BAB II

LANDASAN TEORI

2.1 Algoritma *Tabu Search*

Tabu Search berasal dari Tongan, suatu bahasa Polinesia yang digunakan oleh suku Aborigin Pulau tonga untuk mengindikasikan suatu hal yang tidak boleh "disentuh" karena sakralnya. Menurut kasus Webster, Tabu berarti larangan yang dipaksakan oleh kebudayaan social sebagai suatu tindakan pencegahan atau sesuatu yang dilarang karena berbahaya. Bahaya yang harus dihindari dalam *Tabu Search* adalah penjadwalan yang tidak layak, dan terjebak tanpa ada jalan keluar. Dalam konteks lebih luas, larangan perlindungan dapat diganti jika terjadi tuntutan yang mendadak.

Tabu Search adalah sebuah metode optimasi yang berbasis pada *local search*. Proses pencarian bergerak dari satu solusi ke solusi berikutnya, dengan cara memilih solusi terbaik *neighbourhood* solusi sekarang (*current*) yang tidak tergolong solusi terlarang (*tabu*). Ide dasar dari algoritma *tabu search* adalah mencegah proses pencarian dari *local search* agar tidak melakukan pencarian ulang pada ruang solusi yang sudah pernah ditelusuri, dengan memanfaatkan suatu struktur memori yang mencatat sebagian jejak proses pencarian yang telah dilakukan.

Struktur memori fundamental dalam *tabu search* dinamakan *tabu list*. *Tabu list* menyimpan atribut dari sebagian *move* (transisi solusi) yang telah diterapkan pada iterasi-iterasi sebelumnya. *Tabu search* menggunakan *tabu list* untuk menolak solusi-solusi yang memenuhi atribut tertentu guna mencegah proses

pencarian mengalami *cycling* pada daerah solusi yang sama, dan menuntun proses pencarian menelusuri daerah solusi yang belum dikunjungi. Tanpa menngunakan strategi ini, *local search* yang sudah menemukan solusi optimum local dapat terjebak pada daerah solusi optimum local tersebut pada iterasi-iterasi berikutnya. *List* ini mengikuti aturan LIFO dan biasanya sangat pendek (panjangnya biasanya sebesar $O(\sqrt{N})$, dimana N adalah jumlah total dari operasi). Stiap saat ada langkah itu akan ditempatkan dalam *tau list*.

Perekaman solusi secara lengkap dalam sebuah *forbidden list* dan pengecekan apakah sebuah kandidat solusi tercatat dalam list tersebut merupakan cara yang mahal, baik dari sisi kebutuhan memori maupun kebutuhan waktu komputasi. Jadi, *tabu list* hanya menyimpan langkah transisi (*move*) yang merupakan lawan atu kebalikan dari langkah yang telah digunakan dalam iterasi sebelumnya utnuk bergerak dari satu solusi ke solusi berikutnya. Dengan kata lain *tabu list* berisi langkah-langkah yang membalikan solusi yang baru ke solusi yang lama.

Pada tiap iterasi, dipilih solusi baru yang merupakan solusi terbaik dalam *neighbourhood* dan tidak tergolong sebagai tabu. Kualitas solusi baru ini tidak harus lebih baik dari kualitas solusi sekarang. Apabila solusi baru ini memiliki nilai fungsi objektif lebih baik dibandingkan solusi terbaik yang telah dicapai sebelumnya, maka solusi baru ini dicatat sebagai solusi terbaik yang baru.

Sebagai tambahan dari *tabu list,* dikenal adanya *criteria aspirasi*, yaitu suatu Penanganan khusus terhadap *move* yang dinilai dapat menghasilkan solusi yang dinilai dapat menghasilkan solusi yang baik namun *move* tersebut berstatus tabu. Dalam hal ini, jika *move* tersebut memenuhi criteria aspirasi yang telah ditetapkan

sebelumnya, maka *move* tersebut dapat digunakan utnuk membentuk solusi berikutnya (status tabunya dibatalkan). Berikut ini diberikan kerangka umum algoritma *tabu search* dalam notasi bahasa pascal.

```
begin
{Buat solusi awal s yang feasibel dengan
menggunakan
suatu metode heuristik tertentu atau secara
acak}
best := cost(s);
s* := s; {s* adalah solusi terbaik yang
diperoleh}
tabu_list := null;
repeat
Candidate(s) := (s' ∈ N(s): merupakan move
dari s
ke s' yang tidak tergolong elemen dari tabulist
atau
memenuhi kriteria aspirasi};
(pilih s ∈ Candidate(s): s adalah solusi yang
memiliki nilai cost minimum );
(simpan move yang berlawanan ke dalam tabulist,
yaitu
yang mengubah s ke s );
s := s;
if (cost(s) < best) then
s* := s;
best := cost(s);
until (stopping-criteria = TRUE);
return(s*);
end;
```

2.1.1 Ide Dasar *Tabu Search*

Sebuah informasi akan digunakan sebagai petunjuk untuk bergerak dari i ke solusi selanjutnya dalam N(i). Penggunaan memori sebagai pembatas dalam pemilihan beberapa subset dari N(i) dengan mencegah pergerakan ke beberapa solusi tetangga. Sebuah informasi akan digunakan sebagai petunjuk untuk bergerak dari i ke solusi selanjutnya dalam N(i). Penggunaan memori sebagai

pembatas dalam pemilihan beberapa subset dari N(i) dengan mencegah pergerakan ke beberapa solusi tetangga.

