Indra Ava Dianta, S.Kom., M.T

YAYASAN PRIMA AGUS TEKNIK

IMPLEMENTASI PEMROGRAMAN Visual VB.NET

Indra Ava Dianta, S.Kom., M.T

IMPLEMENTASI PEMROGRAMAN Visual VB.NET

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

IMPLEMENTASI PEMROGRAMAN Visual VB.NET

Penulis :
Indra Ava Dianta, S.Kom., M.T

ISBN : ***978-623-5734-99-6 (PDF)***

Editor :
Danang, S.Kom., M.T

Penyunting :
Irdha Yunianto, S.Ds., M.Kom.

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

Kata Pengantar

Puji syukur penulis panjatkan kepada Allah SWT atas selesainya buku "**Implementasi Pemrograman Visual VB.NET**" ini. Inti dari pemrograman adalah tentang menciptakan algoritma yang efisien yang memecahkan masalah komputasi yang terdefinisi dengan baik. Desain algoritma membutuhkan pemecahan masalah dan keterampilan matematika. Seringkali solusi untuk suatu masalah adalah kombinasi dari metode terkenal dan wawasan baru. Matematika berperan penting dalam pemrograman kompetitif. Sebenarnya, tidak ada batasan yang jelas antara desain algoritma dan matematika. Buku ini telah ditulis sehingga tidak banyak latar belakang matematika yang dibutuhkan. Lampiran buku mengulas beberapa konsep matematika yang digunakan di seluruh buku, seperti himpunan, logika, dan fungsi, dan lampiran dapat digunakan sebagai referensi saat membaca buku.

Buku Implementasi Pemrograman Visual VB.NET ini mempelajari tentang pemrograman database sistem dan implementasi pemrograman visual dalam beberapa bidang. Penulis menyadari masih banyak kekurangan dalam buku ini, sehingga masukan dan saran sangat diharapkan untuk menyempurnakan buku ini.

Semarang, Desember 2022

Penulis

DAFTAR ISI

BAB 1

PENGANTAR PEMROGRAMAN

1.1 Apa itu pemrograman?

Pemrograman merupakan suatu pemrosesan untuk mengetik/menulis, uji coba yang di tulis dan selanjutnya jika ada kesalahan akan diperbaiki atau disebut debug. Selanjutnya kode program dipelihara untuk membangun suatu computer programming. Kode program bisa ditulis/dibuat dari berbagai bahasa pemrograman. Suatu program yang dimuat kedalam suatu program merupakan tujuan dari pemrograman untuk melakukan perhitungan/pekerjaan sesuai kehendak orang yang membuat program. Perlu memperhatikan beberapa aturan jika ingin membuat suatu kode program antara lain yaitu memahamai tentang logika algoritma beserta dengan ketrampilan dalam memodifikasi algoritma, selanjutnya paham tentang bahasa pemrograman apa yang akan digunakan, jika diperlukan seorang programmer harus mempunyai pengetahuan tentang matematika dan terjun langsung dalam studi kasus.

Dibutuhkan seni yang digunakan untuk membuat suatu pemrograman yang terdiri dari satu atau dua bahkan lebih dalam penggunaan algoritma pemrograman untuk menghubungkan bahasa apa yang akan kita gunakan dalam penerapannya dan selanjutnya akan di proses oleh komputer menjadi sebuah aplikasi/program komputer. Pemrograman mempunyai paradigma yaitu banyak sekali pemrograman menggunakan banyak bahasa sebagai gaya pendukung yang berbeda beda pula.

1.2 Sekilas Tentang Algoritma Pemrograman

Pada zaman yunani kuno untuk memastikan proses pembedahan menggunakan konfigurasi dan berbagai macam dimensi memiliki persneling yang disebut kalkulator, kemudian di lacak menggunakan metonik siklus untuk mempelajari bulan dan matahari untuk kalender digunakan untuk menentukan kapan olimpiade diselenggarakan. Pada tahun 1206 masehi memprogram automatic membentuk Al-Jazari, apa itu al-jazari? Aljazari merupakan komponen yang membentuk suatu unit disebut sistem yang didalamnya ada fitur drum kayu pada posisi tertentu dan didalamnya terdapat cams serta pasak yang bisa dipakai untuk mengoperasikan perkusi dengan instrumen yang di dapat dari drum kayu secara berurutan pada saat gilirannya akan menarik tuas.

Inti dari pemrograman adalah tentang menciptakan algoritma yang efisien yang memecahkan masalah komputasi yang terdefinisi dengan baik. Desain algoritma membutuhkan pemecahan masalah dan keterampilan matematika. Seringkali solusi untuk suatu masalah adalah kombinasi dari metode terkenal dan wawasan baru. Matematika berperan penting dalam pemrograman kompetitif. Sebenarnya, tidak ada batasan yang jelas antara desain algoritma dan matematika. Buku ini telah ditulis sehingga tidak banyak latar belakang matematika yang dibutuhkan. Lampiran buku mengulas beberapa konsep

matematika yang digunakan di seluruh buku, seperti himpunan, logika, dan fungsi, dan lampiran dapat digunakan sebagai referensi saat membaca buku.

Pemrograman tidak kompetitif, solusi untuk masalah dievaluasi dengan menguji algoritma yang diimplementasikan menggunakan satu set kasus uji. Jadi, setelah menemukan algoritma yang memecahkan masalah, langkah selanjutnya adalah mengimplementasikannya dengan benar, yang membutuhkan keterampilan pemrograman yang baik. Pemrograman kompetitif sangat berbeda dari rekayasa perangkat lunak tradisional: program pendek (biasanya paling banyak beberapa ratus baris), program harus ditulis dengan cepat, dan tidak perlu mempertahankannya setelah kontes. Saat ini, bahasa pemrograman yang paling populer digunakan dalam kontes adalah .NET, C++, Python, dan Java. Misalnya, di Google Code Jam 2017, di antara3.000 peserta terbaik, 79% menggunakan C++, 16% menggunakan Python, dan 8% menggunakan Java. Banyak orang menganggap .NET sebagai pilihan terbaik untuk programmer. Manfaat menggunakan .NET adalah bahasa yang sangat efisien dan pustaka standarnya berisi kumpulan besar struktur data dan algoritma. Semua contoh program dalam buku ini ditulis dalam .NET Framework dan struktur data dan algoritma perpustakaan standar sering digunakan. Program mengikuti standar, yang dapat digunakan di sebagian besar kontes saat ini. Jika Anda belum dapat memprogram dalam .NET, mulailah untuk belajar dari sekarang.

