BAB II
LANDASAN TEORI

2.1 Data mining

2.1.1 Pengertian Data mining

Data mining adalah kegiatan yang meliputi pengumpulan, pemakaian data historis yang menemukan keteraturan, pola dan hubungan dalam set data berukuran besar [6]. Maksud dari pengertian ini yaitu proses pecarian informasi yang tidak diketahui sebelumnya dari sekumpulan data besar.

Karakteristik *Data mining* sebagai berikut:

a. *Data mining* berhubungan dengan penemuan sesuatu yang tersembunyi dan pola data tertentu yang tidak diketahui sebelumnya.
b. *Data mining* biasa menggunakan data yang sangat besar. Biasanya data yang besar digunakan untuk membuat hasil lebih dipercaya.
c. *Data mining* berguna untuk membuat keputusan yang kritis, terutama dalam strategi[10].

2.1.2 Pengelompokan Data mining

Secara umum ada dua jenis metode pada *Data mining*, yaitu:

a. Metode Prediktive

Proses untuk menemukan pola dari data yang menggunakan beberapa variabel untuk memprediksi variabel lain yang tidak diketahui jenis atau nilainya. Teknik yang termasuk dalam *predikative mining* antara lain Klasifikasi, *Regresi*, dan *Deviasi.*

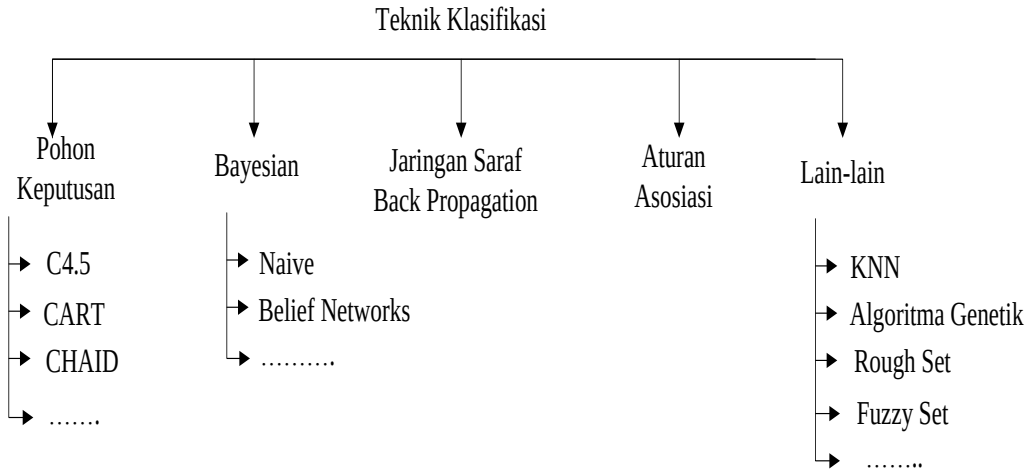
b. Metode Descriptive

Proses untuk menemukan suatu karakteristik penting dari data dalam suatu basis data. Teknik *Data mining* yang termasuk dalam *descriptive mining* adalah *Clustering, Association,* dan *Secuential Mining.*

2.2 Klasifikasi

Klasifikasi data adalah suatu proses yang menemukan properti-properti yang sama pada sebuah himpunan obyek di dalam sebuah basis data, dan mengklasifikasikannya ke dalam kelas-kelas yang berbeda menurut model Klasifikasi yang ditetapkan. Tujuan dari klasifikasi adalah untuk menemukan model dari *training set* yang membedakan atribut ke dalam kategori atau kelas yang sesuai, model tersebut kemudian digunakan untuk mengklasifikasikan atribut yang kelasnya belum diketahui sebelumnya.

Teknik klasifikasi terbagi menjadi beberapa teknik yang diantaranya sebagai berikut:



Gambar II-1Teknik Klasifikasi

2.3 Pohon Keputusan (*Decision tree*)

2.3.1 Pengertian Pohon Keputusan

Decision tree adalah flow-chart seperti struktur tree, dimana tiap internal node menunjukkan sebuah test pada sebuah atribut, tiap cabang menunjukkan hasil dari test, dan leaf node menunjukkan class-class atau class distribution.[7]

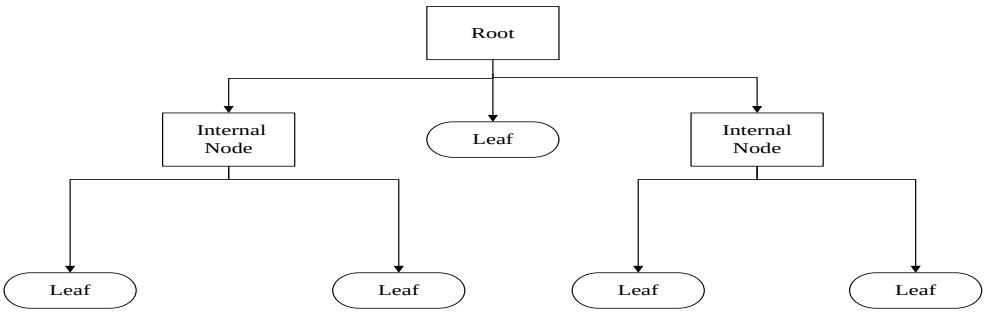
2.3.2 Model Pohon Keputusan

Pohon keputusan adalah model prediksi menggunakan struktur pohon atau struktur berhirarki.

Decision tree merupakan metode klasifikasi yang paling popular digunakan. Selain karena pembangunannya relatif cepat, hasil dari model yang dibangun mudah untuk dipahami. Pada *decision tree* terdapat 3 jenis *node*, yaitu:

a. *Root Node*, merupakan *node* paling atas, pada *node* ini tidak ada *input* dan bisa tidak mempunyai *output* atau mempunyai *output* lebih dari satu.
b. *Internal Node* , merupakan *node* percabangan, pada *node* ini hanya terdapat satu *input* dan mempunyai *output* minimal dua.
c. *Leaf node atau terminal node* , merupakan *node* akhir, pada *node* ini hanya terdapat satu *input* dan tidak mempunyai *output*.

Contoh dari pohon keputusan dapat dilihat di Gambar II.2 berikut ini.



Gambar II-2 Model Pohon Keputusan

Decision tree adalah flow-chart seperti struktur tree, dimana tiap internal node menunjukkan sebuah test pada sebuah atribut, tiap cabang menunjukkan hasil dari test, dan leaf node menunjukkan class-class atau class distribution [7].

2.4 Algoritma *C4.5*

2.4.1 Pengenalan Algoritma C4.5

Algoritma C4.5 merupakan kelompok algoritma decision tree. Algoritma ini mempunyai input berupa *training samples* dan *samples*. *Training samples* berupa data contoh yang akan digunakan untuk membangun sebuah *tree* yang telah diuji kebenarannya. Sedangkan *samples* merupakan *field-field* data yang nantinya akan kita gunakan sebagai parameter dalam melakukan klasifikasi data[7].

Algoritma *C4.5* adalah algoritma hasil pengembangan dari algoritma ID3. Perbaikan algoritma C4.5 dari algoritma ID3 dilakukan dalam hal(Santosa,ID3):

1. Bisa mengatasi *missing value*
2. Bisa mengatasi data kontinyu
3. *Pruning*
4. Adanya aturan

Secara garis beras langkah-langkah yang dilakukan oleh algoritma C4.5 dalam membentuk pohon keputusan adalah sebagai berikut:

1. Pada saat awal pembentukan pohon akan dimulai dengan membuat suatu node yang melambangkan *training* sample.
2. Jika sampel-sampel tersebut memiliki kelas yang sama, maka node tersebut dijadikan *leaf node* dengan label kelas tersebut.
3. Jika sampel-sampel tersebut tidak memiliki kelas yang sama maka algoritma ini akan mencari *gain ratio* tertinggi dari atribut yang tersedia, sebagai cara untuk memilih atribut yang paling berpengaruh paada training sample yang tersedia. Nantinya atribut ini akan dijadikan atribut “penguji”

atau “keputusan” pada node tersebut. Hal yang perlu diperhatikan adalah ketika atribut tersebut bernilai continu, maka atribut tersebut harus di diskritkan terlebih dahulu.

4. Cabang untuk setiap *node* akan dibentuk berdasarkan nilai-nilai yang diketahui dari atribut pengujian.
5. Algoritma ini akan terus melakukan proses yang sama rekursif untuk membentuk suatu pohon keputusan untuk setiap sample di setiap bagiannya.
6. Proses rekursif ini akan berhenti, ketika salah satu dari kondisi di bawah telah terpenuhi. Kondisi-kondisinya adalah:

a. Semua sample yang diberikan pada *node* adalah berasal dari satu kelas yang sama.
b. Tidak ada atribut lainnya yang dapat digunakan untuk mempartisi sample lebih lanjut.
c. Tidak ada sample yang memenuhi *test-attribute* = a_i. Dalam hal ini, sebuah daun dibuat dan dilabeli dengan kelas yang memiliki sample terbanyak (*majority voting*).

Algoritma untuk melakukan kontruksi pohon keputusan dapat dilihat pada gambar 2.3.1.

Table II-1 Algoritma Membuat Pohon Keputusan

Algorithm : Generete_decision_tree.
Narrative : Generate a decision tree from the given training data.
Input : The training samples, samples, represented by discrete-valued attribute, the set of candidate attributes, attribute-list.
Output : A dicision tree.
 	
Method
1) Craete a node *N*;
2) If *samples* are all of the same class, C then
3) Return N as a leaf node labeled with class C
4) If attribute-list is empty then
5) Return N as leaf node labeled with the most common class in samples; // majority voting
6) Selsct test-attribute, the attribute among attribute-list with the

highest gain ratio;
7) Label node N with last attribute;
8) For each known value a, of test-attribute
9) Grow a branch from node N for the condition test-attribute = a_i
10) Let S_i be the set of samples in samples for wichh test-attribute = a_i // a partition
11) If S_i is empty then
12) attach a leaf labeled with the most common class in *sample*;
13) Else attach the node returned by Generate_decision_tree (S_i, attribute-list-list-attribute);

Pada tahap pembelajaran algoritma C4.5 memiliki 2 prinsip kerja yaitu:

1. Pembuatan pohon keputusan. Tujuan dari algoritma penginduksi pohon keputusan adalah mengkontruksi struktur data pohon yang dapat digunakan untuk memprediksi kelas dari sebuah kasus atau *record* baru yang belum memiliki kelas. C4.5 melakukan konstruksi pohon keputusan dengan metode *divide and conquer*. Pada awalnya hanya dibuat *node* akar dengan menerapkan algoritma *divide and conquer*. Algoritma ini memilih pemecahan kasus-kasus yang terbaik dengan menghitung dan membandingkan *gain ratio*, kemudian *node-node* yang terbentuk di level berikutnya, algoritma *divide and conquer* akan diterapkan lagi sampai terbentuk daun-daun.
2. Pembuatan aturan-aturan (*rule set*). Aturan-aturan yang terbentuk dari pohon keputusan akan membentuk suatu kondisi dalam bentuk *if-then*. Aturan-aturan ini didapat dengan cara menelusuri pohon keputusan dari akar sampai daun. Setiap *node* dan syarat percabangan akan membentuk suatu kondisi atau suatu *if*, sedangkan untuk nilai-nilai yang terdapat pada daun akan membentuk suatu hasil atau suatu *then*.

2.4.2 Information Gain

Information gain adalah salah satu *attribute selection measure* yang digunakan untuk memilih *test attribute* tiap *node* pada *tree*. Atribut dengan informasi gain tertinggi dipilih sebagai test atribut dari suatu node [7].

Dalam prosesnya perhitungan gain bisa terjadi *missing value* dan juga bisa tidak terjadi *missing value*. Rumus yang digunakannya juga berbeda.

2.4.2.1 Perhitungan gain tanpa missing value[7].

Misalkan S berisi s data samples. Anggap atribut untuk class memiliki m nilai yang berbeda, C_i (untuk i=1,…,I). Anggap S_i menjadi jumlah samples S pada class C_i. Maka besar informasinya dapat dihitung dengan:

$$\mathrm{I}\left(S_{1,}S_{2,}\ldots\ldots S_m\right) = -\sum_{j=1}^{m} \mathrm{P_i} * \mathrm{Log_2}\,(\mathrm{P_i}) \qquad (1)$$

dimana $P_i = \frac{S_i}{S}$ adalah probabilitas dari *sample* yang mempunyai class C_i.

Misalkan atribut A mempunyai v nilai yang berbeda, $\{a_1,\ a_2,\ldots a_v\}$. Atribut A dapat digunakan untuk mempartisi S menjadi v subset, $\{S_1,\ S_2,\ldots S_v\}$, dimana S_j berisi samples pada S yang mempunyai nilai a_j dari A. jika A terpilih menjadi test atribut (yaitu, *best* atribut untuk *splitting*), maka subset-subset akan berhubungan dengan pertumbuhan node-node cabang yang berisi S. Anggap S_{ij} sebagai jumlah samples class $C_i$pada subset S_j. Entropy, atau nilai *information* dari subset A adalah :

$$\mathrm{E}(A) = \sum_{f=1}^{y} \frac{\mathrm{S_{1j}+\cdots+S_{mj}}}{\mathrm{S}}\ 1\ (\ S_1,\ S_2,\ldots S_m\ ) \qquad (2)$$

$\frac{\mathrm{S_{1j}+\cdots+S_{mj}}}{\mathrm{S}}$ adalah bobot dari subset *j*th dan jumlah samples pada subset (yang mempunyai nilai a_j dari A) dibagi dengan jumlah total samples pada S. Untuk subset S_j,

$$\mathrm{I}\ (\ S_{1j},\ S_{2j},\ldots S_{mj}\ ) = -\sum_{j=1}^{m} \mathrm{P_{ij}} * \mathrm{Log_2}\,(\mathrm{P_{ij}})$$

Dimana $\mathrm{P_{ij}} = \frac{S_{ij}}{|S_j|}$ adalah probabilitas sample $\mathrm{S_j}$ yang mempunyai kelas Ci.

Maka nilai information gain atribut A pada subset S adalah

$$\mathrm{Gain\ (A)} = \mathrm{I}\ (\ S_1,\ S_2,\ldots S_m\ ) - \mathrm{E(A)} \qquad (3)$$

2.4.2.2 Perhitungan information gain dengan missing value[7].

Untuk atribut dengan *missing value* perhitungan information gain-nya diselesaikan dengan *gain ratio*. Sebelum menghitung *gain ratio* terlebih dahulu dihitung I ($S_1, S_2,\ldots S_m$) dan E(A)

$$I (S_1, S_2,\ldots S_m) = -\sum_{j=1}^{m} P_i * \text{Log}_2 (P_i) \quad (4)$$

$$E(A) = \sum_{f=1}^{y} \frac{S_{1j}+\cdots+S_{mj}}{S} I (S_1, S_2,\ldots S_m) \quad (5)$$

dimana perhitungan $I (S_1, S_2,\ldots S_m)$ dan E(A) hanya dilakukan pada atribut yang ada nilainya. Kemudian untuk mencari gain dan atribut A dihitung dengan rumus sebagai berikut:

Gain (A) = Prob S yang diketahui * E(A) (6)

dimana:

A = Atribut dengan missing value yang sedang dicari nilai gainnya

S = Jumlah samples pada subset A yang diketahui nilainya.

Sedangkan nilai split pada atribut A dinyatakan dengan:

$$\text{Spilit(A)} = \text{-u} * Log_2 \text{ u} - \sum_{j=1}^{m} P_j * \text{Log}_2 (P_j) \quad (7)$$

dimana,

U adalah probabilitas samples pada atribut A yang merupakan *missing values*.

$P_j = \frac{S_j}{|S|}$ adalah probabilitas sample S_j yang diketahui nilainya.

Nilai *gain ratio* pada atribut A :

Gain Ratio(A) = Gain(A) / Split(A). (8)

2.4.2.3 Penanganan Atribut Kontinyu[7].

Algoritma C4.5 juga menangani masalah atribut kontinyu. Salah satu cara adalah dengan *Entropy-Based Discretization* yang melibatkan perhitungan kelas antropy.

Misalkan T membagi S example menjadi subset S1 dan S2. Umpamakan ada class C1, C2, …Ck. Misal P(C_i, S_j) menjadi perbandingan dari example pada S_j yang mempunyai class i.

Maka class entropy dari subset S_j didefinisikan dengan:

$$\text{Ent(S)} = \sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S)) \quad (9)$$

Dan class information entropy E(A, T, S)

$$\text{E(A, T, S)} = \frac{|S1|}{|S|} \text{Ent(S1)} + \frac{|S2|}{|S|} \text{Ent(S2)} \quad (10)$$

Dimana Ent(S_j) = class entropy dari subset S_j

S_j = subset dari S

C_i = class i

P(C_i, S_j) = Perbandingan instance dari S_j yang berada pada class C_i

E(A, TA;S) = class information entropy partisi dengan cut point T_A di A

A = Atribut

|Sk| = jumlah instance di Sk

Cut point yang terbaik adalah yang memberikan class information entropy yang paling kecil diantara semua kandidat cut point.

2.4.2.4 Pruning Tree

Pruning tree adalah melakukan suatu kegiatan untuk mengganti suatu subtree dengan suatu *leaf*. Penggantian dilakukan jika *error rate* pada subtree lebih besar jika dibandingkan dengan *single leaf*[7].