Secara formal,kita dapat menganggap masalah optimalisasi dalam cara berikut: Diberikan sebuah himpunan solusi *S* dan sebuah fungsi *f : S*, temukan solusi *i** dalam *S* sehingga *f(i*)* dapat diterima dengan beberapa kriteria. Secara umum kriteria untuk dapat diterima sebagai solusi *i** harus memenuhi *f(i*)=f(i)* untuk setiap *i* dalam *S*. Dalam situasi metode pencarian tabu akan menjadi sebuah algoritma minimimasi secara pasti yang menyediakan proses eksplorasi yang menjamin setelah sejumlah langkah-langkah terhingga *i** dapat dicapai.

Untuk mendalami lagi prinsip kerja metode pencarian tabu, kita dapat memformulakan metoda menurun klasik (*classical descent method*) dalam beberapa langkah, yaitu:

1. Memilih sebuah solusi awal *i* dalam *S*
2. Membangkitkan subset *V** sebagai solusi dalam *N(i)*
3. Mencari j 'terbaik' dalam *V** dan menetapkan *i=j*
4. Jika *f(j)=f(i)* maka berhenti, namun jika tidak kembali ke langkah ke-2

Dalam metode menurun secara umum akan dapat langsung ditetapkan bahwa *V* = N(i)*. Tetapi hal ini seringkali membutuhkan waktu yang lama. Untuk itulah cara pemilihan *V** yang tepat seringkali dijadikan sebagai peningkatan yang penting dalam metode pencarian. Dalam metode menurun secara umum akan dapat langsung ditetapkan bahwa *V* = N(i)*. Tetapi hal ini seringkali membutuhkan waktu yang lama. Untuk itulah cara pemilihan *V** yang tepat seringkali dijadikan sebagai peningkatan yang penting dalam metode pencarian. Pada kasus yang berlawanan, akan ditetapkan *|V*|=1*. Hal ini akan

menurunkan fase pemilihan *j* 'terbaik'. Solusi *j* akan diterima jika *f(j)=f(i)*, sebaliknya hal ini akan diterima dengan kemungkinan tertentu yang bergantung pada nilai-nilai *f* pada *i* dan *j* serta pada sebuah parameter yang disebut temperatur. Tidak ada temperatur dalam metode pencarian tabu, namun pemilihan *V** akan menjadi hal yang penting guna mendefinisikannya dalam setiap langkah di mana akan terjadi penggunaan memori secara sistematis untuk memanfaatkan informasi yang ada di luar fungsi f dan lingkungan *N(i)*.

Sebagai pengecualian pada kasus-kasus istimewa, penggunaan prosedur menurun (*descent procedure*) secara umum lebih rumit karena kita akan terperangkap pada sebuah minimum lokal yang mungkin masih jauh dari minimum global. Maka untuk proses iterasi dalam eksplorasi apapun sebaiknya dalam beberapa hal juga menerima langkah-langkah yang tidak akan memberikan perkembangan dari i ke j dalam *V** (misal *f(j)>f(i)*). Metode pencarian tabu secara berbeda memilih *j* 'terbaik' dalam *V**.

Selama pergerakan yang tidak memberi perkembangan itu mungkin, resiko pengunjungan kembali sebuah solusi atau lebih umumnya disebut sebagai siklus mungkin untuk terjadi. Dalam hal inilah penggunaan memori sangat diperlukan untuk mencegah terjadinya pergerakan ke solusi yang telah dikunjungi. Jika memori seperti itu diperkenalkan, maka kita dapat menganggap struktur *N(i)* tergantung pada pengelilingan yang merupakan pengulangan k. Jadi kita dapat merujuk ke *N(i,k)* daripada *N(i)*.

Dengan perujukan ini kita dapat mencoba untuk melakukan peningkatan algoritma menurun dalam beberapa hal untuk lebih mendekatkan metode ini ke prosedur dalam metode pencarian tabu secara umum. Hal ini dapat didefinisikan

dalam beberapa langkah (catatan i^* adalah solusi 'terbaik' yang ditemukan dan k adalah penghitung dalam pengulangan) :

1. Memilih solusi awal I dalam S. Tetapkan $i^*=I$ dan $k=0$.
2. Tetapkan nilai $k=k+1$ dan membagkitkan subset V^* sebagai solusi dalam $N(i,k)$
3. Pilih j 'terbaik' dalam V^* dan tetapkan $i=j$.
4. Jika $f(i) < f(i^*)$ maka tetapkan $i^*=i$.
5. Jika kondisi berhenti ditemukan, maka proses dihentikan, sedangkan jika belum kembali ke langkah 2.

Amati bahwa langkah-langkah pada metode penurunan klasik termasuk dalam formula ini (kondisi berhenti secara sederhana *jika* $f(i)=f(i^*)$ dan i^* akan selalu menjadi solusi akhir). Selain itu kita juga dapat mempertimbangkan penggunaan f yang telah dimodifikasi daripada f yang dalam beberapa keadaan di deskripsikan kemudian.

Dalam metode pencarian tabu, kondisi berhenti dapat berupa:

1. $N(i,k+1)$ = tidak terdefinisi
2. k lebih besar daripada bilangan maksimum pada pengulangan
3. Banyaknya pengulangan selama peningkatan terakhir i^* lebih besar dari bilangan tertentu.
4. Pembuktian dapat diberikan daripada solusi optimum yang telah didapatkan.

Selama aturan-aturan berhenti ini memungkinkan untuk memiliki beberapa pengaruh dalam prosedur pencarian dan pada hasil-hasilnya, penting untuk menyadari bahwa pendefinisian $N(i,k)$ dalam tiap pengulangan k dan pemilihan

*V** adalah hal yang krusial. Definisi dari *N(i,k)* menyatakan secara tidak langsung bahwa beberapa solusi yang telah dikunjungi dihapus dari *N(i),* mereka dianggap sebagai solusi-solusi tabu yang harus dihindari dalam pengulangan selanjutnya. Sebagai contoh, pemeliharaan pada pengulangan *k* sebuah list *T*(*list tabu*) pada solusi yang telah dikunjungi terakhir |*T*| akan mencegah terjadinya siklus pada ukuran paling banyak sebesar |T|. Pada kasus ini, kita bisa mengambil *N(i,k)=N(i)-T*. Namun list *T* kemungkinan tidak dapat digunakan secara praktis. Oleh karena itu, kita akan mendeskripsikan proses eksplorasi pada *S* dalam masa pergerakan dari satu solusi ke solusi lainnya. Untuk setiap solusi *I* dalam *S,* kita dapat mendefinisikan *M(i)* sebagai himpunan gerak yang dapat digunakan oleh *i* untuk memperoleh solusi baru *j* (notasi: $j=i \nabla m$). Secara umum kita dapat menggunakan gerakan-gerakan yang dapat dibalik. Untuk setiap m terdapat gerakan m^{-1} sehingga $(i \nabla m) \nabla m^{-1} = i$.

Jadi daripada mempertahankan list *T* dari solusi-solusi yang telah dikunjungi, kita dapat secara sederhana memelihara jalur gerak terakhir |*T*| atau gerak balik terakhir |*T*| yang diasosiasikan dfengan gerakan-gerakan yang sebenarnya telah dilakukan. Maka dengan jelas bahwa pembatasan yang ada adalah kehilangan informasi, dan hal itu tidak menjamin tidak terjadinya siklus dengan panjang paling banyak |*T*|. Untuk kepentingan efisiensi, maka diperlukan penggunaan beberapa list *Tγ* dalam satu waktu maka beberapa unsur pokok *tγ* (dari *i* atau dari *m*) akan diberikan tabu status untuk mengindikasi bahwa unsur pokok ini sedang tidak diperbolehkan untuk terlibat dalam pergerakan. Secara umum pergerakan untuk tabu status adalah fungsi tabu status pada unsur-unsur pokoknya yang dapat diubah pada setiap pengulangan. Gerak tabu m digunakan

pada solusi *i* yang mungkin tampak menarik karena itu diberikan sebagai contoh sebuah solusi yang lebih baik dari pada yang telah ditemukan. Kita akan dapat menerima *m* tanpa memperhatikan statusnya. Kita akan melakukan hal tersebut jika *m* memiliki tingkat aspirasi (*aspiration level*) *a(i,m)* yang lebih baik daripada permulaan *A(i,m)*.

Sekarang kita dapat mendefinisikan karakteristik dari prosedur pencarian tabu dalam langkah-langkah berikut, antara lain:

1. Memilih solusi awal i dalam S. Tetapkan i*=I dan k=0.
2. Tetapkan k=k+1 dan bangkitkan sebuah subset V* dari solusi dalam N(I,k) sehingga salah satu dari kondisi tabu tγ yang melanggar (γ=1,2,…,t) atau setidaknya satu kondisi aspirasi aγ yang memiliki (γ=1,2,…,a).
3. Pilih j terbaik melalui j=i∇m dalam V* dan tetapkan i=j.
4. Jika f(i)<f(i*) maka tetapkan i*=i.
5. Perbaharui kondisi Tabu dan aspirasi.
6. Jika kondisi berhenti ditemukan, maka proses dihentikan. Jika tidak, kembali ke langkah 2.

Hal-hal di atas dapat dikatakan sebagai bahan-bahan dasar dari metode pencarian tabu yang nantinya akan digunakan untuk memecahkan masalah penjadwalan pekerjaan (job scheduling) pada pembahasan selanjutnya.

2.2 Graf

2.2.1 Definisi Graf

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. Graf digunakan untuk mempresentasikan obkel-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek sebagai noktah, bulatan atau titik, sedangkan hubungan antara objek dinyatakan dengan garis.

Secara metematis, graf didefinisikan sebagai berikut:

DEFINISI 1. Graf *G* didefinisikan sebgai pasangan himpunan (V, E), yang dalam hal ini:

V = himpunan tidak kosong dari simpul-simpul (*vertices* atau *node*)

$= \{ v_1 , v_2 , ... , v_n\}$

dan

E = himpunan sisi (*edges* atau *arc*) yang menghubungkan sepasang simpul

$=\{e_1 , e_2 , ... , e_n \}$

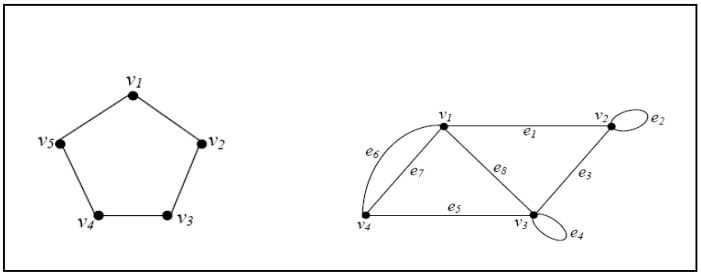
atau dapat ditulis singkat notasi *G = (V, E)*.

Definisi 1 menyatakan bahwa *V* tidak boleh kosong, sedangkan *E* boleh kosong. Kadi, sebuah graf dimungkinkan tidak mempunyai sisi satu buah pun, tetapi simpulnya harus ada, minimal satu. Graf yang hanya mempunyai satu buah simpul tanpa sebuah sisi pun dinamakan **graf trivial**.

Simpul pada graf dapat dinomori dengan huruf, seperti *a, b , c, ...v, w,....* dengan bilangan asli 1, 2, 3, ... atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul v_i dengan simpul v_j dinyatakan dengan pasangan (v_i, v_j) atau dengan lambing e_1, e_2, ... Dengan kata lain, jika *e* adalah sisi yang

menghubungkan simpul v_i dengan simpul v_j, maka e dapat ditulis sebagai $e = (v_i, v_j)$.

Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi).



Gambar 2.1 Graf Sederhana dan Graf Ganda

2.2.2 Jenis-jenis Graf

Graf dapat dikelompokan menjadi beberapa kategori (jenis) bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

1. **Graf sederhana** (*simple graph*).

   Graf yang tidak mengandung gelang maupun sisi-ganda dinmakan graf sederhana. G_1 pada gambar 2.1(a) adalah contoh graf sederhana yang mempresentasikan jaringan komputer. Simpul menyatakan komputer, sedangkan sisi menyatakan saluran telepon untuk berkomunikasi. Saluran telepon dapat beroprasi pada sua arah.

2. Graf tak-sederhana (*unsimple – graph*).

Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu **graf ganda** (*multigraph*) dan **graf semu** (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda. Sisi ganda yang menghubungkan sepasang simpul bias lebih dari dua buah. G_2 pada gambar 2.1(b) adalah graf ganda. Sisi ganda pada G_2 dapat diandaikan sebagai saluran telepon tambahan apabila beban komunikasi data antar komputer sangat padat. Graf semu adalah graf yang mengandung gelang. G_3 adalah graf semu (termasuk bila memiliki sisi ganda sekalipun). Sisi gelang pada G_3 dapat dianggap sebagai saluran telepon tambahan yang menghubungakan komputer dengan dirinya sendiri (mungkin untuk tujuan diagnostic). Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri.

Jumlah simpul pada graf disebut sebagai kardinalitas graf, dan sinyatakan dengan $n = |V|$, dan jumlah sisi dinyatakan dengan $m = |E|$. Berdasarkan jumlah simpul pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

2.2.3 Model Matematis Graf Sebagai Representasi *Hard Constraint*

Constrain adalah keadaan yang didefinisan agar penjadwalan yang dihasilkan merupakan jadwal yang baik dan tidak boleh dilanggar. Untuk mempresentasikan *constrain* yang ada dibangunlah sebuah model matematis dalam bentuk graf.

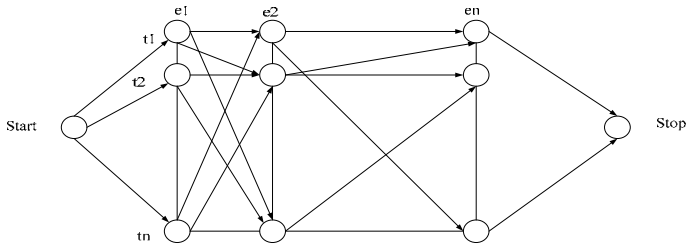
Graf G didefiniskan sebagai pasangan himpunan (V,E) dimana:

V= himpunan berhingga yang tidak kosong dari simpul-simpul atau vertex/ node $\{V_1,V_2, \ldots\ldots,V_n\}$.

E= himpunan sisi (edge atau arc yang menghubungkan sepasang simpul $\{e_1, e_2,\ldots\ldots, e_n\}$.

Untuk masalah penjadwalan mata pelajaran yang akan diselesaikan menggunakan algoritma *Tabu Search*, graf yang dibangun merupakan graf berarah. Node-node yang terbentuk nantinya merupakan representasi dari *subject-subject* yang akan sijadwalkan pada waktu t, *subject-subject* tersebut merupakan sekumpulan data yang terdiri dari kelas, dan mata pelajaran yang diajarkan.

Untuk masalah penjadwalan mata pelajaran yang akan diselesaikan dengan menggunakan algoritma *Tabu Search* dibuat graf berarah. Node-node yang terbentuk merupakan representasi dari beberapa *subject* yang akan dijadwalkan pada *timeslot* t, *subject-subject* tersebut merupakan sekumpulam *record* yang terdiri dari kelas, guru, dan mata pelajaran yang diajarkan. *Timeslot* adalah kumpulan waktu yang memungkinkan untuk dijadwalkan pada *subject-subject* tadi. Sedangkan keterhubungan antar node-node yang terbentuk (edge) merupakan representasi dari *constrain* yang ada, sebagai contoh dapat dilihat pada gambar di bawah ini:



Gambar 2.2 Graf sebagai Representasi *Constraint*

Maksud dari gambar di atas adalah mempresentasikan penelusuran node yang mencari pasangan antara *subject* dan *timeslot* yang memebuhi *constrain*. Pada setiap *subject* terdapat nilai *constrain* yaitu kelas, dan guru. *Subject* pertama sudah mendapatkan *timeslot* yang berimpit atau beririsan maka pemeriksaan dilakukan terhadap nilai *constrain* dari *subject* yaitu kelas dan guru. Jika salah satunya sama dengan *subject* sebelumnya maka *timeslot* tidak bisa terhubung dan harus mencari lagi *timelot* lain yang terbebas dari nilai *constrain* tadi.

2.3 Konsep Basis Data

Hampir disemua aspek pemanfaatan perangkat komputer dalam sebuah organisasi atau perusahaan senantiasa berhubungan dengan basis data. Perangkat komputer dalam suatu organisasi atau perusahaan biasanya digunakan untuk menjalankan fungsi pengolahan system informasi, yang dewasa ini sudah menjadi suatu keharusan demi untuk meningkatkan efisiensi, daya saing, dan kecepatan operasional perusahaan.

2.3.1 Pengertian Basis Data

Basis data atau *database* adalah kumpulan dari data-data yang berhubungan antara satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Basis data merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis dalam menyediakan informsi bagi para pemakai.

Tujuan dari desain basis data adalah untuk menentukan data-data yang dibutuhkan dalam sistem, sehingga informasi yang dihasilkan dapat terpenuhi dengan baik. Perancangan *database* yang digunakan adalah untuk memudahkan

dalam mengetahui *file-file database* yang digunakan dalam perancangan sistem, sekaligus untuk mengetahui hubungan antara file dari *database* tersebut.

Ada beberapa istilah atau definisi yang digunakan dalam sistem manajemen basis data, antara lain :

a. Entitas

Orang, tempat, kejadian (konsep) yang informasinya direkam.

b. Atribut

Setiap entitas mempunyai atribut atau sebutan untuk mewakili suatu entitas.

c. Nilai / isi data

Data aktual informasi yang disimpan pada tiap data.

d. *Record / Tuple*

Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu atau seseorang.

e. *File*

Kumpulan *file-file* yang mempunyai panjang elemen yang sama, atribut sama, namun berbeda isi datanya.

f. *Database*

Kumpulan *file-file* yang mempunyai kaitan antara *file-file* dengan *file* lainnya sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan atau instansi dalam batasan tertentu.

g. *Database Management System* (DBMS)

Kumpulan *file* yang saling berkaitan bersama dengan program untuk pengelolaanya.

2.3.2 Normalisasi

Normalisasi merupakan proses yang menggunakan pendekatan formal untuk menelaah dan kemudian mengelompokan data *item / field / atribut* ke bentuk yang lebih baik dalam menghadapi perubahan-perubahan bisnis dimasa mendatang serta meminimumkan pengaruh perubahan pada sistem aplikasi atau program. Berikut ini merupakan bentuk-bentuk normalisasi:

1. Bentuk normal pertama (1NF)

   Suatu tabel dapat disebut bentuk normal pertama jika semua atributnya memiliki nilai yang atomik (atribut yang bersangkutan tidak dapat dibagi lagi menjadi atribut-atribut yang lebih kecil) tetapi masih mengandung *redundancy* (atribut yang tampil berulang-ulang).

2. Bentuk Normal Kedua (2NF)

   Suatu tabel bentuk normal pertama yang memenuhi syarat tambahan bahwa semua atribut bukan kuncinya hanya bergantung pada kunci primer.

3. Bentuk Normal Ketiga (3NF)

   Suatu tabel bentuk normal kedua yang memenuhi syarat tambahan bahwa semua atribut bukan tidak memiliki ketergantungan transitif terhadap kunci primer.

4. Bentuk Normal Keempat (BCNF)

   Suatu tabel yang memiliki semua *field* penentu yang merupakan *candidate key*. Bentuk merupakan perbaikan bentuk normal ketiga.

2.4 Perankat Lunak Penunjang

2.4.1 Visual Basic (VB 6.0)

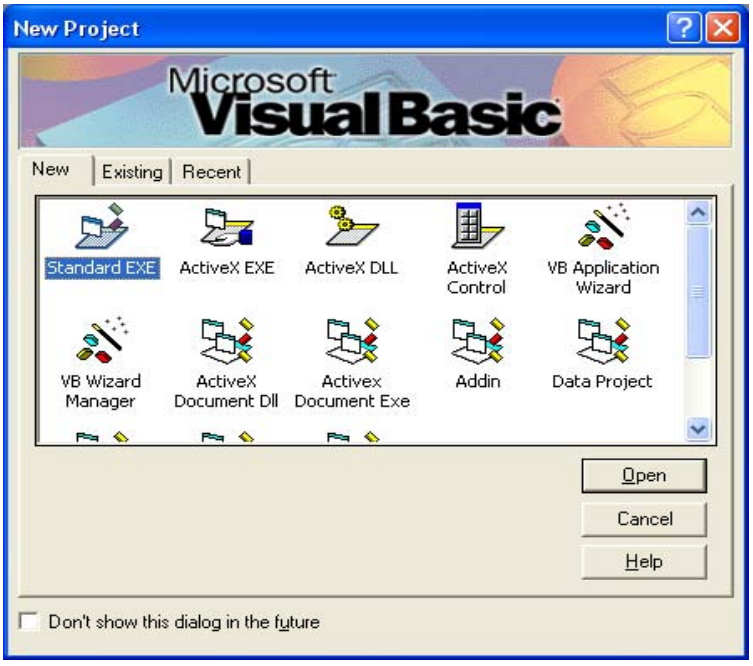
Visual basic merupakan bahasa pemograman yamg tercepat dan termudah untuk membuat satu aplikasi dalam microsoft windows. Dengan menggunakan metode *Graphical User Interface (GUI).* Visual basic memudahkan pemograman untuk berinteraksi langsung dengan elemen-elemen untuk setiap bentuk pemograman. Visual basic dibuat sebagai langkah pengembangan untuk menyesuaikan BASIC *(Beginners All-Purpose Symbolic Intruction Code*) yang berbasis DOS yang tidak mempunyai kemampuan menggunakan metode GUI dalam basis windows.

Sebagai programs yang berbasis windows, Visual Basic mempunyai kemampuan untuk berinteraksi dengan seluruh aplikasi Windows seperti Microsoft Word, Microsoft Excel, Microsoft Acces dan sebagainya. Dengan kemampuan yang hampir tidak terbatas, Visual basic dapat digunakan untuk semau jenis aplikasi pemograman. Anda dapat membuat aplikasi windows swperti word atau excel ataupun game, multimedia, program penghitungan dan

sebagainya. Visual basic dapat digunakan untuk membuat objek-objek pembantu program seperti misalnya Control Activex, file help, aplikasi internet dan sebagainya. Selain itu visual basic juga dapat digunakan untuk menguji program (debugging) dan menghasilkan program akhir berakhiran exe yang bersifat executable atau dapat langsung digunakan.

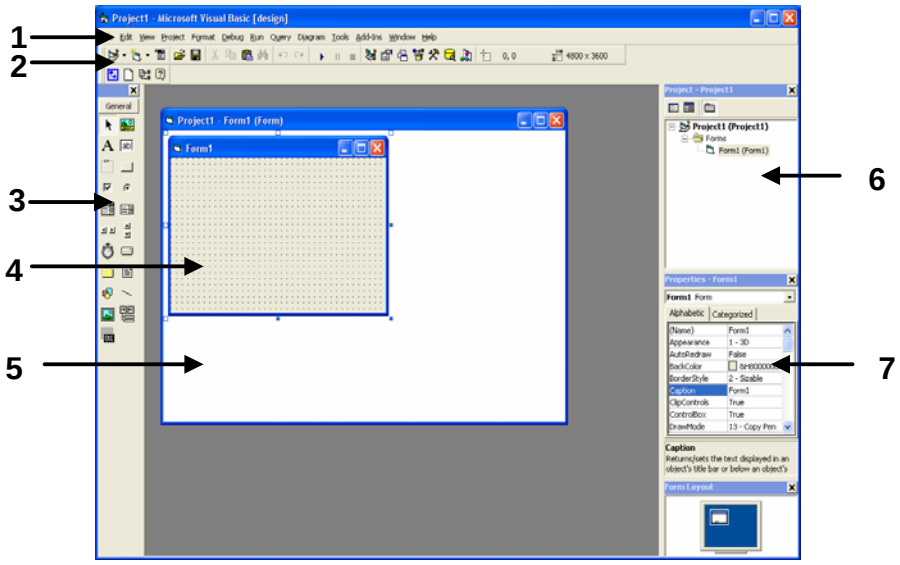
2.4.1.1 Integrated Depelopment Environment (IDE) VB 6.0

Merupakan lingkungan pengembangan dari visual basic 6 yang terintegrasi. Ketika pertama kali visual basic diaktifkan maka akan muncul tampilan seperti pada gambar dibawah ini :



Gambar 2.3 Tampilan Awal *Visual Basic*

Setelah kita memilih layar new project standard exe, selanjutnya akan tampil layar kerja atau area kerja dari visual basic, seperti gambar dibawah ini:



Gambar 2.4 Tampilan *New Project*

Keterangan :

1. Menubar

   Menubar berfungsi untuk memilih tugas-tugas tertentu, seprti memulai, membuka dan menyimpan project, mengompilasi project menjadi file executable (exe) dan lain-lain.

2. Toolbar

   Toolbar memiliki fungsi yang sama seperti menubar, dan juga berfungsi seperti jalan pintas karena lebih praktis dalam penggunaanya.

3. Toolbox

   Toolbox berisi komponen-komponen yang merupakan sarana untuk membentuk user interface.

4. Jendela Form Designer

   Form designer merupakan tempat untuk merancang user interface, dimana kontrol-kontrol yang dibutuhkan bisa diletakkan.

5. Jendela Kode

   Jendela kode merupakan tempat untuk menuliskan kode-kode bagi objek yang dibuat. Melalui jendela ini dapat mengatur kerakteristik dari form yang dibuat.

6. Jendela Project

   Jendela project adalah jendela yang menampilkan semua file yang berhubungan denagn apliaksi atau project yang saat itu sedang dijalankan.

7. Jendela Properties

   Jendela properties ini berisi daftar property untuk objek (form atau control) yang dipilih dan berfungsi untuk mengatur karakteristik, seperti warna, ukuran dan lain-lain.

Banyak sekali toolo-tool dan jendela-jendela pendukung lainnya yang ada di lingkungan visual basic 6 sebagai alat untuk mengembangkan aplikasi yang akan dibuat. Seperti jendela form layout yang menunjukkan tampilan form pada saat dijalankan.

Ada juga jendela modul yang merupakan tempat untuk menuliskan kode-kode yang lebih bersifat global. Pada jendela modul ini, bisa mendeklarasikan variabel dan fungsi dengan awalan private maupun public. Jika deberi awalan private, variabel atau fungsi tersebut hanya berlaku untuk pemakaian modul yang bersangkutan. Sedangkan awalan public akan mengakibatkan variabel atau fungsi tersebut bersifat global. Jendela modul ini bisa ditampilakan denagn mengklik menu project pada menubar kemudian pilih add module.

2.4.2 Microsoft Access

Microsoft Access (atau Microsoft Office Access) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi Microsoft Office, selain tentunya Microsoft Word, Microsoft Excel, dan Microsoft PowerPoint. Aplikasi ini menggunakan mesin basis data Microsoft Jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. Versi terakhir adalah Microsoft Office Access 2007 yang termasuk ke dalam Microsoft Office System 2007.

Microsoft Access dapat menggunakan data yang disimpan di dalam format Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/*programmer* yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. Access juga mendukung teknik-teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

2.4.2.1 Sejarah

Microsoft merilis Microsoft Access 1.0 pada bulan November 1992 dan dilanjutkan dengan merilis versi 2.0 pada tahun 1993. Microsoft menentukan spesifikasi minimum untuk menjalankan Microsoft Access 2.0 adalah sebuah

komputer dengan sistem operasi Microsoft Windows 3.0, RAM berkapasitas 4 *megabyte* (6 megabyte lebih disarankan) dan ruangan kosong hard disk yang dibutuhkan 8 *megabyte* (14 *megabyte* lebih disarankan). Versi 2.0 dari Microsoft Access ini datang dengan tujuh buah disket *floppy* 3½ inci berukuran 1.44 *megabyte*.

Perangkat lunak tersebut bekerja dengan sangat baik pada sebuah basis data dengan banyak *record* tapi terdapat beberapa kasus di mana *data* mengalami kerusakan. Sebagai contoh, pada ukuran basis data melebihi 700 *megabyte* sering mengalami masalah seperti ini (pada saat itu, memang *hard disk* yang beredar masih berada di bawah 700 *megabyte*). Buku manual yang dibawanya memperingatkan bahwa beberapa kasus tersebut disebabkan oleh *driver* perangkat yang kuno atau konfigurasi yang tidak benar.

Nama kode (*codename*) yang digunakan oleh Access pertama kali adalah **Cirrus** yang dikembangkan sebelum Microsoft mengembangkan Microsoft Visual Basic, sementara mesin pembuat *form* antarmuka yang digunakannya dinamakan dengan **Ruby**. Bill Gates melihat purwarupa (*prototype*) tersebut dan memutuskan bahwa komponen bahasa pemrograman BASIC harus dikembangkan secara bersama-sama sebagai sebuah aplikasi terpisah tapi dapat diperluas. Proyek ini dinamakan dengan **Thunder**. Kedua proyek tersebut dikembangkan secara terpisah, dan mesin pembuat *form* yang digunakan oleh keduanya tidak saling cocok satu sama lainnya. Hal tersebut berakhir saat Microsoft merilis Visual Basic for Applications (VBA).

2.4.2.2 Penggunaan

Microsoft Access digunakan kebanyakan oleh bisnis-bisnis kecil dan menengah, di dalam sebuah organisasi yang kecil bahkan mungkin juga digunakan oleh perusahaan yang cukup besar, dan juga para programmer untuk membuat sebuah sistem buatan sendiri untuk menangani pembuatan dan manipulasi data. Access juga dapat digunakan sebagai sebuah basis data untuk aplikasi Web dasar yang disimpan di dalam *server* yang menjalankan Microsoft Internet Information Services (IIS) dan menggunakan Microsoft Active Server Pages (ASP). Meskipun demikian, penggunaan Access kurang disarankan, mengingat telah ada Microsoft SQL Server yang memiliki kemampuan yang lebih tinggi.

Beberapa pengembang aplikasi profesional menggunakan Microsoft Access untuk mengembangkan aplikasi secara cepat (digunakan sebagai *Rapid Application Development*/*RAD tool*), khususnya untuk pembuatan purwarupa untuk sebuah program yang lebih besar dan aplikasi yang berdiri sendiri untuk para *salesman*.

Microsoft Access kurang begitu bagus jika diakses melalui jaringan sehingga aplikasi-aplikasi yang digunakan oleh banyak pengguna cenderung menggunakan solusi sistem manajemen basis data yang bersifat klien/server. Meskipun demikian, tampilan muka Access (*form*, *report*, *query*, dan kode Visual Basic) yang dimilikinya dapat digunakan untuk menangani basis data yang sebenarnya diproses oleh sistem manajemen basis data lainnya, seperti halnya Microsoft Jet Database Engine (yang secara *default* digunakan oleh Microsoft

Access), Microsoft SQL Server, Oracle Database, dan beberapa produk lainnya yang mendukung ODBC.

2.4.2.3 Fitur

Salah satu keunggulan Microsoft Access dilihat dari perspektif *programmer* adalah kompatibilitasnya dengan bahasa pemrograman Structured Query Language (SQL); query dapat dilihat dan disunting sebagai statemen-statemen SQL, dan statemen SQL dapat digunakan secara langsung di dalam Macro dan VBA Module untuk secara langsung memanipulasi tabel data dalam Access. Para pengguna dapat mencampurkan dan menggunakan kedua jenis bahasa tersebut (VBA dan Macro) untuk memprogram *form* dan logika dan juga untuk mengaplikasikan konsep berorientasi objek.

Microsoft SQL Server Desktop Engine (MSDE) 2000, yang merupakan sebuah versi mini dari Microsoft SQL Server 2000, dimasukkan ke dalam Office XP Developer Edition dan dapat digunakan oleh Microsoft Access sebagai alternatif dari Microsoft Jet Database Engine.

Tidak seperti sebuah sistem manajemen basis data relasional yang komplit, Microsoft JET Database Engine tidak memiliki fitur *trigger* dan *stored procedure*. Dimulai dari Microsoft Access 2000 yang menggunakan Microsoft Jet Database Engine versi 4.0, ada sebuah sintaksis yang mengizinkan pembuatan kueri dengan beberapa parameter, dengan sebuah cara seperi halnya sebuah stored procedure, meskipun prosesur tersebut dibatasi hanya untuk sebuah pernyataan tiap prosedurnya. Access juga mengizinkan *form* untuk mengandung kode yang dapat

dieksekusi ketika terjadi sebuah perubahan terhadap tabel basis data, seperti halnya *trigger*, selama modifikasi dilakukan hanya dengan menggunakan form tersebut, dan merupakan sesuatu hal yang umum untuk menggunakan kueri yang akan diteruskan (*pass-through* dan teknik lainnya di dalam Access untuk menjalankan *stored procedure* di dalam RDBMS yang mendukungnya.

Dalam berkas Access Database Project (ADP) yang didukung oleh Microsoft Access 2000 dan yang selanjutnya, fitur-fitur yang berkaitan dengan basis data berbeda dari versi format/struktur data yang digunakan Access (*.MDB), karena jenis berkas ini dapat membuat koneksi ke sebuah basis data MSDE atau Microsoft SQL Server, ketimbang menggunakan Microsoft JET Database Engine. Sehingga, dengan menggunakan ADP, adalah mungkin untuk membuat hampur semua objek di dalam server yang menjalankan mesin basis data tersebut (tabel basis data dengan *constraints* dan *trigger*, *view*, *stored procedure*, dan *UDF*). Meskipun demikian, yang disimpan di dalam berkas ADP hanyalah *form*, *report*, *macro*, dan modul, sementara untuk tabel dan objek lainnya disimpan di dalam server basis data yang membelakangi program tersebut.

2.4.2.4 Pengembangan dengan Access

Access mengizinkan pengembangan yang relatif cepat karena semua tabel basis data, kueri, form, dan report disimpan di dalam berkas basis data miliknya (`*.MDB`). Untuk membuat Query, Access menggunakan Query Design Grid, sebuah program berbasis grafis yang mengizinkan para penggunanya untuk membuat query tanpa harus mengetahui bahasa pemrograman SQL. DI dalam Query Design Grid, para pengguna dapat memperlihatkan tabel basis data sumber

dari *query*, dan memilih *field-field* mana yang hendak dikembalikan oleh proses dengan mengklik dan menyeretnya ke dalam *grid*. *Join* juga dapat dibuat dengan cara mengklik dan menyeret field-field dalam tabel ke dalam *field* dalam tabel lainnya. Access juga mengizinkan pengguna untuk melihat dan memanipulasi kode SQL jika memang diperlukan.

Bahasa pemrograman yang tersedia di dalam Access adalah Microsoft Visual Basic for Applications (VBA), seperti halnya dalam beberapa aplikasi Microsoft Office. Dua buah pustaka komponen Component Object Model (COM) untuk mengakses basis data pun disediakan, yakni Data Access Object (DAO), yang hanya terdapat di dalam Access 97, dan ActiveX Data Objects (ADO) yang tersedia dalam versi-versi Access terbaru.