Mempelajari pemrograman membutuhkan banyak pekerjaan. Namun, ada banyak cara untuk berlatih, dan antara satu dengan yang lainnya memiliki keunggulan dan kelemahan. Ketika memecahkan masalah, kita harus ingat bahwa jumlah masalah yang diselesaikan tidak begitu penting daripada kualitas masalah. Sangat menggoda untuk memilih masalah yang terlihat bagus dan mudah dan menyelesaikannya, dan melewatkan masalah yang terlihat sulit dan membosankan. Namun, cara untuk benar-benar meningkatkan keterampilan seseorang adalah dengan fokus pada jenis masalah yang terakhir.

Pengamatan penting lainnya adalah bahwa sebagian besar masalah kontes pemrograman dapat diselesaikan dengan menggunakan algoritme sederhana dan pendek, tetapi bagian yang sulit adalah menemukan algoritme. Pemrograman kompetitif bukan tentang mempelajari algoritme yang kompleks dan tidak jelas, melainkan tentang mempelajari pemecahan masalah dan cara mendekati masalah yang sulit menggunakan alat sederhana.

Terakhir, beberapa orang membenci implementasi algoritma: mendesain algoritma itu menyenangkan tetapi membosankan untuk mengimplementasikannya. Namun, kemampuan untuk mengimplementasikan algoritma dengan cepat dan benar merupakan aset penting, dan keterampilan ini dapat dipraktikkan. Adalah ide yang buruk untuk menghabiskan sebagian besar waktu kontes untuk menulis kode dan menemukan bug, daripada memikirkan bagaimana memecahkan masalah.

Tentu saja, buku algoritme umum juga merupakan bacaan yang bagus untuk programmer yang kompetitif. Yang paling komprehensif adalah Pengantar Algoritma yang ditulis oleh Cormen, Leiserson, Rivest, dan Stein, juga disebut CLRS. Buku ini adalah sumber yang bagus jika Anda ingin memeriksa semua detail tentang algoritma dan bagaimana membuktikan dengan sungguh-sungguh bahwa itu benar. Desain Algoritma Kleinberg dan Tardos berfokus pada teknik desain algoritma, dan secara menyeluruh membahas metode membagi dan menaklukkan, algoritma serakah, pemrograman dinamis, dan algoritma aliran maksimum. Algorithm Design Manual dari Skiena adalah buku yang lebih praktis yang mencakup katalog besar masalah komputasi dan menjelaskan cara menyelesaikannya. Algoritma Rekursi sering memberikan cara yang elegan untuk mengimplementasikan suatu algoritma. Pada bagian ini, kita membahas algoritma rekursif yang secara sistematis melalui kandidat solusi untuk suatu masalah. Pertama, kita fokus pada pembangkitan himpunan bagian dan permutasi dan kemudian membahas teknik backtracking yang lebih umum. Algoritma backtracking dimulai dengan solusi kosong dan memperluas solusi langkah demi langkah. Pencarian secara rekursif melewati semua cara yang berbeda bagaimana solusi dapat dibangun. Sebagai contoh, perhatikan masalah menghitung jumlah ratu yang dapat ditempatkan pada papan catur nxn sehingga tidak ada dua ratu yang saling menyerang. Masalah dapat diselesaikan dengan menggunakan backtracking dengan menempatkan ratu di papan baris demi baris. Lebih tepatnya, tepat satu ratu akan ditempatkan di setiap baris sehingga tidak ada ratu yang menyerang salah satu ratu yang ditempatkan sebelumnya. Sebuah solusi telah ditemukan ketika semua n ratu telah ditempatkan di papan tulis.

Efisiensi algoritme memainkan peran sentral dalam pemrograman. Dalam bab ini, kita mempelajari alat yang mempermudah merancang algoritme yang efisien. Bab ini akan memperkenalkan konsep kompleksitas waktu, yang memungkinkan untuk memperkirakan waktu berjalan dari algoritma tanpa menerapkannya. Kompleksitas waktu dari suatu algoritma menunjukkan seberapa cepat waktu berjalannya meningkat ketika ukuran input bertambah. Bagianselanjutnya menyajikan dua contoh masalah yang dapat diselesaikan dengan banyak cara. Dalam kedua masalah tersebut, kita dapat dengan mudah merancang solusi brute force yang lambat, tetapi ternyata kita juga dapat membuat algoritme yang jauh lebih efisien. Kompleksitas waktu dari perkiraan algoritma menunjukkan banyak waktu yang akan digunakan oleh algoritma untuk input yang diberikan. Dengan menghitung kompleksitas waktu, kita sering dapat mengetahui apakah algoritma tersebut cukup cepat untuk menyelesaikan suatu masalah—tanpa mengimplementasikannya. Kompleksitas waktu dilambangkan dengan O (⋯) di mana tiga titik mewakili beberapa fungsi. Biasanya, variabel n menunjukkan ukuran input. Misalnya, jika inputnya adalah array angka, n akan menjadi ukuran array, dan jika inputnya adalah string, n akan menjadi panjang string. Kompleksitas waktu dari perkiraan algoritma menunjukkan banyak waktu yang akan digunakan oleh

algoritma untuk input yang diberikan. Dengan menghitung kompleksitas waktu, kita sering dapat mengetahui apakah algoritma tersebut cukup cepat untuk menyelesaikan suatu masalah—tanpa mengimplementasikannya. Kompleksitas waktu dilambangkan dengan O (···) di mana tiga titik mewakili beberapa fungsi. Biasanya, variabel n menunjukkan ukuran input. Misalnya, jika inputnya adalah array angka, n akan menjadi ukuran array, dan jika inputnya adalah string, n akan menjadi panjang string.

```
for (int i = 1; i <= n; i++) {
    ...
}
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        ...
    }
}
for (int i = 1; i <= n; i++) {
    ...
}
```

Terkadang kompleksitas waktu tergantung pada beberapa faktor, dan rumus kompleksitas waktu mengandung beberapa variabel. Misalnya, kompleksitas waktu dari kode berikut adalah O(nm). Kompleksitas waktu dari fungsi rekursif tergantung pada berapa kali fungsi dipanggil dan kompleksitas waktu dari satu panggilan. Kompleksitas waktu total adalah produk dari nilai-nilai ini. Sebagai contoh, perhatikan fungsi berikut:

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
        ...
    }
}
```

Pemanggilan f(n) menyebabkan n pemanggilan fungsi, dan kompleksitas waktu setiap pemanggilan adalah O(1), sehingga total kompleksitas waktu adalah O(n). Daftar berikut berisi kompleksitas waktu umum dari algoritma:

- O (1) Waktu berjalan dari algoritma waktu-konstan tidak bergantung pada ukuran input. Algoritma waktu-konstan yang khas adalah formula langsung yang menghitung jawabannya.
- O (log n) Sebuah algoritma logaritmik sering membagi dua ukuran input pada setiap langkah. Waktu berjalan dari algoritma tersebut adalah logaritma, karena log2 n sama dengan berapa kali n harus dibagi 2 untuk mendapatkan 1. Perhatikan bahwa basis logaritma tidak ditampilkan dalam kompleksitas waktu.
- O(√n) Algoritma akar kuadrat lebih lambat dari O(log n) tetapi lebih cepat dari O(n) Sifat khusus akar kuadrat adalah bahwa n =n/√n, s pada elemen dapat dibagi menjadi O(√n) blok elemen O(√n).

- O(n) Algoritme linier melewati input beberapa kali secara konstan. Ini seringkali merupakan kompleksitas waktu terbaik, karena biasanya diperlukan untuk mengakses setiap elemen input setidaknya sekali sebelum melaporkan jawabannya.
- O(n log n) Kompleksitas waktu ini sering menunjukkan bahwa algoritma mengurutkan input, karena kompleksitas waktu dari algoritma pengurutan yang efisien adalah O(n log n). Kemungkinan lain adalah bahwa algoritma menggunakan struktur data di mana setiap operasi membutuhkan waktu O(log n).
- O(n2) Sebuah algoritma kuadrat sering berisi dua loop bersarang. Dimungkinkan untuk melewati semua pasangan elemen input dalam waktu O(n2).
- O(n3) Sebuah algoritma kubik sering berisi tiga loop bersarang. Dimungkinkan untuk melewati semua triplet elemen input dalam waktu O(n3).
- O(2n) Kompleksitas waktu ini sering menunjukkan bahwa algoritme iterasi melalui semua himpunan bagian dari elemen input. Misalnya, himpunan bagian dari {1, 2, 3} adalah ∅, {1}, {2}, {3}, {1, 2}, {1, 3},{2, 3}, dan {1, 2, 3}.
- O (n!) Kompleksitas waktu ini sering menunjukkan bahwa algoritme iterasi melalui semua permutasi elemen input. Misalnya, permutasi {1, 2, 3} adalah (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1 , 2), dan (3, 2, 1).

Suatu algoritma disebut polinomial jika kompleksitas waktunya paling banyak O(nk) di mana k adalah konstanta. Semua kompleksitas waktu di atas kecuali O(2n) dan O(n!) adalah polinomial. Dalam praktiknya, konstanta biasanya kecil, dan oleh karena itu kompleksitas waktu polinomial secara kasar berarti bahwa algoritma dapat memproses input yang besar . Kebanyakan algoritma dalam buku ini adalah polinomial. Namun, ada banyak masalah penting yang algoritma polinomialnya tidak diketahui, yaitu, tidak ada yang tahu bagaimana menyelesaikannya secara efisien. Masalah NP- hard adalah kumpulan masalah penting, yang tidak diketahui algoritma polinomialnya.

BAB 2

PENGANTAR PEMROGRAMAN DATABASE

2.1 DATA WAREHOUSE

Data warehouse yaitu data yang dikumpulkan dan dirancang guna mendukung manajemen dalam proses sebuah keputusan di eksekusi. Ini adalah kumpulan suatu data yang memiliki orientasi pada subjek, terintegrasi tiap bagian, memiliki waktu yang bervariasi , tidak dapat didata, yang digunakanuntuk mendukung proses pengambilan keputusan manajemen dan intelijen bisnis. Ini berisi berbagai macam data yang menyajikan gambaran yang koheren tentang kondisi bisnis pada satu titik waktu. Ini adalah jenis database unik yang berfokus pada intelijen bisnis, data eksternal, dan data varian waktu. Pergudangan data adalah proses, di mana organisasi mengekstrak makna dan pengambilan keputusan informasi dari aset informasi mereka melalui penggunaan Data warehouse.

2.2 DATA DICTIONARY (KAMUS DATA)

Data Dictionay adalah sistem manajemen Database mini yang mengelola metadata. Ini adalah gudang informasi tentang database yang mendokumentasikan elemen data dari database. Sistem manajemen database pada bagian integral diartikan sebagai kamus data dan menyimpan metadata atau informasi tentang database, nama atribut dan definisi untuk setiap tabel dalam database. Kamus data biasanya merupakan bagian dari katalog sistem yang dihasilkan untuk setiap database. Sistem kamus data yang berguna biasanya menyimpan dan mengelola jenis informasi berikut:

- Deskripsi skema database.
- Informasi rinci tentang desain database fisik, seperti struktur penyimpanan, jalurakses dan ukuran file dan catatan.
- Deskripsi pengguna database, tanggung jawab dan hak akses mereka.
- Deskripsi tingkat tinggi dari transaksi & aplikasi database dan hubungan penggunadengan terjemahan.
- Hubungan antara transaksi database dan item data yang direferensikan oleh mereka. Ini berguna dalam menentukan transaksi mana yang terpengaruh ketika definisi data tertentu diubah.

Catatan

Catatan adalah kumpulan bidang atau item data yang terkait secara logis, dengan setiap bidang memproses sejumlah byte dan memiliki tipe data tetap. Sebuah record terdiri dari nilai-nilai untuk

setiap field. Pengelompokan item data dapat dicapai melalui cara yang berbeda untuk membentuk catatan yang berbeda untuk tujuan yang berbeda. Catatan ini diambil atau diperbarui menggunakan program.

2.3 FILE

File merupakan kumpulan dari urutan record yang saling terhubung. Dalam banyak kasus, semua record dalam sebuah file memiliki tipe record yang sama (setiap record memiliki format yang sama). Jika setiap record dalam file memiliki ukuran yang sama persis dalam byte, file tersebut dikatakan terdiri dari record dengan panjang tetap. Jika record yang berbeda dalam file memiliki ukuran yang berbeda, file tersebut dikatakan dibuat dari recorddengan panjang variabel.

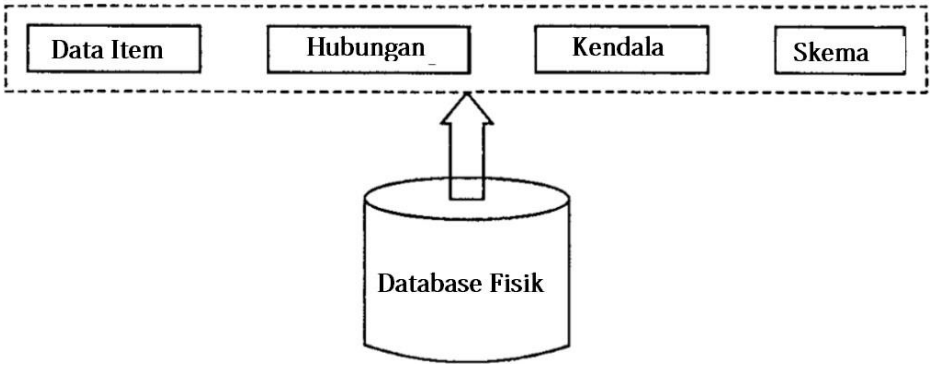
2.4 DATABASE

Database diartikan data yang dikumpulkan mempunyai keterkaitan secara logis kemudian data tersebut di simpan secara bersama dan di rancang sebagai pemenuhan kebutuhan suatu organisasi maupun suatu informasi. Database lebih lanjut dapat dijabarkan di bawah ini:

- Adalah data yang dikumpulkan dan memiliki keterkaitan data kemudian di simpan secara bersama dengan menghilangkan data yang kembar/redudanci yang dianggap berbahaya.
- Database melayani banyak aplikasi di mana setiap pengguna memiliki pandangannya sendiri tentang data. Data ini dilindungi dari akses yang tidak sah oleh mekanisme keamanan dan akses bersamaan ke data disediakan dengan mekanisme pemulihan.
- menyimpan data independen dari program dan perubahan dalam struktur penyimpanan data atau strategi akses tidak memerlukan perubahan dalam mengakses program atau kueri.

Sebuah database terdiri dari empat komponen berikut seperti yang ditunjukkan pada gambar.

- Data item
- Hubungan
- Kendala
- Skema



Gambar 2.1 Database Fisik

2.5 SISTEM MANAJEMEN DATABASE

Sistem Manajemen Database merupakan data yang dikumpulkan dan mempunyai keterkaitan yang digunakan untuk masuk/mengakses data yang diinginkan.

Aplikasi pada Sistem Databased

Berikut ini merupakan aplikasi database.

- Perbankan
- Maskapai penerbangan
- Universitas
- Kereta Api
- Keuangan
- Penjualan
- Telekomunikasi
- Sistem penggajian
- Manufaktur
- Sumber daya manusia

Fungsi dan Pelayanan DBMS

Sebuah DBMS melakukan beberapa fungsi penting yang menjamin integritas dan konsistensi data dalam database.

1. ***Data Storage Management/Manajemen Penyimpanan Data***: DBMS menciptakan struktur kompleks yang diperlukan untuk penyimpanan data dalam database fisik. Ini menyediakan mekanisme untuk pengelolaan penyimpanan permanen data.

2. ***Manajemen Transaksi:*** Transaksi adalah serangkaian operasi Database, yang dilakukan oleh program aplikasi, yang mengakses atau mengubah isi Database. Oleh karena itu, DBMS harus menyediakan mekanisme untuk memastikan bahwa semua pembaruan yang terkait dengan transaksi tertentu dilakukan atau tidak ada satu punyang dibuat.
3. ***Layanan Integritas***: database yang memiliki tingkat integritas tinggi mengacu pada fakta/kevalidan/kebenaran dan konsistensi suatu data yang disimpan dan sangat penting dalam sistem database berorientasi transaksi. Oleh karena itu, DBMS harus menyediakan untuk memastikan bahwa baik data dalamdatabase dan perubahan data mengikuti aturan tertentu. Ini meminimalkan redundansi data dan memaksimalkan konsistensi data. Hubungan data yang disimpan dalam kamus data digunakan untuk menegakkan integritas data. Berbagai jenis mekanisme dan batasan integritas dapat didukung untuk membantu memastikan bahwa nilai data dalam database valid, bahwa operasi yang dilakukan pada nilai tersebut valid dan database tetap dalam keadaan konsisten.
4. ***Manajemen Pencadangan dan Pemulihan***: DBMS menyediakan mekanisme untuk berbagai jenis kegagalan. Ini mencegah hilangnya data. Mekanisme pemulihan DBMS, memastikan bahwa database dikembalikan ke keadaan yang konsisten setelah transaksi gagal atau dibatalkan karena sistem crash, kegagalan media, kesalahan perangkat keras atau perangkat lunak, kegagalan daya, dan sebagainya.
5. ***Concurrency Control Services:*** Karena DBMS mendukung berbagi data di antara banyak pengguna, mereka harus menyediakan mekanisme untuk mengelola akses bersamaan ke database. DBMS memastikan bahwa database disimpan dalam keadaan yang konsisten dan integritas data dipertahankan. Ini memastikan bahwa database diperbarui dengan benar ketika banyak pengguna memperbarui database secara bersamaan.
6. ***Data Manipulation Management/Manajemen Manipulasi Data:*** DBMS melengkapi pengguna dengan kemampuan untuk mengambil, memperbarui dan menghapus data yang ada dalam database atau untuk menambahkan data baru ke database. Ini termasuk komponen prosesor DML untuk menangani bahasa manipulasi data (DML).
7. ***Data Dictionary/Manajemen Katalog Sistem:*** DBMS mempunyai kamus yang didalamnya terdapat data atau fungsi katalog sistem di mana deskripsi item data disimpan dan dapat diakses oleh pengguna. Katalog sistem atau kamus data adalah Database sistem, yang merupakan tempat penyimpanan informasi yang menjelaskan data dalam Database. Ini adalah data tentang data atau metadata. Misalnya, DBMS akan berkonsultasi dengan katalog sistem untuk memverifikasi bahwa tabel yang diminta ada dan pengguna yang

mengeluarkan permintaan memiliki hak akses yang diperlukan.

8. ***Otorisasi/Manajemen Keamanan*** : DBMS melindungi database terhadap akses yang tidak sah, baik disengaja atau tidak disengaja. Ini melengkapi mekanisme untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses database. Ini menciptakan sistem keamanan yang memberlakukan keamanan pengguna dan privasi data dalam database. Aturan keamanan menentukan pengguna mana yang dapat mengakses database, item data mana yang dapat diakses setiap pengguna, dan operasi data mana (menambah, menghapus, dan memodifikasi)yang dapat dilakukan pengguna.
9. ***Utility Services/Layanan Utilitas:*** DBMS menyediakan satu set layanan utilitas yang digunakan oleh DBA dan perancang Database untuk membuat, mengimplementasikan, memantau, dan memelihara Database. Layanan utilitas ini membantu DBA untuk mengelola database secara efektif.

10. ***Akses Database dan Antarmuka Pemrograman Aplikasi***: Semua DBMS menyediakan antarmuka untuk memungkinkan aplikasi menggunakan layanan DBMS. Mereka menyediakan akses data melalui bahasa kueri terstruktur (SQL). Bahasa query DBMS berisi dua komponen:
    a. Bahasa definisi data*/data definition language* (DDL)
    b. Bahasa manipulasi data*/data manipulation language* (DML)

mendefinisikan struktur di mana data disimpan dan DML memungkinkan pengguna akhir untuk mengekstrak data dari database. DBMS juga menyediakan akses data ke pemrogram aplikasi melalui bahasa prosedural seperti C, C++, Java dan lain-lain.

11. ***Data Independence Services/Layanan Independensi Data:*** DBMS harus mendukung independensi program dari struktur database yang sebenarnya.
12. ***Data Definition Services/Layanan Definisi Data:*** DBMS menerima definisi data seperti skema eksternal, skema konseptual, skema internal, dan semua pemetaan terkait dalam bentuk sumber. Ini mengubahnya ke bentuk objek yang sesuai menggunakan komponen prosesor DDL untuk masing-masing dari berbagai bahasa definisi data (DDL).

Database versus Sistem File

File adalah urutan record.

- Semua record dalam sebuah file memiliki tipe record yang sama.
- Sistem pemrosesan file didukung oleh sistem operasi konvensional. Sistem menyimpan

catatan permanen dalam berbagai file, dan memerlukan program aplikasi yang berbeda untuk mengekstrak catatan dari file yang sesuai dan menambahkan catatan ke file yang sesuai.

Kelebihan dari Sistem Pemrosesan File

Meskipun sistem pemrosesan file sekarang sebagian besar sudah usang, berikut adalah keuntungan dari sistem pemrosesan file.

- Ini memberikan perspektif sejarah yang berguna tentang cara menangani data.
- Karakteristik sistem berbasis file membantu dalam pemahaman keseluruhan kompleksitas desain sistem database.
- Memahami masalah dan pengetahuan tentang batasan yang melekat pada sistem berbasis file membantu menghindari masalah yang sama saat merancang sistem Database dan dengan demikian menghasilkan transisi yang mulus.

Kekurangan Sistem Pemrosesan File

1. ***Upaya Pemrograman Berlebihan*** : Program aplikasi baru sering kali membutuhkan kumpulan definisi file yang sama sekali baru. Meskipun file yang ada mungkin berisi beberapa data yang diperlukan, aplikasi seringkali membutuhkan sejumlah item datalainnya. Akibatnya, programmer harus mengkode ulang definisi item data yangdibutuhkan dari file yang ada serta definisi semua item data baru. Jadi dalam sistemberorientasi file, ada saling ketergantungan yang berat antara program dan data.
2. ***Inkonsistensi Data:*** Redundansi Data juga menyebabkan inkonsistensi data, karena format data mungkin tidak konsisten atau nilai data mungkin tidak lagi sesuai atau keduanya.
3. ***Berbagi data terbatas*** : Ada peluang berbagi data terbatas dengan sistem berorientasi file tradisional. Setiap aplikasi memiliki file pribadinya sendiri dan pengguna memiliki sedikit kesempatan untuk berbagi data di luar aplikasi mereka sendiri. Untuk mendapatkan data dari beberapa file yang tidak kompatibel dalam sistem yang terpisah akan membutuhkan upaya pemrograman yang besar.
4. ***Kontrol data yang buruk:*** Sistem berorientasi file bersifat desentralisasi, tidak ada kontrol terpusat pada tingkat elemen data (bidang). Bisa jadi sangat umum untuk bidang data memiliki beberapa nama yang ditentukan oleh berbagai departemen organisasi dan tergantung pada file yang ada di dalamnya. Hal ini dapat menyebabkan arti yang berbeda dari data yang diajukan dalam konteks yang berbeda, dan sebaliknya, arti yang sama untuk bidang yang berbeda. Hal ini menyebabkan kontrol data yang buruk, mengakibatkan kebingungan besar.

5. ***Kemampuan manipulasi data yang tidak memadai:*** Karena sistem berorientasi file tidak menyediakan koneksi yang kuat antara data dalam file yang berbeda dan oleh karena itu kemampuan manipulasi datanya sangat terbatas.
6. ***Redundansi Data (atau duplikasi):*** Aplikasi dikembangkan secara independen dalam sistem pemrosesan file yang mengarah ke file duplikat yang tidak direncanakan. Duplikasi adalah pemborosan karena membutuhkan ruang penyimpanan tambahan dan perubahan dalam satu file harus dilakukan secara manual di semua file. Hal ini juga mengakibatkan hilangnya integritas data. Ada juga kemungkinan bahwa item data yang sama mungkin memiliki nama yang berbeda dalam file yang berbeda, atau nama yang sama dapat digunakan untuk item data yang berbeda dalam file yang berbeda.
7. ***Masalah atomisitas:*** Atomisitas berarti semua operasi transaksi tercermin dalam database atau tidak ada sama sekali, yaitu, jika semuanya bekerja dengan benar tanpa kesalahan, maka semuanya akan dikomit ke database. Jika ada bagian dari transaksi yang gagal, seluruh transaksi akan dibatalkan. Transfer dana harus atomik - itu harus terjadi secara keseluruhan atau tidak sama sekali. Sulit untuk memastikan atomisitas dalam sistem pemrosesan file konvensional.
8. ***Masalah keamanan :*** Masalah keamanan dalam pemrosesan file adalah orang yang tidak berwenang dapat mengambil, mengubah, menghapus, atau memasukkan data ke dalam sistem file. Tapi ini tidak mungkin di DBMS.
9. ***Masalah integritas :*** Nilai data yang disimpan dalam database harus memenuhi jeniskendala konsistensi tertentu. Pengembang memberlakukan batasan ini dalam sistem dengan menambahkan kode yang sesuai di berbagai program aplikasi. Ketika kendala baru ditambahkan, sulit untuk mengubah program untuk menegakkannya. Masalahnya diperparah ketika kendala melibatkan beberapa item data untuk file yang berbeda.
10. ***Ketergantungan Data Program :*** Deskripsi file (struktur fisik, penyimpanan file data dan catatan) didefinisikan dalam setiap program aplikasi yang mengakses file tertentu.
11. ***Isolasi data :*** Karena data tersebar di berbagai file, dan file mungkin dalam format yang berbeda, menulis program aplikasi baru untuk mengambil data yang sesuai menjadi sulit.
12. ***Kesulitan dalam mengakses data:*** Lingkungan pemrosesan file konvensional tidak memungkinkan data yang dibutuhkan diambil dengan cara yang nyaman dan efisienseperti DBMS. Sistem pengambilan data yang lebih baik harus dikembangkan untuk penggunaan umum.

13. ***Anomali akses serentak :*** Untuk meningkatkan kinerja sistem secara keseluruhan dan memperoleh waktu respons yang lebih cepat, banyak sistem memungkinkan banyak pengguna untuk memperbarui data secara bersamaan. Dalam lingkungan seperti itu, interaksi pembaruan bersamaan dapat menghasilkan data yang tidak konsisten.

Kelebihan DBMS

1. ***Mengontrol redundansi data:*** Redundansi data, menyimpan data yang sama beberapa kali menyebabkan beberapa masalah. Pertama, ruang penyimpanan terbuang ketika data yang sama disimpan berulang kali. Kedua, file yang mewakili data yang sama mungkin menjadi tidak konsisten. Ini mungkin terjadi karena pembaruan diterapkan ke beberapa file tetapi tidak untuk yang lain. Sebagian besar DBMS menyediakan fasilitas untuk mengontrol redundansi data menggunakan konsep normalisasi dan kunci.
2. ***Membatasi akses yang tidak sah:*** Ketika beberapa pengguna mengakses database, oleh karena itu beberapa pengguna tidak akan diizinkan untuk mengakses (!. Semua informasi dalam database. DBMS harus menyediakan subsistem keamanan dan otorisasi, yang ditentukan oleh Data Base Administor (DBA). DBMS kemudian harus memberlakukan pembatasan ini secara otomatis.Misalnya, data Perbankan sering dianggap rahasia, dan karenanya hanya orang yang berwenang yang diizinkan untukmengakses data tersebut.
3. ***Menyediakan backup dan Recovery :*** Sebuah DBMS harus menyediakan fasilitas untuk recovery dari hardware atau software. Subsistem pencadangan dan pemulihan DBMS bertanggung jawab atas pemulihan. Misalnya, jika sistem komputer gagal di tengah program pembaruan yang kompleks, subsistem pemulihan bertanggung jawab dan memastikan bahwa database dipulihkan ke keadaan sebelum program mulai dijalankan. Alternatifnya, subsistem pemulihan memastikan bahwa program dilanjutkan dari titik di mana ia terganggu sehingga efek penuhnya dicatat dalam database.
4. ***Menyediakan Beberapa Antarmuka*** :Pengguna Karena banyak jenis pengguna dengan berbagai tingkat pengetahuan teknis menggunakan database, DBMS harus menyediakan berbagai antarmuka pengguna. Ini termasuk bahasa kueri untuk pengguna biasa. Antarmuka bahasa pemrograman untuk pemrogram aplikasi, formulir dan kode perintah untuk pengguna parameter dan antarmuka pengguna grafis untuk pengguna yang berdiri sendiri.
5. ***Inforcing Integritas kendala :*** Integritas data berarti bahwa data yang terkandung dalam database akurat dan konsisten. Integritas berarti batasan, yaitu aturan konsistensi yang tidak boleh dilanggar oleh sistem Database. Sebagian besar aplikasi database memiliki

batasan integritas tertentu yang harus dimiliki untuk data. SebuahDBMS harus menyediakan kemampuan untuk mendefinisikan dan menegakkan kendala ini. Jenis batasan integritas paling sederhana yang menentukan tipe data untuk setiap item data.

6. ***Akses data yang efisien:*** DBMS menggunakan berbagai teknik canggih untuk menyimpan dan mengambil data secara efisien. Fitur ini sangat penting jika data disimpan di perangkat penyimpanan eksternal.
7. ***Peningkatan berbagi data: Karena, sistem Database adalah penyimpanan data terpusat milik seluruh organisasi,*** sistem Database dapat dibagikan oleh semua pengguna yang berwenang. Program aplikasi yang ada dapat berbagi data dalam database. Selanjutnya, program aplikasi baru dapat dikembangkan pada data yang ada di database untuk berbagi data yang sama dan hanya menambahkan data yang saat ini tidak disimpan. Oleh karena itu, lebih banyak pengguna dan aplikasi dapat berbagi lebih banyak data.
8. ***Peningkatan keamanan:*** Keamanan Database adalah perlindungan Database dari pengguna yang tidak berwenang. Database administrator (DBA) memastikan bahwa prosedur akses yang tepat diikuti, termasuk skema otentikasi yang tepat untuk akses ke DBMS dan pemeriksaan tambahan sebelum mengizinkan akses ke data sensitif. DBA dapat menentukan nama pengguna dan kata sandi untuk mengidentifikasi orang yang berwenang menggunakan database.
9. ***Konsistensi data yang ditingkatkan:*** Jika redundansi dihapus atau dikendalikan, kemungkinan data yang tidak konsisten juga dihapus dan dikendalikan. Dalam sistem database, inkonsistensi seperti itu dihindari sampai batas tertentu dengan membuatmereka tahu ke DBMS. DBMS memastikan bahwa setiap perubahan yang dibuat ke salah satu dari dua entri dalam database secara otomatis diterapkan ke yang lain juga. Proses ini dikenal sebagai menyebarkan pembaruan.
10. ***Kemandirian data program:*** Dalam lingkungan Database, memungkinkan perubahan pada satu tingkat Database tanpa mempengaruhi tingkat lainnya. Perubahan ini diserap oleh pemetaan antar level dengan pendekatan database, metedata disimpandi lokasi sentral yang disebut repository. Properti sistem data ini memungkinkan data organisasi berubah tanpa mengubah program aplikasi yang memproses data.
11. ***Peningkatan kualitas data:*** Sistem database menyediakan sejumlah alat dan proses untuk meningkatkan kualitas data.

12. ***Menyediakan penyimpanan persisten untuk objek program dan struktur data:*** Database dapat digunakan untuk menyediakan penyimpanan persisten untuk objek program dan struktur data. Ini adalah salah satu alasan utama untuk sistem database berorientasi objek. Penyimpanan persisten dari objek program dan struktur data merupakan fungsi penting dari sistem database.
13. ***Mewakili hubungan yang kompleks antara data:*** Sebuah database dapat mencakup berbagai jenis data yang saling terkait dalam banyak cara. Sebuah DBMS harus memiliki kemampuan untuk mewakili berbagai hubungan yang kompleks antara data serta untuk mengambil dan memperbarui data dengan mudah dan efisien.
14. ***Mengizinkan inferensi dan tindakan menggunakan aturan:*** Beberapa sistem database menyediakan kemampuan untuk mendefinisikan aturan duduksi untuk menyimpulkan informasi baru dari fakta database yang disimpan. Sistem seperti ini disebut sistem Database deduktif. Fungsionalitas yang lebih kuat disediakan oleh sistem database aktif, yang menyediakan aturan aktif yang dapat secara otomatis memulai tindakan ketika peristiwa dan kondisi tertentu terjadi.
15. ***Ketersediaan informasi terkini untuk semua pengguna:*** DBMS membuat database tersedia untuk semua pengguna. Segera setelah pembaruan satu pengguna diterapkan ke database, semua pengguna lain dapat segera melihat pembaruan ini. Ketersediaan informasi terkini ini sangat penting untuk banyak aplikasi pemrosesan transaksi, seperti sistem reservasi, database perbankan, dan dimungkinkan oleh subsistem kontrol konkurensi dan pemulihan DBMS.
16. ***Fleksibilitas:*** Mungkin perlu untuk mengubah struktur database sebagai perubahan persyaratan. DBMS modern memungkinkan jenis perubahan evolusioner tertentu pada struktur database tanpa mempengaruhi data yang disimpan dan programaplikasi yang ada.
17. ***Peningkatan konkurensi:*** DBMS mengelola akses database konkuren dan mencegah masalah hilangnya informasi atau hilangnya integritas.
18. ***Keseimbangan persyaratan yang saling bertentangan*** : DBA menyelesaikan persyaratan yang saling bertentangan dari berbagai pengguna dan aplikasi. DBA dapat menyusun sistem untuk menyediakan layanan keseluruhan yang terbaik bagi organisasi. DBA dapat memilih struktur file dan metode akses terbaik untuk mendapatkan kinerja optimal untuk operasi respons-kritis, sementara mengizinkan aplikasi yang kurang penting untuk terus menggunakan database.

Kekurangan DBMS

Berikut ini kelemahan DBMS.

1. ***Kompleksitas Pencadangan dan Pemulihan*** : Agar Database bersama terpusat menjadi akurat dan tersedia setiap saat, prosedur yang komprehensif diperlukan untuk dikembangkan dan digunakan untuk menyediakan salinan cadangan data dan untuk memulihkan Database ketika terjadi kerusakan. DBMS modern biasanya mengotomatiskan lebih banyak tugas pencadangan dan pemulihan daripada sistem berorientasi file.
2. ***Peningkatan biaya instalasi dan manajemen:*** Perangkat lunak DBMS yang besar dan kompleks memiliki biaya awal yang tinggi. Ini membutuhkan tenaga terlatih untuk menginstal dan mengoperasikan dan juga memiliki biaya pemeliharaan dan dukungan tahunan yang besar. Perangkat lunak Database tambahan mungkin diperlukan untuk memberikan keamanan dan untuk memastikan pemutakhiran data bersama yang tepat secara bersamaan.
3. ***Biaya perangkat keras tambahan:*** Biaya instalasi DBMS sangat bervariasi, tergantung pada lingkungan dan fungsionalitas, ukuran perangkat keras dan biaya pemeliharaan perangkat keras dan perangkat lunak tahunan yang berulang.
4. ***Kebutuhan tenaga kerja baru dan khusus*** : Karena perubahan yang cepat dalam teknologi database dan kebutuhan bisnis organisasi, kebutuhan organisasi untuk mempekerjakan, melatih kembali tenaganya secara teratur untuk merancang dan mengimplementasikan database, menyediakan layanan administrasi database dan mengelola staf orang baru. Oleh karena itu, organisasi perlu mempertahankan tenaga terampil khusus.
5. ***Peningkatan kompleksitas:*** DBMS multi-pengguna menjadi perangkat lunak yang sangat kompleks karena fungsionalitas yang diharapkan darinya. Menjadi penting bagi perancang Database, pengembang, administrator Database, dan pengguna akhir untuk memahami fungsionalitas ini secara maksimal.
6. ***Masalah yang terkait dengan sentralisasi***: Sentralisasi berarti bahwa data dapat diakses dari satu sumber yang disebut database.
7. ***Ukuran DBMS yang besar:*** Kompleksitas yang besar dan fungsionalitas yang luas membuat DBMS menjadi perangkat lunak yang sangat besar. Ini menempati banyak gigabyte ruang disk penyimpanan dan membutuhkan sejumlah besar memori utama untuk berjalan secara efisien.

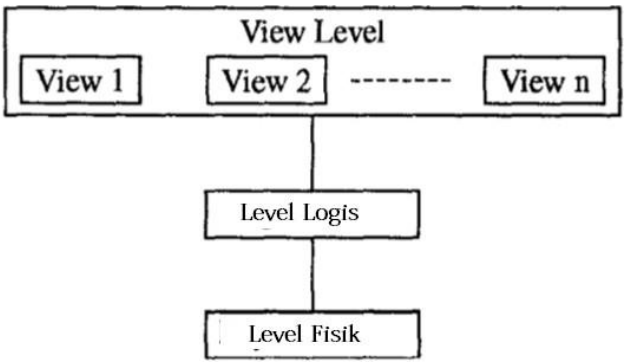
2.6 ABSTRAKSI DATA

Ada tiga tingkat abstraksi data.

1. **Level Fisik:** Level fisik dari abstraksi data menggambarkan bagaimana datasebenarnya disimpan.
2. **Level Logis** : Level logis dari abstraksi data menggambarkan data "apa" yang disimpandalam database dan hubungan apa yang ada di antara data tersebut.

Jadi tingkat logis menggambarkan seluruh database.

Administrator Database, yang harus memutuskan informasi apa yang akan disimpandalam Database, menggunakan tingkat abstraksi logis.



Gambar 2.2 Tingkat Abstraksi Data

3. **Level tampilan:** Level tampilan abstraksi data hanya menjelaskan sebagian darikeseluruhan database. Tingkat tampilan abstraksi menyederhanakan interaksi mereka dengan sistem. Sistem dapat menyediakan banyak tampilan untuk databaseyang sama.

***View of Data*:** Sebuah sistem database adalah kumpulan file terintegrasi dan satu set program yang memungkinkan pengguna untuk mengakses dan memodifikasi file-file ini.

2.7 INSTANCES DAN SKEMA

Instances: Kumpulan informasi yang disimpan dalam database pada saat tertentu disebut instance dari database.

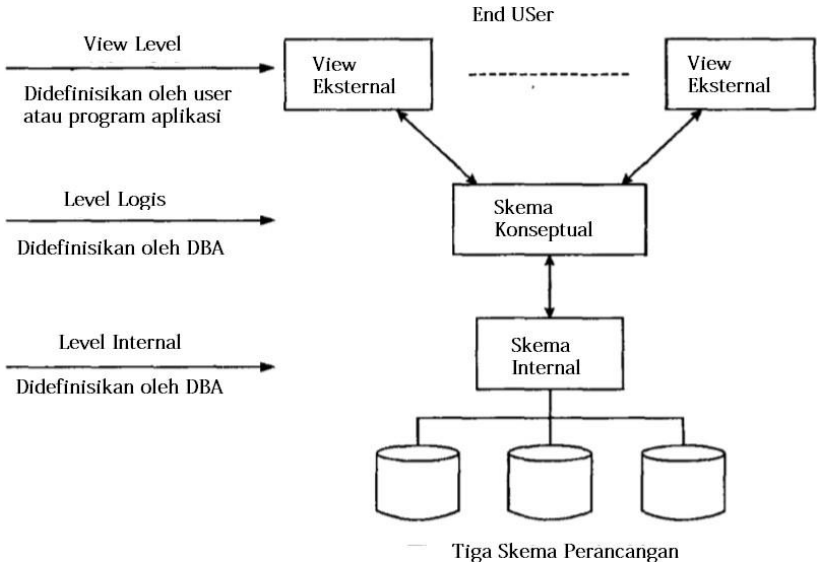
Skema: Desain keseluruhan database disebut skema database. Artinya, deskripsi database disebut skema database yang ditentukan selama desain database dan diharapkan tidak sering berubah. Skema database dapat dipartisi sesuai dengan tingkatabstraksi.

a. **Skema Fisik atau Internal:** Skema fisik, menjelaskan struktur penyimpanan fisik database. Tingkat internal berkaitan dengan kegiatan berikut:
   1. Alokasi ruang penyimpanan untuk data dan penyimpanan.

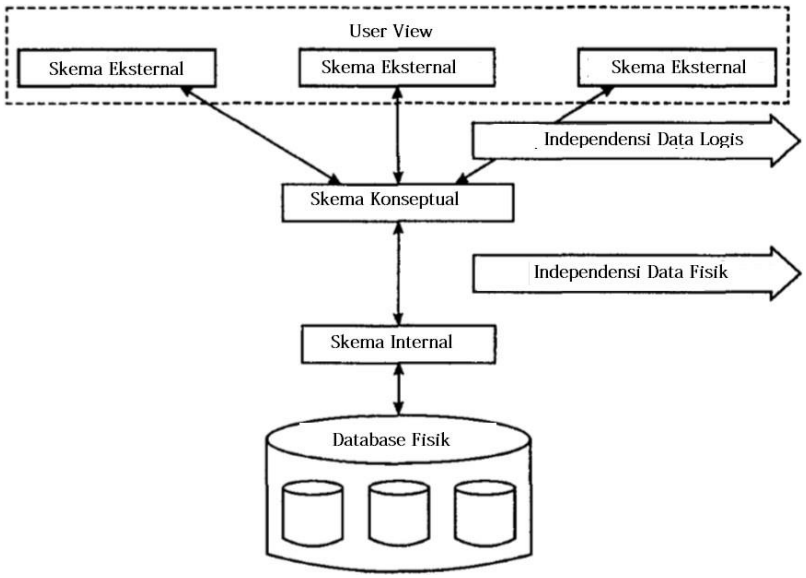
2. Rekam deskripsi untuk penyimpanan dengan ukuran yang disimpan untuk itemdata.
3. Penempatan Rekam.
4. Teknik kompresi data dan enkripsi data.

Skema ini menggunakan model data fisik dan menjelaskan detail lengkappenyimpanan data dan jalur akses untuk database.

b. **Skema konseptual atau logis:** Skema logis menggambarkan struktur seluruh database untuk komunitas pengguna. Skema konseptual menggambarkan entitas, tipe data, hubungan, operasi pengguna, dan batasan serta menyembunyikan detail struktur penyimpanan fisik.



Gambar 2.3 Skema Database



Gambar 2.4 Skema Database

Tingkat konseptual berkaitan dengan aktivitas berikut:

1. Semua entitas, atributnya, dan hubungannya.
2. Kendala pada data.
3. Informasi semantik tentang data.
4. Informasi keamanan
5. Pemeriksaan untuk menjaga konsistensi dan integritas data.

c. **Skema Eksternal:** