagu "We are the world" dan "Tears are not enough" direkam) tujuannya adalah untuk tidak menyalahkan perancang, dan juga bukan kesempatan bagi peserta untuk membuktikan bahwa dia dapat melakukannya lebih baik. Para perancang juga harus menyadari bahwa setiap kritik sifatnya adalah membangun--dia tidak boleh bersifat difensive, objektivitasnya adalah utnuk menemukan masalah, menyarankan alternatif dan membuat pilihan yang mungkin terbaik. Beberapa orang menyarankan bahwa manajer jangan diundang pada walk-throughs. Manajer dianggap dapat menghalangi alur ide dan diskusi.

7.5 DESAIN TINGKAT MENENGAH

Setelah TLD terpilih, anda harus memisahkan setiap fungsi atau komponen utama menjadi sub fungsi atau komponen. Kita akan lihat bagaimana hal tersebut dilakukan untuk menggabungkan sistem perusahaan Basketweaving. Diawali dengan menomori setiap komponen utama pada TLD.



Gambar 7.3 Sistem penomoran untuk TLD

Urutan dalam desain top down ini ditentukan bahwa seluruh proses harus dimulai dengan box menu. Mari kita asumsikan bahwa komponen ini dipanggil ketika seluruh sistem dimulai dan ditampilkan dalam bentuk main menu kepada bagian registrasi.

```

MAIN MENU
1. INQUIRE ON A COURSE/STUDENT
2. UPDATE COURSE/STUDENT DATA
3. REGISTER STUDENT
4. WAREHOUSE
5. REPORT
6. QUIT

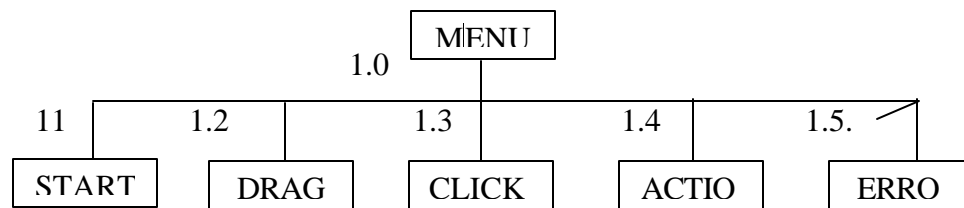
USE UP/DOWN ARROW TO HILITE YOUR
CHOICE THEN
PRESS RETURN,
OR MOVE MOUSE TO HILITE YOUR CHOICE,
THEN PUSH
BUTTON ON MOUSE
  
```

Kemudian program menunggu hingga pemakai menggerakkan mouse. Subfungsi utama dalam komponen utama dapat berupa:

1. Jalankan sistem dan tampilkan main menu.

2. Kendalikan gerakan mouse.
3. Kendalikan tombol pada mouse.
4. Pergi ke menu INQUIRE, UPDATE, WAREHOUSE atau REPORT ketika akan memilih.
5. Mengendalikan masalah saat on-line dengan menggunakan help message untuk seluruh sistem.
6. Sistem akan mati jika memilih QUIT.

Struktur diagram level selanjutnya untuk komponen menu dapat digambarkan sbb:



Gbr. 7.5. Level kedua dari pecahan diagram.

Level ketiga dari diagram dapat dilihat pada Appendix A - Spesifikasi disain. Anda boleh beranggapan bahwa kerusakan ini akan berlanjut. Ingat, anda dapat pindah ke suatu level dimana programmer memulai pembuatan modul serta pemrogramannya. Level yang terendah dari suatu menu menggambarkan suatu modul. Modul adalah bagian yang paling mudah diuji dan diartikan. Lihat bagian 7.7 untuk menyusun modul yang baik.

Pada struktur diagram (Appendix A) kita dapat melihat alur kontrol: garis tebal yang menghubungkan antar modul. Kita juga dapat melihat aliran data : panah bergaris putus-putus yang menunjukkan parameter yang dilewati. Arah panah menunjukkan aliran pergerakan data.

Aturan Penamaan

Modul diberi nama untuk mengindikasi sistem, fungsi, atau subfungsi seperlunya (lihat Appendix A, Spesifikasi disain, Bagian 6 untuk penamaan modul sistem ABC). Bahasa yang diperbolehkan untuk menggunakan lebih banyak karakter, dibuat suatu sistem penamaan yang lebih rinci yang secara jelas mengindikasikan fungsi-fungsi baik modul maupun karakter. Janganlah mencoba untuk meningkatkan

Aturan Penomoran

Urutan dari setiap kotak disusun sebagai berikut: Pada setiap level yang lebih rendah sisipkan suatu angka dan titik untuk angka pada kotak di atasnya. Angka dapat diurutkan dari kiri ke kanan. Angka menunjukkan jalur dari struktur pohon. Sesuai dengan level dimana kotak itu dijumpai.

7.6 KAMUS DESAIN

Modul Kamus

Perkembangan selanjutnya dari disain yang anda buat, buatlah 3 buah kamus di bawah ini:

Kamus 1. Urutkan angka sesuai dengan urutan komponen, beri nama yang tetap dan deskripsikan secara singkat untuk setiap modul. Contoh:

0.0	A0000000	Amalgamated Basketweaving System
1.0	AM000000	Menu system
1.1	AMST0000	Startup, disp first menu, shutdown etc.

Kamus 2. Urutkan secara alphabet berdasarkan nama komponen, berikan angka yang tetap dan deskripsi singkat untuk setiap modul. Contoh:

A0000000	0.1	Amalgamated Basketweaving System
AM000000	1.0	Menu system
AMST0000	1.1	Startup, disp first menu, shutdown

Ini dapat dibuat dengan mudah dari kamus 1 dengan menggunakan program yang singkat.

Kamus 3. Urutkan deskripsi secara alphabet, beri nomor komponen dan nama rutin. Contoh :

Amalgamatd Basketweaving System	0.0	A0000000
Menu system	1.0	AM000000
Startup, disp first menu, shutdown	1.1	AMST0000

Ini juga dapat dibuat dari kuamus 1 menggunakan program singkat. Anda dapat menggunakan kamus ini selama mendesain, membuat program atau pengujian dan perawatan rutin setiap kali anda menginginkan suatu modul, anda dapat menggunakan kamus ini.

Kamus Data Umum / The Common Data Dictionary (CDD)

Urutkan dalam urutan alphabet semua parameter yang ditunjukkan oleh panah aliran data. Untuk setiap item, tampilkan tipe, panjang, aturan dan modul. CDD ini kemudian akan mengandung semua parameter yang didefinisikan pada level yang terendah dari pemrograman dan desain sebagaimana field didefinisikan dalam suatu file. CDD memastikan bahwa parameter akan konsisten berlaku dalam seluruh sistem. Beberapa sistem operasi seperti VAX VMS menyediakan CDD.

7.7 MODUL YANG TERSTRUKTUR, ATAU SEJAUH MANA ANDA DAPAT MENYELESAIKANNYA ?

Bagaimana anda tahu jika suatu kotak pada level terendah mengalami kerusakan. Kotak pada level terendah itu harus digambarkan sebagai modul yang terstruktur. Ini akan dikodekan ke dalam modul program atau subprogram. Modul Yang terstruktur memiliki ciri-ciri sbb:

1. Berfungsi sepenuhnya sebagai fungsi tunggal. Misalnya dapat diterima, diedit, di format ulang dan melewati parameter tunggal.
2. Ukurannya kecil. Ukuran yang ditetapkan berkisar antara 50 - 100 baris yang dapat dieksekusi atau paling banyak 2 halaman.
3. Dapat diduga. Semua ciri dapat terlihat dengan membaca kode program. Hal ini tidak dipengaruhi oleh kode tersembunyi dalam modul lain atau sistem operasi.
4. (Paling penting). Independent. Perubahan dalam modul atau parameter tidak berpengaruh apa-apa dalam sistem. Misalnya: US Postal Services mengubah kode ZIP (1988) dari 5 digit ke 9 digit. Bayangkan berapa banyak programmer yang akan dipekerjakan selama bertahun-tahun untuk menyesuaikan sistem terhadap perubahan ini. Suatu modul independen yang baik akan memudahkan programmer untuk mengubah satu modul yang mengandung kode ZIP dan modul yang lain tidak akan terpengaruh atau dapat juga dengan hanya merubah kamus data.
5. Meskipun hal ini tidak dijelaskan secara jelas dalam modul terstruktur, lihatlah kegunaannya - suatu modul yang cukup lengkap dan cukup umum sehingga anda dapat menggunakannya dalam aplikasi lain dengan modifikasi sedikit mungkin. Untuk lebih jelasnya baca artikel asli dalam referensi 20.

Pesan untuk programmer

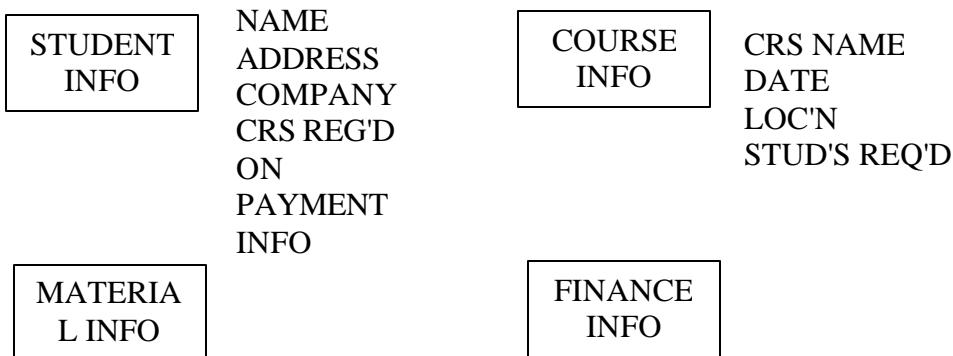
Jika anda membagi modul terus-menerus, maka pada akhirnya anda akan mengakhiri penggambaran kode program dalam bahasa Inggris. Programmer tidak menerjemahkan "pseudo-code" baris demi baris hingga menjadi barisan program. Untuk lebih jelas mengetahui proses ini lihat bagian 9.3. Proses ini dinamakan desain modul.

7.8 DESAIN FILE

Mendapatkan gambaran yang nyata

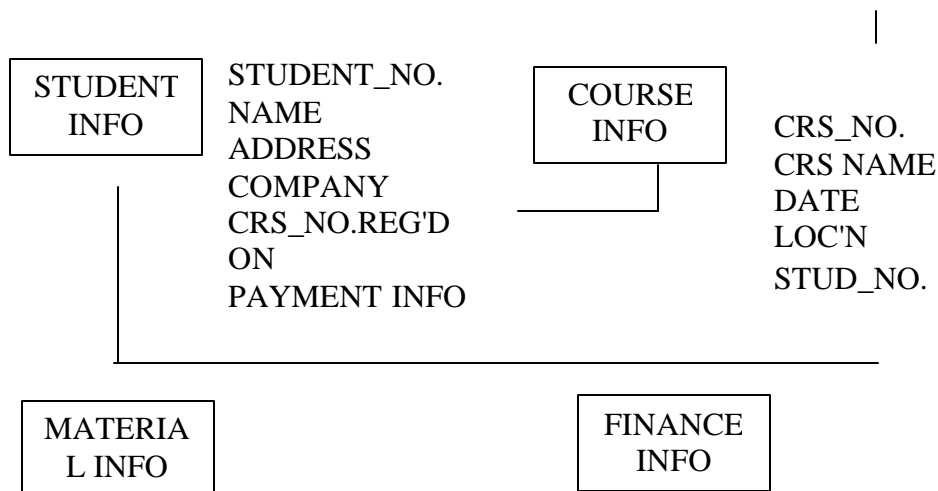
Desain file akan membuat /memecahkan aplikasi anda. Terutama ketika menggunakan Bahasa Generasi ke IV (lihat Bab 16). Beberapa perancang bahkan advokat mendesain file-file sebelum yang lainnya. Mari kita desain file system ABC menggunakan system file indexed sequential seperti ISAM

milik IBM atau RMS milik DEC. Anda mulai mendesain dengan melihat hasil dari fase analisis, persyaratan dan level paling atas dari hasil desain. Disarankan untuk kembali ke diagram alur data. (gbr. 6.1) dan gambarkan semua tipe informasi yang disebutkan. Hasilnya akan dikotakkan pada gbr.7.6.



Gbr.7.6 Tipe data

Sekarang mari kita lihat persyaratan detailnya (lihat dokumen persyaratan di Appendix A) dan coba alokasikan setiap item data yang ada dalam persyaratan ke salah satu kotak. Tambahkan kotak jika perlu. Misalnya, persyaratan "pendaftaran murid pada suatu kursus" akan menghasilkan penambahan dalam field yang tertulis di samping setiap kotak pada gbr.7.6. Kemudian, sadari bahwa proses logika dibutuhkan untuk menangani persyaratan. Jika STUDENT INFO dan COURSE INFO adalah file yang terpisah, mereka akan dihubungkan dengan sebuah key. Tambahkan key STUD_NO dan CRS_NO dan logika akses dapat digambarkan dengan tanda panah, seperti pada gbr. 7.7