Tree Pruning dilakukan untuk menyederhanakan *tree* sehingga akurasi dapat bertambah. *Pruning* ada dua pendekatan, yaitu :

a. *Pre-pruning*, yaitu menghentikan pembangunan suatu *subtree* lebih awal (yaitu dengan memutuskan untuk tidak lebih jauh mempartisi data

training). Saat seketika berhenti, maka *node* berubah menjadi *leaf* (*node* akhir).

b. *Post-pruning*, yaitu menyederhanakan *tree* dengan cara membuang beberapa cabang *subtree* setelah *tree* selesai dibangun.

2.5 Aplikasi Pendukung

2.5.1 Borland Delphi 7

2.5.1.1 Sejarah Delphi

Delphi awalnya sebuah proyek penelitian rahasia di Borland yang berevolusi menjadi sebuah produk yang disebut AppBuilder. Tak lama sebelum rilis pertama dari Borland AppBuilder, Novell AppBuilder dirilis, meninggalkan Borland membutuhkan nama baru.

Pengembang Danny Thorpe memilih nama kode *Delphi* di referensi ke Oracle di Delphi . Salah satu tujuan asli dari Delphi adalah untuk menyediakan konektivitas database untuk programmer sebagai kunci dan sebuah paket database yang populer pada saat itu adalah database Oracle , maka, “Jika Anda ingin berbicara dengan Oracle, pergi ke Delphi”. Sebagai perkembangan lebih lanjut, nama mereka mulai berkembang dan ada dukungan yang tumbuh di dalam nama Borland Delphi.

Arsitek dibelakang suksenya Delphi adalah Anders Hejlsberg , yang telah mengembangkan Turbo Pascal . Ia pindah ke Microsoft pada 1996 .

2.5.1.2 Borland Delphi 7

Delphi 7, dirilis pada bulan Agustus 2002, menjadi versi standar yang digunakan oleh pengembang Delphi lebih dari versi tunggal lainnya. Ini adalah salah satu keberhasilan paling IDE yang diciptakan oleh Borland karena kecepatan, yang stabilitas dan persyaratan perangkat keras rendah dan masih aktif

digunakan untuk tanggal ini (2009). Delphi 7 menambahkan dukungan untuk Windows Theme XP, dan menambahkan lebih banyak dukungan untuk membangun aplikasi Web. Ini juga merupakan versi terakhir dari Delphi yang dapat digunakan tanpa aktivasi.

2.5.1.3 Kegunaan Delphi:

1. Untuk membuat aplikasi pada Windows.
2. Untuk merancang aplikasi berbasis grafis.
3. Untuk membuat program berbasis jaringan *(client/server).*
4. Untuk merancang program .Net (berbasis internet).

2.5.1.4 Keunggulan Delphi:

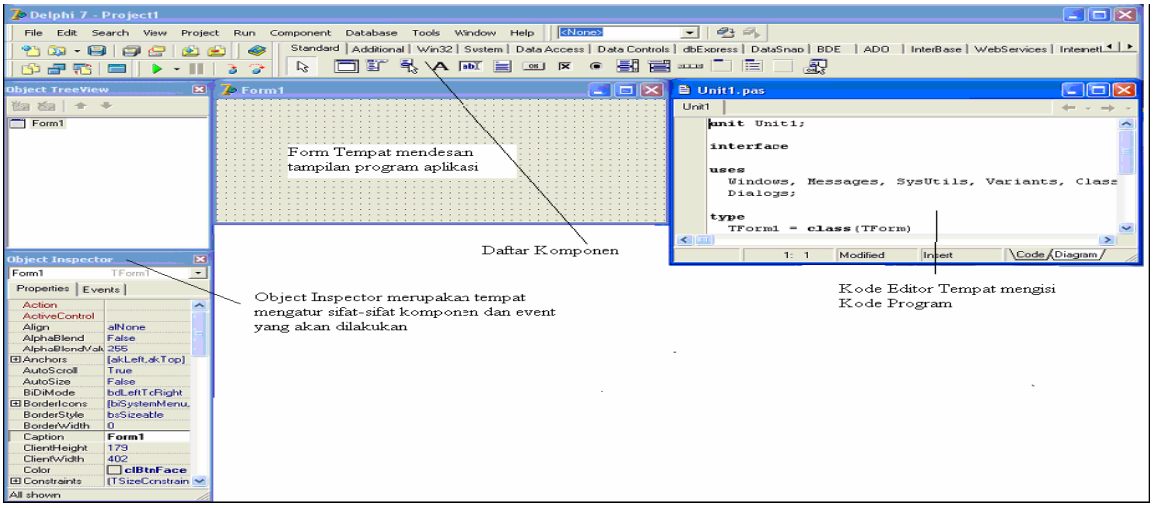
1. Memiliki IDE *(integrated Development Environment)* atau lingkungan pengembangan terintegrasi yang didalamnya terdapat menu-menu yang memudahkan programmer untuk membuat sebuah program aplikasi.
2. Proses kompilasi cepat, pada saat program dijalankan, secara otomatis akan dibaca sebagai sebuah program, tanpa dijalankan terpisah.
3. Mudah digunakan, karena source code Delphi merupakan turunan dari bahasa pemrograman Pascal.
4. Bersifat multi purpose, artinya bahasa pemrograman Delphi dapat digunakan untuk mengembangkan berbagai keperluan pengembangan aplikasi.
5. Dapat mengkompilasi menjadi single executable, memudahkan distribusi dan meminimalisir masalah yang terkait dengan versioning

2.5.1.5 Instalasi IDE Borland Delphi 7



2.11.1.1. Tampilan Delphi

Tampilan utama program Borland Delphi 7 adalah seperti ditunjukkan pada gambar berikut:



Gambar II-3 Tampilan interface Borland Delphi 7

IDE *(Integrated Development Environment)* atau lingkungan pengembangan terpadu pada program Delphi terbagi menjadi delapan bagian utama, yaitu:

1. Main Window
2. ToolBar
3. Component Palette
4. Form Designer
5. Code Editor
6. Object Inspector
7. Object TreeView

Untuk lebih jelasnya dapat dilihat pada gambar 1 di atas. IDE merupakan sebuah lingkungan di mana semua tombol perintah yang diperlukan untuk mendesain aplikasi, menjalankan dan menguji suatu aplikasi disajikan dengan baik untuk memudahkan pengembangan program.

1. Menu Bar

Berfungsi untuk memilih tugas-tugas tertentu, seperti memulai, membuka, dan menyimpan

project, mengompilasi *project* menjadi *file executable* (EXE), dan lain-lain



Gambar II-4 Menu Bar

2. Toolbar

Delphi memiliki beberapa toolbar yang masing-masing memiliki perbedaan fungsi dan setiap tombol pada bagian toolbar berfungsi sebagai pengganti suatu menu perintah yang sering digunakan.



Gambar II-5 Toolbar

Toolbar terletak pada bagian bawah baris menu. Pada kondisi default Delphi memiliki enam bagain toolbar, antara lain: **Standart**, **View**, **Debug**, **Desktops**, **Custom** dan **Componen Palette**.

3. Component Pallete

Component Palette berisi kumpulan ikon yang melambangkan komponen-komponen yang terdapat pada *VCL (Visual Component Library)*. Pada Componen Palette Anda akan menemukan beberapa page control, seperti **Standart**, **Additional**, **Win32**, **System**, **Data Access** dan lain-lain seperti tampak pada gambar 4

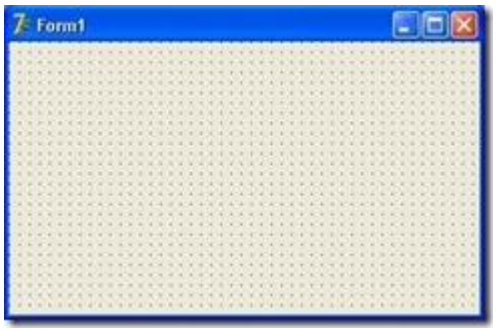


Gambar II-6 Component Palette

Ikon tombol Pointer terdapat di setiap page control. Tombol ini dipakai untuk menekan atau memilih posisi. Jika Anda memilih sebuah item dari sebuah page control, tombol pointer ini akan berada dalam keadaan tidak aktif. Hal ini berarti Anda akan meletakkan komponen pada form, Anda cukup klik pada form.

4. Form Designer

Merupakan suatu objek yang dapat dipakai sebagai tempat untuk merancang program aplikasi. Form berbentuk sebuah meja kerja yang dapat diisi dengan komponen-komponen yang diambil dari Component Palette. Pada saat Anda memulai Delphi, Delphi akan memberikan sebuah form kosong yang disebut form1, seperti gambar 5 di bawah ini.



Gambar II-7 Lembar kerja Form

Sebuah form mengandung unit yang berfungsi untuk mengendalikan form dan Anda dapat mengendalikan komponen- komponen yang terletak dalam form dengan menggunakan Object Inspector dan Code Editor.

5. Code Editor

Code Editor merupakan tempat di mana Anda dapat menuliskan kode program. Pada bagian ini Anda dapat menuliskan pernyataan-pernyataan dalam Object Pascal. Satu diantara keuntungan bagi pengguna Delphi adalah bahwa Anda tidak perlu menuliskan kode-kode sumber, karena Delphi telah menyediakan kerangka penulisan sebuah program seperti pada gambar di bawah ini.



Gambar II-8Lembar kerja Code Editor

6. Object Inspector

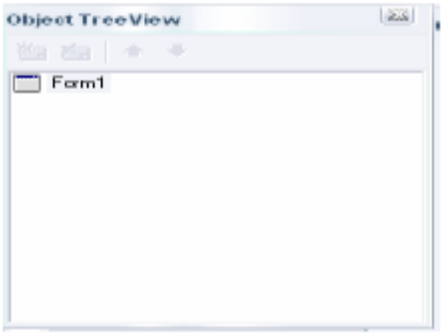
Digunakan untuk mengubah properti atau karakteristik dari sebuah komponen. Object Inspector terdiri dari dua tab, yaitu **Properties** dan **Events** seperti gambar 7 di bawah ini.



Gambar II-9 Lembar kerja Object Inspector

7. Object TreeView

Object TreeView menampilkan diagram pohon dari komponen-komponen yang bersifat visual maupun nonvisual yang telah terdapat dalam form, data module, atau frame. Object TreeView juga menampilkan hubungan logika antar komponen. Apabila Anda mengklik kanan salah satu item yang terdapat di dalam diagram pohon, Anda dapat melihat konteks menu komponen versi sebelumnya. Untuk mengakses menu secara penuh, klik kanan pada komponen yang sama dalam form, data module, atau frame.



Gambar II-10Object TreeView

2.5.2 Interbase

Interbase merupakan *software* yang berguna untuk memanipulasi data-data pada basis data. Interbase ini merupakan produk dari bahasa pemrograman

Borland Delphi yang mengolah Basis data yang berbasiskan *Client/Server*. Interbase sifatnya *open source*. Perusahaan *Borland* berkedudukan di *Scott Valley*, *California*. *Borland* setelah melebur dengan *Visigenic* berganti nama menjadi *Inspirise Coorporation* pada pertengahan tahun 1998. Interbase merupakan *software* yang berguna untuk memanipulasi data-data pada basis data. Interbase ini merupakan produk dari bahasa pemrograman *Borland Delphi* yang mengolah Basis data yang berbasiskan *Client/Server*. Interbase sifatnya *open source*. Perusahaan *Borland* berkedudukan di *Scott Valley*, *California*. *Borland* setelah melebur dengan *Visigenic* berganti nama menjadi *Inspirise Coorporation* pada pertengahan tahun 1998. Karena *software* Interbase tidak terpisah/tidak berdiri sendiri (masih satu kesatuan dalam *Borland Delphi*) maka secara otomatis penyebaran interbase ini sama dengan penyebaran *Borland Delphi*. Interbase banyak digunakan untuk mengolah basis data dengan kapasitas yang sangat besar dan berbasiskan *Client/Server*. Biasanya Interbase ini digunakan pada instansi atau perkantoran yang membutuhkan pengolahan Basis data dengan kapasitas yang sangat besar dan berbasis jaringan. Interbase akan melakukan penyimpanan dan pemrosesan data-data yang diperlukan sehingga bila memerlukan suatu informasi maka akan dengan mudah mendapatkannya dengan menggunakan fasilitas *searching*, *SQL* dan sebagainya yang disediakan dalam komponen Basis data.

2.5.3 Membuat Database

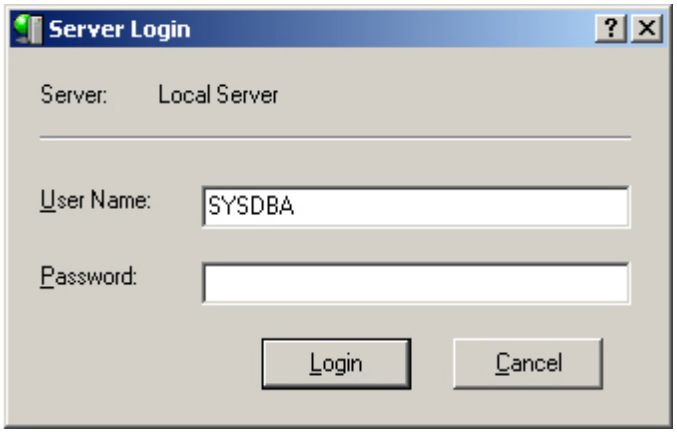
IBConsole merupakan program client yang menyertai Interbase. Superusernya bernama *sysdba* dengan password *masterkey*. Untuk membuat database di interbase dapat mengikuti langkah-langkah sebagai berikut :

1. Jalankan atau aktifkan IBConsole.
2. Klik kanan pada *local server* dan pilih *login.*



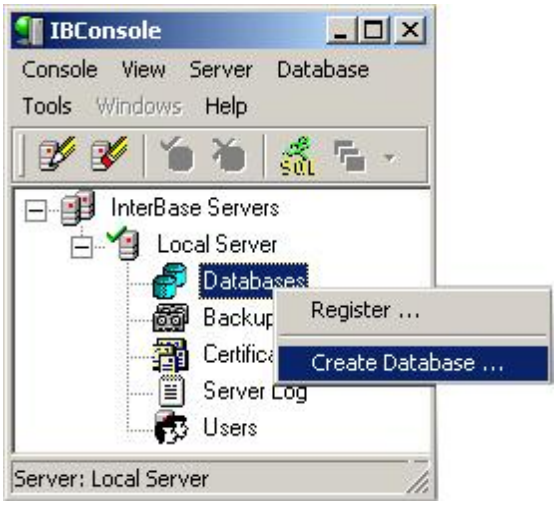
Gambar II-11 Login

3. Masukan *sysdba* dalam user name dan *masterkey* dalam password.



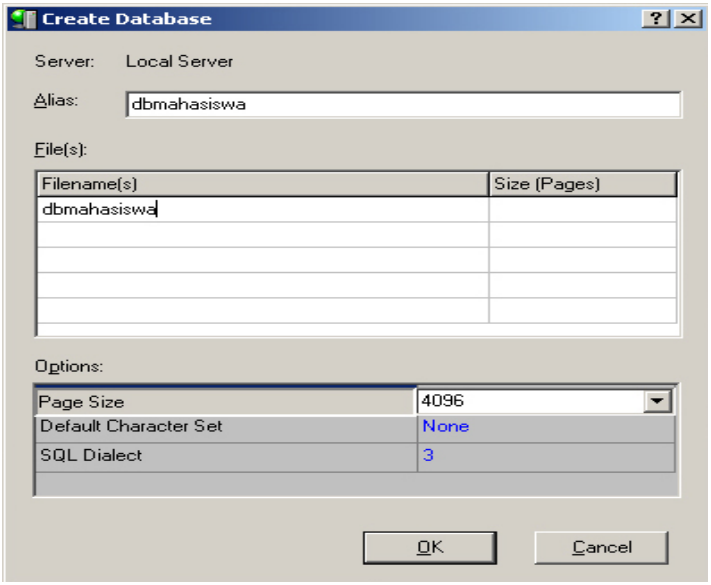
Gambar II-12 Login

4. Klik kanan pada database kemudian pilih *create database*.



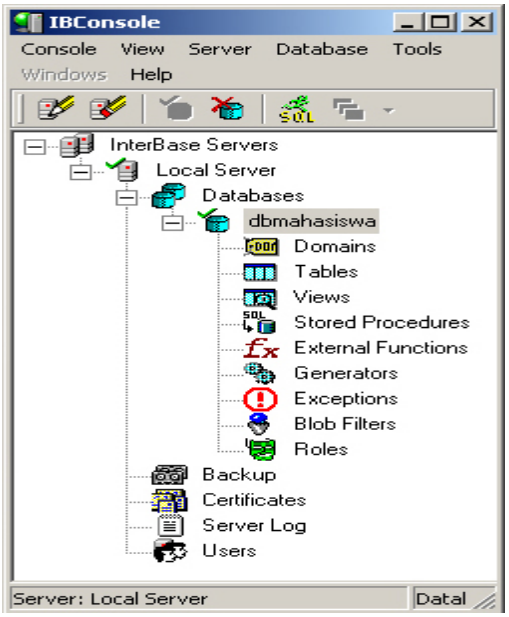
Gambar II-13 Create Database

5. Semisal akan dibuat database dengan nama *dbmahasiswa* dan alias *databasemahasiswa*. Maka isikan databasemahasiswa pada kota alias dan dbcoba pada kolom filename(s). klik OK untuk membuat database atau Cancel untuk membatalkannya.



Gambar II-14 Create Database

6. Jika database berhasil dibuat akan muncul dalam IBConsole sebagai berikut :



Gambar II-15 Daftar Database