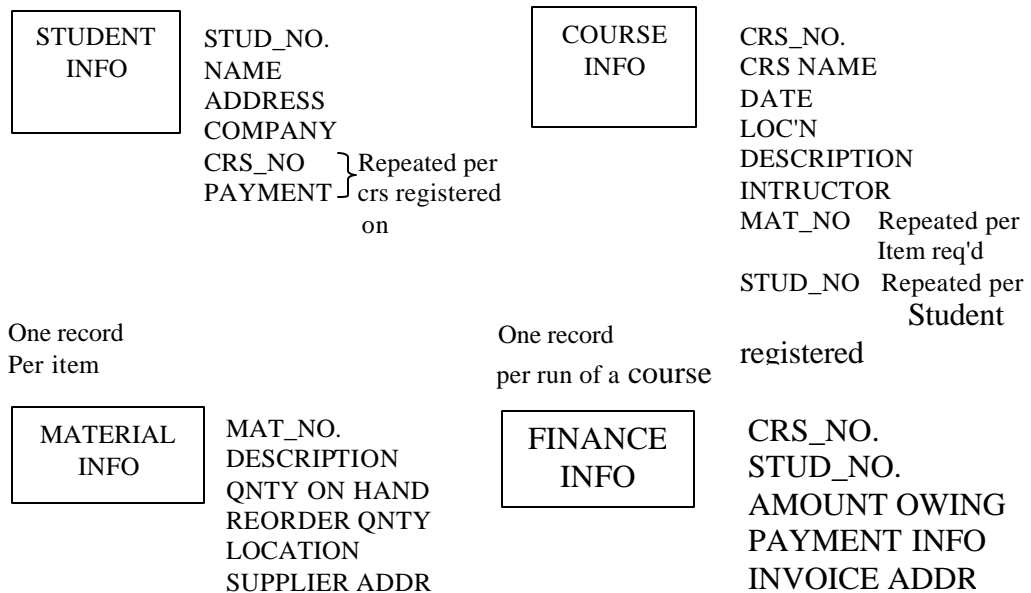


Gbr. 7.7 Tipe data, key dan akses

Sekarang kita dapat menangani pendaftaran murid sebaik-baiknya seperti: "memberikan nama murid, mencari kursus apa saja yang ia ikuti" (akses file "STUDENT FILE dengan nama, kemudian dapat CRS_NO, lihat COURSE FILE dengan CRS_NO tersebut). Diagram dilanjutkan sampai semua pernyataan terpenuhi. Hasilnya dapat digambarkan pada gbr. 7.8.

One record
per student
enrolled
on a course
not yet run

One record per
unique course/
location/date



Gbr. 7.8. Tipe data, key dan akses

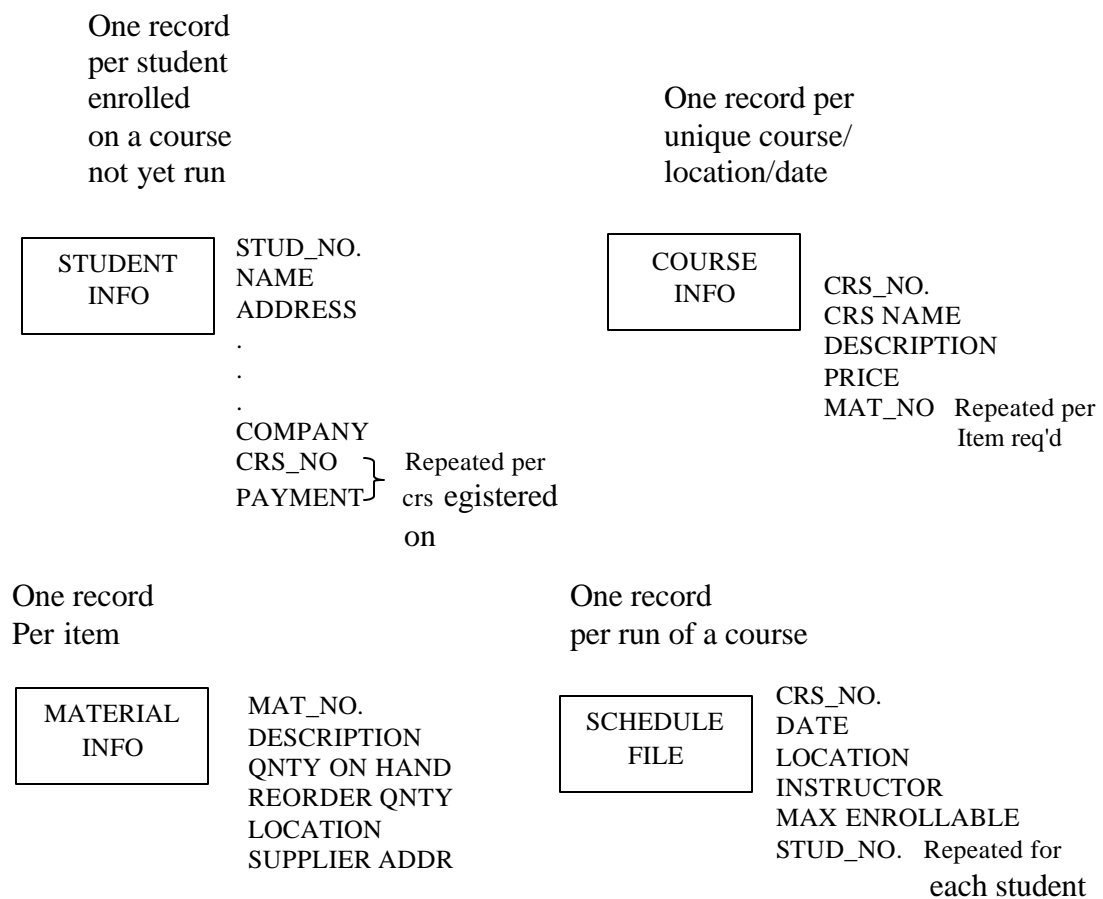
Optimalitas File

Langkah selanjutnya adalah mengoptimalkan penyimpanan disket dengan mengeliminasi kerangkapan field-field dan file-file.

Pada STUDEN FILE, jika banyak pelajar mempunyai alamat yang sama, seperti perusahaan yang sama, field alamat akan terulang. File ADDRESS dengan satu record setiap perusahaannya dan COMPANY_NO di record pelajar menunjukkan hal itu. File ini juga dapat berisi alamat faktur yang dibutuhkan oleh FINANCE FILE.

Pada item-item COURSE FILE seperti DESCRIPTION, INSTRUCTOR dan MATERIAL NO akan selalu terulang setiap mereka kursus pada tempat yang sama. Kemudian di pecah menjadi file baru yang bernama SCHEDULE FILE yang mempunyai item yang unik untuk setiap kursus dimulai, dan membiarkan COURSE FILE hanya dengan tipe informasi yang dependent.

FINANCE FILE hanya adapat di tandai/key dengan STUD_NO atau COURSE_NO. Sudah ada beberapa file yang menggunakan key tersebut, yang biasanya mengindikasikan field tersebut dalam file ini dapat menggabungkan dengan beberapa file yang lain, jika pembayaran dan informasi tentang faktur dapat digabungkan dengan pelajar. File FINANCE FILE tidak akan dibutuhkan. Hasil Desain dapat anda lihat pada gbr. 7.9.



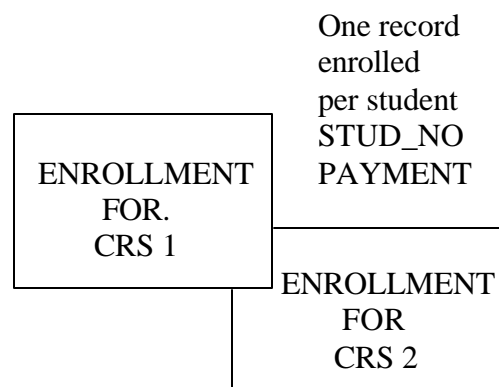
Gbr. 7.9 Tipe data, key dan akses

Mengoptimisasi Sejumlah Item Variabel

Dalam file STUDENT terdapat 2 field yaitu CRS_NO dan DYMNT, yang berulang di setiap Bidang Studi yang diambil siswa. Hal yang sama, yaitu pengulangan Field juga terjadi pada file SCHEDULE dan COURSE. Hal ini dapat diatasi dengan membuat program yang menggunakan file yang mempunyai ukuran yang dapat berubah-ubah. Panjang dari sebuah record berubah sesuai dengan penambahan maupun penghapusan sebuah item. Metode ini dapat menghemat ruang penyimpanan.

Dalam hal lain, jika jumlah maksimum dari suatu variabel diketahui maka dapat digunakan record yang memiliki panjang yang tetap. Sebagai contoh, jika suatu bidang studi tidak akan pernah diambil oleh lebih dari 30 siswa, maka jumlah record di file SCHEDULE disediakan untuk 30 siswa. Metode ini menggunakan ruang penyimpanan yang lebih besar dari metode pertama, namun metode ini membutuhkan waktu proses yang lebih singkat. Selain itu, record dengan ukuran yang tetap lebih mudah untuk dirancang, dimengerti, dan dalam hal pemeliharaannya dibandingkan dengan record yang memiliki panjang yang berubah-ubah. Karena harga media penyimpanan yang semakin murah saat ini, maka disarankan untuk menggunakan record yang memiliki ukuran tetap jika memungkinkan.

Sebuah masalah dapat muncul jika suatu batasan tidak dapat ditentukan. Sebagai contoh, jumlah siswa di setiap bidang studi tidak dapat dibatasi dengan jumlah yang sama. Untuk mengatasi hal ini dapat dibuat suatu file lain yang hanya digunakan untuk menyimpan informasi yang tidak tetap. File ENROLLMENT dapat dibuat untuk setiap bidang studi, dan setiap file mengandung 1 record per 1 siswa yang mengambil bidang studi tersebut (Gambar 7.10). Hal ini mungkin butuh biaya yang besar baik dalam media penyimpanan maupun dalam hal pemeliharaan file.



Gambar 7.10 Penanganan informasi yang berubah-ubah

Cara mudah untuk menangani masalah tersebut adalah membuat file yang bernama ENROLLMENTS, yang mengandung 1 record per siswa untuk setiap bidang studi yang diambil. Field-field yang digunakan hanya STUDENT_NO dan COURSE_NO. (pertanyaan No. 8 pada akhir bab ini meminta anda untuk merancang masalah ini).

File Histori

Apa yang akan kita lakukan terhadap data siswa yang telah menyelesaikan bidang studi yang telah diambilnya ?. Baik para akuntan maupun staf marketing akan menganggap bahwa informasi ini tidak penting dan harus dihapus, namun kita tidak ingin bekerja dengan file yang mengandung informasi yang sudah usang dan tidak akurat. Hal ini dapat dipecahkan dengan membuat file STUDENT_HISTORY, setelah siswa selesai mengikuti bidang studi tertentu, recordnya di file STUDENT akan dipindahkan ke file histori.

Pengujian Desain File

Pada desain ini, setiap permintaan kebutuhan yang melibatkan pengaksesan data harus "diproses" dengan desain file. Hal ini menandai perkembangan selanjutnya. Sebagai contoh jika ada permintaan "Tampilkan semua peristiwa pada tempat pelatihan XYZ, lokasi dan harga". Berikut logika pengaksesannya. Nama bidang studi diubah menjadi CRS_NO. Record-record pada file SCHEDULE diakses oleh CRS_NO untuk mendapatkan harga. Dalam permintaan yang umum semacam ini, mungkin nama bidang studi dapat dijadikan "key" pada file SCHEDULE. Mungkin harga bisa ditambahkan ke dalam file SCHEDULE untuk mengurangi pengaksesan file COURSE. Untuk menghemat ruang penyimpanan, kode harga dapat digunakan.

Pastikan bahwa setiap permintaan dapat dipenuhi. Carilah kemungkinan permintaan lain di masa depan, walaupun pemakai tidak memintanya. Sebagai contoh, bagaimana cara memproses permintaan seperti: "tampilkan semua bidang studi yang menggunakan bahan X", atau "Tangani kenaikan harga pada 6 bulan mendatang dengan menaikkan harga secara bertahap dan harga baru setelah itu. Kita tidak menangani masalah langsung ketika masalah itu muncul. Masalah yang muncul dicatat dan akan dipecahkan kemudian.

7.9 SISTEM MANAJEMEN DATABASE RELASIONAL (RDBMS)

Pada bagian 7.8 kita mengasumsikan bahwa anda mendapatkan suatu record dari suatu file yang mengandung key. Dalam kenyataannya harus ada suatu DBMS untuk memenuhi hal ini. Sebagian besar perusahaan minikomputer dan mainframe menyatakan DBMS dalam sistem operasinya. RMS dan RDB biasa digunakan dalam VMS. Contoh yang digunakan Contoh yang digunakan pada bagian 7.8 menggunakan DBMS terindeks, yaitu RMS. Sebagai manajer,

anda tidak harus mengerti bagaimana cara DBMS bekerja, namun anda dapat memutuskan DBMS mana yang akan digunakan. Berikut adalah contoh kasus dari sistem relasional.

Organisasi data relasional sebenarnya sangat mudah untuk dimengerti dan dibentuk. Dalam DB, setiap item diekspresikan sebagai tabel atau relasi. Baris pada tabel (tupel) disebut juga dengan rekord, sedangkan kolom (domain) disebut field. Aturan dalam membuat suatu tabel adalah bahwa setiap tupel (rekord) dalam suatu tabel harus bersifat unik, dan tidak ada field yang berulang pada tabel lain kecuali jika field tersebut diperlukan untuk menggabungkan suatu tabel dengan tabel lain. Menggabung berarti mencari suatu rekord pada suatu relasi dengan suatu field yang terdapat pada tabel lain. Gambar 7.11 adalah contoh dari suatu relasi (tabel) yang dapat digolongkan pada sistem ABC. Perhatikan kemiripan contoh ini dengan contoh desain file pada gambar 7.9 dan pertanyaan no.8 nama file menjadi nama relasi (tupel), rekord dan field menjadi baris dan kolom. Memasukkan data pada tabel seperti pada gambar 7.11, telah memenuhi segala persyaratan untuk membentuk suatu database.

Students relations:

Stud-No	Stud-Name	Company-No	all items unique to student
1	JOHN BLAKE	999	
2	JANE SMITH	999	

Run of a course relations :

Course-No	Course-date	Location
123	1/1/90	OTTAWA
123	1/2/90	NEW YORK

Enrollment relations :

Course-No	Stud-No	Pymnt
123	1	0

Course relations :

Course-No	Crs-name	Desc	Mat-No	items unique to a course
123	WEAVING	INTRO	001	

Company relations :

Comp-No	Addr	Ship-To	Bill-To	Tot-Owing
999	FIRST ST.	X Y	A B	10000

Material relations :

Mat-No	Desc	Whse	Source	Cost
001	STRAW	1-1	X Co	1.00

Gambar 7.11 Relasi pada database relasional ABC

Seperti pada metode nonrelasional. DBMS akan mengambil field berdasarkan field yang lain. Sebagai contoh, jika anda menginginkan daftar dari siswa pada COURSE_No '123', maka sistem relasional akan mencari pada tabel ENROLLMENT berdasarkan nama tersebut. Kemudian dibentuk suatu tabel yang berasosiasi dengan STUD_NO, lalu mencari STUDENT berdasarkan STUD_NOs untuk membentuk tabel STUD_NAMES. Keunggulan dari RDBMS adalah bahwa tahap-tahap diatas akan dilakukan secara otomatis. Sangatlah bijaksana untuk membentuk suatu relasi dengan mempertimbangkan hal-hal yang diperlukan pada masa mendatang, karena terdapat banyak kebutuhan tak terduga yang berhubungan dengan pembuatan tabel.

Bentuk dari query database ad-hoc seperti ini telah terstandarisasi dan disebut sebagai Bahasa Query Terstruktur (SQL). Sebagian besar produsen database relasional telah menyertakan SQL dalam produknya. Oracle adalah salah satu produk yang terkenal menggunakan SQL. Sebagai contoh, dibawah ini akan meminta SQL menampilkan pada bidang studi apa saja 'Smith' terdaftar.

```
SELECT CRS-NAME FROM COURSE
WHERE COURSE-NO IN
      SELECT COURSE-NO FROM ENROLLMENT
      WHERE STUD-NO IN
            SELECT STUD-NO FROM STUDENT
            WHERE STUD-NAME = 'SMITH'
```

Sayangnya, SQL yang dibuat oleh IBM, memiliki banyak tambahan komponen, sehingga suatu produk lama harus diperbarui bahasanya agar dapat menjalankan suatu komponen baru. Anda dapat melihat banyak bahasa 4GL yang baru (yang mendukung query, contohnya (QBE)) pada daftar di bawah ini.

STUD_NAME = Smith

CRS_NAME = _____

"

Kelebihan utama dari RDBMS adalah fleksibilitasnya. Sebagai contoh, jika kita ingin mengakses data yang sejenis secara berlainan dari berbagai aplikasi, maka sistem ini akan dapat menampungnya. Jika anda menginginkan query ad-hoc (sangat sulit untuk membayangkannya secara pasti),RDBMS adalah alat yang tepat untuk digunakan. Di bagian 16.6 Computer pada Aided

Software Engineering, kita membicarakan penggunaan bahasa generasi keempat bersamaan dengan RDBMS untuk mengotomatisasikan hampir semua aplikasi komputer versi yang terdahulu.

Satu-satunya kekurangan RDB adalah penggunaannya yang membutuhkan terlalu banyak memori dan waktu untuk menyimpan, menjalankan dan merinci seluruh bagian tabel. Namun dalam siklus harga komputer cenderung semakin murah. Penghematan waktu yang diakibatkan oleh pemakaian yang mudah dan fleksibilitas dari sistem, membuat RDBMS yang terdapat pada sebuah komputer (CPU) yang baik sebagai investasi yang berharga,

7.10 KEUNTUNGAN DARI STRUKTUR ANALISA DAN DESAIN

Pengurangan Jumlah Kesalahan

Statistik data yang ada diambil dari hasil survei oleh TRW untuk proyek besar dan DEC'S Customer Services System Engineering (yaitu departemen yang bertanggung jawab dalam memastikan bahwa produk-produk DEC' baik software maupun hardwarenya benar-benar bebas dari virus / kesalahan).

USING INSTRUCTURED METHODS

	ANALYSIS	REMAINING	AFTER
	AND	PHASE	OPER
	DESIGN		
EFFORT SPENT:	10%	23%	67%
PROBLEMS INTRODUCED:	64%	36%	
PROBLEMS FOUND:	19%	27%	54%
DOLLARS SPENT(AVG)	25K	57.5K	167.5K
(TOTAL=250K)			

USING INSTRUCTURED METHODS

	ANALYSIS	REMAINING	AFTER
	AND	PHASE	OPER
	DESIGN		
EFFORT SPENT:	20%	50%	30%
PROBLEMS INTRODUCED:	32%	68%	
PROBLEMS FOUND(1):	30%	33%	37%
DOLLARS SPENT(AVG)	40K	50K	100K
(TOTAL=250K)			

(1) Setengah dari sebelumnya

Gambar 7.12 Kasus-kasus dan biaya dari masalah

Gambar 7.12 menunjukkan bahwa meskipun biaya dimuka mengalami kenaikan, metode terstruktur tetap mengurangi biaya sistem secara keseluruhan ada beberapa keuntungan lain untuk menemukan kesalahan-kesalahan up front dari later on dalam siklus. Data statistik menunjukkan bahwa adalah 100 x lebih mahal bila memperbaiki kesalahan dalam analisa setelah penerimaan (ACC) dibandingkan pada saat menganalisanya.

7.11 PROSES DESAIN

Tim Desain

Pilihlah orang yang terbaik sebagai tim perancang. Tim perancang yang baik tidaklah perlu orang yang menguasai bahasa pemrograman. Mereka haruslah orang yang dapat mengkonsep semuanya. Hindari orang-orang yang selalu menginginkan kesempurnaan dalam tim perancang. Gunakanlah waktu yang memadai dalam mengerjakan perancangan, namun kita juga harus memperhatikan keterbatasan waktu dan biaya yang ada. Sejak perkembangan transaksi dan keputusan yang terjadi selama waktu perancangan, lebih baik untuk memiliki sejumlah orang dalam tim secara ganjil, atau setidaknya memiliki seorang moderator yang baik.

Pertemuan untuk Pembuatan Desain

Merancang sesuatu ibaratnya adalah mencari inspirasi : beberapa orang berkumpul dalam suatu ruangan yang tenang dan tidak terganggu. Setiap orang diharapkan dapat mencurahkan semua ide mereka agar semua elemen yang berfungsi dapat digunakan dan juga memikirkan bagaimana cara menguasainya. Sejak ide-ide berjalan dengan sembarang perintah, ini membuktikan bahwa ada sarana untuk mendapat semua ide yang ada. Tulis semua ide yang ada, dan kemudian akhirnya ide-ide yang ada disusun kedalam suatu modul yang unik.

7.12 DOKUMENTASI TEKNIK

Spesifikasi rancangan merupakan suatu dokumentasi teknik. Dokumentasi ini ditujukan untuk dibaca dan dimengerti oleh para pembuat program. Pemakai boleh membuat desain tersebut tapi mereka tidak harus mengerti tentang itu. Pertimbangkan hal-hal ini dalam membuat dokumentasi teknik:

1. Gunakan secara formal/resmi, bentuk yang sesuai. Sumber terbesar yang kedua dari kerusakan-kerusakan /gangguan-gangguan pada sistem perangkat lunak adalah ketika pembuat program salah menginterpretasikan/mengartikan rancangan. (Sumber terbesar,sekilas,

adalah ketika penganalisa/pengamat salah mengartikan/salah mengerti akan kebutuhan pemakai.) Baca hukum teks hukum. Ini tidak berbelit-belit atau sulit untuk dipahami. Para pengacara mencoba untuk menggunakan bahasa yang tidak dapat menimbulkan salah pengertian.

2. Gunakan gambar--semacam diagram dan sejenisnya.
3. Buatlah agar maksud/pengertian dari desain tersebut jelas pada beberapa lembar pertama. Lalu teliti kembali.
4. Cobalah untuk konsisten pada grafik-grafik dan kalimat yang terstruktur. Yang terbaik adalah ketika seseorang menuliskan semuanya itu. Jika tenggang waktu mendesak anda untuk memakai beberapa orang, pastikanlah bahwa mereka menggunakan gaya yang sama.

7.13 MENDIKTE STANDAR-STANDAR SAAT PENDESAINAN

Hal-hal tertentu harus dapat selesai dengan cara yang sama tidak terkecuali siapa saja yang melakukannya. Hal ini khususnya benar pada saat memasuki tahap-tahap membuat program, dimana bentuk yang paling paralel dapat menjadi dominan. Anda mungkin bisa mengerutkan dahi melihat bentuk standar "birokrasi" yang mengesankan namun membentuk aturan-aturan seperti yang berikut ini :

- *Desain Konvensional.* Metode putus-turun, format diagram struktur. Modul dan variabel dinamai dengan kaidah tertentu. Ini biasanya digunakan pada semua level tingkat rendah.
- *Jalur Parameter.* Rincian perintah, panjang, format, tempat penyangga jika jika hilang dan seterusnya.
- *Penanganan Gangguan.* Standar-standar yang dibuat secara khusus menyarankan bahwa penanganan pada sebuah gangguan diperlukan setiap konteks yang dilalui modul (situasi dimana gangguan terdapat) dan gangguan yang harus ditangani. Penanganan menampilkan pesan pesan gangguan. Jaminan-jaminan ini tetap menangani gangguan, tetapi penampilan bisa lebih buruk pada segala panggilan tambahan pada penanganan gangguan.
- *Standar-standar Pemrograman.* Standar-standar pemrograman terstruktur seperti pemunculan kode (jarak putih, pemasukkan, komentar), gagasan diperbolehkan, organisasi, ukuran modul, dan ketergantungan secara terperinci. Menciptakan 'template' atau kerangka yang berisi komentar untuk :

kop (judul, pengarang, tanggal, modifikasi sejarah)

parameter-parameter (menerima, mengirim)

masuk (hanya satu)

variabel yang digunakan

memanggil subrutin

penanganan gangguan

keluar (hanya satu)

- Pembuat program memulai dengan 'template' ini dan mengisi kode proses. Lihat bagian 9.4 untuk alat pemrograman yang membantu format program secara tetap.

Jika anda mengatur standar-standar ini, maka anda akan bisa menggunakannya untuk proyek-proyek yang lain.

7.14 GARIS BESAR DARI SPESIFIKASI RANCANGAN

Sebagai dokumen pembuat rencana, Appendix A memberikan contoh nyata.

1. *Judul halaman dan tabel yang dimuat* semua bagian dengan nomor halaman.
2. *Pandangan* Walaupun saat seorang pembuat program membaca dokumen yang berisi catatan tentang permintaan dan spesifikasi fungsional, spesifikasi desain harus dimulai dengan ringkasan dari permasalahan, solusi umum dan bagaimana sebuah sistem akan cocok bagi lingkungan pemakai. Ini dibuat spesifikasi rancangan sebagai dokumen yang dapat berdiri sendiri.
3. *Perangkat keras/Perangkat lunak* Daftar dari perangkat keras yang berada pada sistem yang akan berjalan. Daftar dari operai sistem dan versi, semua perangkat lunak, program utility serta bahasa pemrograman yang akan digunakan.
4. *Perioritas rancangan* Daftar perioritas utama dalam urutan menurun. Prioritas rancangan akan dibahas pada bagaian 7.3 mengutamakan pertukaran yang mungkin dapat dibuat.
5. *Rancangan diagram dan kumpulan modul kamus* Menjelaskan struktur dari kumpulan diagram : bagaimana setiap kotak menemukan fungsi dari kotak tersebut dan bagaimana data dikirim selama kotak tersebut ada. Memberitahu programer bahwa selama ia memecah modul rancangan, ia harus mengikuti diagram yagn sama dan menyimpannya dalam kamus perjanjian.
6. *Modul nama perjanjian* Menjelaskan nama perjanjian yang dijelaskan pada bagian 7.5
7. *Memberikan Parameter & Kamus Data* Daftar aturan-aturan untuk memberikan parameter diantara modul. Menunjukkan dimana kamus data ditemukan dan bagaimana mereka diatur. Sejauh ini kamus dat berisi semua parameter yang dibatasi untuk TLD & MLD. Programer akan dapat menambah beberapa parameter baru yang didefinisikan untuk submodul. Contohnya tipe format CALL.
8. *Pengendalian Kesalahan* Mendefinisikan bagaimana kesalahan akan menunjukkan kepada programmer bagaimana cara memenggilnya dan kemudian memperbaikinya. Berikan contohnya.
9. *Struktur Pemrograman Standar* Daftar dari standar diskusi pada bagian 7.12 menunjukkan bagaimana model program dapat ditemukan.
10. *Alamat Pemrograman* Pengkodean dan alat pengujian seperti Editor / Language Sensitive Editor, Compiler, debug, Tester Automata dan

Source Code Analyzer akan didiskusikan di bagian 9.4. Ini semua akan membuat kerja program dan debug menjadi lebih mudah. Jika semua alat ini ada maka mereka dapat diakses dan digunakan.

Di bagian 9.4 kita akan mendiskusikan kode sistem manajemen (CMS) untuk menyimpan kode sumber dan track yang telah dirubah. Jika ada CMS yang dapat digunakan, tunjukkan bagaimana. Daftar setiap pengulangan sumber atau kode object yang tersedia sebagaimana yang telah ada, modul akan dapat digunakan kembali. Jangan membuat pengulangan. Jika spesifikasi paket software seperti DBMS atau FMS digunakan, jelaskan bagaimana dan kapan mereka digunakan.

11. *Desain Level Atas* Diagram struktur TLD terdapat pada gambar 7.1. singkatnya menjelaskan tentang TLD, dan fungsi umumnya ditampilkan oleh 5 komponen utama. Jelaskan bagaimana komponen yang satu dengan yang lainnya dapat cocok. Ceritakan semua bagian yang dimiliki oleh desain level menengah.

12. *Desain Level Menengah* Termasuk semua struktur diagram MLD. Jelaskan semua fungsi umum dari setiap modul atau kotak. Contohnya: Modul AM000000 memberikan kontrol ketika operator mengetik ABC di level perintah (mungkin dapat secara otomatis dijalankan oleh login file). Modul tersebut memanggil AMST000 untuk membuka semua sistem dan mengerjakan beberapa inisialisasi. Dan selanjutnya semua fungsi umum dari modul dirinci.

Untuk garis besar dari modul level bawah menggunakan format berikut ini:

```

Module name:
Called by:          list all callers
Subroutines called: (to be filled in by programmer)
Input parameters:  list
Displays:          the I/O with the terminal or user
Returned parameters: list
External variabls used: list
Files used:        list
Functions:         list ini English statements. If you have
                  pseudocode for each module, it can go
                  here.

```

Dan seterusnya sampai modul level menengah dirinci.

13. *Modul dan Kamus data:* Menjelaskan konstruksi dan penggunaan 3 kamus yang dibahas di bagian 7.6. Menjelaskan dimana CDD berada dan bagaimana menggunakannya. Tunjukkan bagaimana menampilkan CDD. Untuk mengetahui apakah CDD telah berada pada level atas dan menengah dari desain dan apa yang ditambahkan pada proses desain.

14. *File dan Tabel* Memanggil kembali file yang berada di spesifikasi fungsi dan mendaftarkan elemen data yang akan kita simpan. Di dalam rancangan kita memperlihatkan bagaimana elemen-elemen ini akan dapat cocok bagi file. Untuk sistem ABC kita harus menyediakan file

KURSUS.DAT, SISWA.DAT dan lain-lain. Untuk setiap file menunjukkan organisasi (contohnya RMS), atribut, panjang record, key, dan modul yang digunakan dalam sistem tersebut.

Termasuk penempatan record secara mendetail setiap nama field, pajang, penempatan, dan lain-lain. Menunjukkan modul apa yang dapat diakses dan untuk keperluan apa.

Jangan lupa untuk memasukkan ketika sebuah file dibuat, berapa besar akan dibuat, dan berapa besar penambahan yang dapat ditangani.

Jelaskan struktur data lain yang akan digunakan, seperti di dalam memori tabel-tabel dan array-array.

7.15 MENGUJI DESAIN

Ketika desain telah diselesaikan, semuanya harus berjalan dengan benar. Maksudnya adalah untuk menjamin hal-hal berikut ini :

1. Semua keperluan spesifikasi fungsional sudah ketemu.
Lakukan ini dengan melalui FS, kalimat demi kalimat. Untuk beberapa fungsi yang disajikan di FS, perencana harus mendapatkan modul dan menyampaikannya, "kita dapat mengatasinya disini" sebaliknya semua fungsi desain perlu disebut di FS. Menjamin bahwa tidak ada bell dan melodi yang ditambahkan oleh perancang.
2. Desain mudah diprogram dan dipelihara.
Ini akan menjadi kasus jika terstruktur, pendekatan digunakan sedikit demi sedikit. Lihat dari hal kecil, mandiri, dan dapat cukup dimengerti.
3. Hal ini dapat diimplementasikan dalam waktu dan anggaran.
Ini sebuah pertanyaan subyektif yang hanya dapat dijawab oleh pemimpin desain. Pertanyaan yang harus ditanyakan adalah :
 - Sudahkah semua komponen software & hardware telah didesain dan ditunjukkan untuk bekerja dengan semestinya.
 - Apakah hal ini mudah, mendesain yang lebih maju ?
 - Apakah setiap komponen masih dapat diperkirakan, apakah perkiraan masih dalam ballpark yang asli ?

7.16 MERUBAH PERMINTAAN SESUAI DENGAN DESAIN

Beberapa dari desain yang detail akan tetap membimbing pada perubahan permintaan. Anda harus kembali kepada pemakai sekarang dan menyakinkan bahwa sebenarnya dia tidak menginginkan apa yang dia minta sebelumnya. Seperti sebelumnya argumentasi anda harus berdasarkan pada biaya/keuntungan. Jika perubahan yang ditunjukkan dapat menghemat uang dalam pengembangan/pemeliharaan lakukan desain yang sederhana yang anda anjurkan, pemakai harus setuju. Jika anda bertahan dalam kontrak harga mati

gunakan argumentasi "Kita dapat melakukannya dengan jalan yang telah disetujui sebelumnya, dan kita tahu kita tidak bisa menaikkan harga, tapi kita bisa coba dalam 3 bulan kedepan ", itulah sebabnya saya menganjurkan anda pada sebuah kontrak sampai desain itu selesai.

7.17 PERENCANAAN PENERIMAAN

Meskipun penerimaan dengan sendirinya adalah sebuah tahapan, merencanakan penerimaan dapat dimulai desain tingkat menengah selesai. Mempersiapkan perencanaan penerimaan yang diberikan pada bab berikutnya semenjak aktifitas kronologis berikutnya. Hal ini dapat diselesaikan kapan waktu setelah yang berikut ini selama diselesaikan oleh tingkatan penerimaan.

PERTANYAAN

1. Apa tujuan dari tingkatan desain ?
2. Apakah yang dimaksud desain terstruktur itu ? Apakah yang dimaksud dengan desain top-down ?
3. Apakah desain bottom-up itu ? Tipe desain apa yang biasanya digunakan oleh desain bottom-up ?
4. Apakah yang mungkin bisa menjadi efek pengaruh dari penggunaan 'User Friendliness' sebagai prioritas desain yang paling tinggi ?
5. Apakah tujuan dari desain top level secara keseluruhan ? Apa yang dapat menjadi masalah utama dan bagaimana anda menghindari itu ?
6. Apakah yang dimaksud dengan kamus data ? Dan kenapa ini bermanfaat ?
7. Kenapa memecah sebuah sistem menjadi modul-modul terstruktur ?
8. Menurut desain file dalam bagian 7.8 menganggap bahwa dua permintaan yang sama dalam sistem bisa jadi "berapa banyak mahasiswa yang terdaftar dalam kursus no. NNN ?" Dan "Di kursus apa mahasiswa tersebut terdaftar ?" Dapatkah anda menggambarkan sebuah desain file dimana kedua pertanyaan itu dapat dijawab dengan akses dalam satu file saja ?
9. Apa keuntungan dan kerugian dari penggunaan sistem manajemen database relasional ?
10. Apakah keuntungan dari analisis struktur dan desain ?
11. Kepribadian yang bagaimana yang diperlukan pada seorang perancang sistem ?
12. Hal apa yang harus distandarisasi oleh perancang dan mengapa ?
13. Apa tujuan dari desain tingkat menengah secara keseluruhan ?
14. Kerja kelompok : Tulis spesifikasi desain untuk 'sistem komunikasi keluarga 'Bell '

JAWABAN :

1. Tujuan dari tingkatan desain adalah untuk membuat pembagian tingkatan dalam tahap pendesainan yang terdiri dari :
Kegiatan utama tahap desain adalah membuat tingkat tertinggi dan tingkat menengah dari desain sistem dan mendokumentasikannya dalam perincian desain. Kegiatan kedua dalam tahapan ini akan mulai menerima rencana pengujian atau *Acceptance Test Plan* (ATP). ATP adalah sebuah dokumen laporan pengujian yang akan digunakan untuk mendemonstrasikan semua fungsi sistem kepada pemakai pada tahap penerimaan.
2. Desain terstruktur adalah desain yang dibuat terstruktur dengan memecah sistem menjadi lebih kecil , dapat diatur, dan dapat dibentuk bagian-baginnnya. Dan yang dimaksud dengan desain Top-Down adalah desain yang dimulai dengan tingkatan yang paling tinggi, menurun ke tingkat selanjutnya dan seterusnya.
3. Desain Bottom-Up adalah desain dari tingkat yang terendah ke tingkat yang lebih tinggi. Hal ini sering ditemui pada kasus sistem kontrol proses dimana perangkat keras kontrol pada tingkat bawah menentukan bagaimana sistem tersebut disatukan (integrasi sistem). Desain Bottom Up cocok digunakan pada kasus dimana komponen perangkat lunak yang ada digabungkan dan disatukan dengan modul baru untuk membangun sebuah sistem.
4. Yang bisa menjadi pengaruh dari penggunaan “User Friendliness” sebagai prioritas desain yang paling tinggi adalah : - Kemampuan user
- Kesederhanaan
5. Tujuan dari desain tingkat atas secara keseluruhan adalah untuk menentukan :Biaya Sistem, Waktu yang diperlukan untuk membangun sistem, Sifat mudah dipakai, Penampilan (kinerja), Ukuran Sistem, Keandalan, Kemampuan modifikasi.
Yang dapat menjadi masalah utama adalah saat membuat desainnya dan cara mengatasinya memecahkan sistem ke dalam komponen-komponen utama. Metode yang lain mungkin dengan menggunakan Sistem Manajemen Berbasis Data.
6. Yang dimaksud dengan kamus data umum (CDD) adalah kamus data yang mengandung semua parameter yang didefinisikan pada level yang terendah dari pemrograman dan desain sebagaimana field didefinisikan dalam suatu file. Dan manfaat kamus data umum (CDD) adalah memastikan bahwa parameter akan konsisten berlaku dalam seluruh sistem.
7. Memecah sistem menjadi modul yang terstruktur maksudnya adalah untuk mempermudah dalam Berfungsi sepenuhnya sebagai fungsi tunggal. Misalnya dapat diterima, diedit, di format ulang dan melewati parameter tunggal. Ukurannya kecil. Ukuran yang ditetapkan berkisar antara 50 - 100 baris yang dapat dieksekusi atau paling banyak 2 halaman. Dapat

diduga. Semua ciri dapat terlihat dengan membaca kode program. Hal ini tidak dipengaruhi oleh kode tersembunyi dalam modul lain atau sistem operasi. (Paling penting). Independent. Perubahan dalam modul atau parameter tidak berpengaruh apa-apa dalam sistem. Suatu modul independen yang baik akan memudahkan programmer untuk mengubah satu modul yang mengandung kode ZIP dan modul yang lain tidak akan terpengaruh atau dapat juga dengan hanya merubah kamus data.

8. Melakukan query terhadap student yang mengambil CRS NO "NNN", serta menghitung jumlah student yang mengambil CRS tersebut dengan fungsi group by kemudian hasilnya dimasukkan database grid, setelah itu apabila ingin mengetahui CRS yang diikuti untuk salah satu student yang terdapat pada database grid, anda hanya perlu mengklik dua kali record student tersebut. Kemudian akan dilakukan query terhadap CRS_NO yang diambil student tersebut dan dimasukkan ke dalam list box.
9. Keuntungan penggunaan database relasional : terkontrolnya kerangkapan data, terpeliharanya keselarasan, data dapat dipakai secara bersama, dapat diterapkan standarisasi, keamanan terjamin, terpeliharanya integritas data, terpeliharanya keseimbangan, data independence.
 Kerugian penggunaan data base relasional : storage yang digunakan besar, dibutuhkan tenaga yang terampil dalam mengelola data, softwaranya mahal, kerusakan pada sistem database dapat mempengaruhi departemen yang terkait.
10. Keuntungan dari analisis struktur dan desain adalah pengurangan jumlah kesalahan.
11. Kepribadian yang diperlukan oleh seorang perancang sistem adalah seorang perancang yang baik tidaklah perlu orang yang menguasai bahasa pemrograman. Seorang perancang haruslah orang yang dapat mengkonsep semuanya. Bukan orang-orang yang selalu menginginkan kesempurnaan.
12. Hal-hal yang perlu distandarisasi oleh perancang adalah : desain konvensional, parameter passing, penanganan gangguan, standar pemrograman, programmer memulai dengan template dan mengisi kode proses. Hal ini dibuat agar untuk hal-hal tertentu dapat selesai dengan cara yang sama.
13. Tujuan dari desain tingkat menengah secara keseluruhan adalah memisahkan setiap fungsi atau komponen utama menjadi sub fungsi atau komponen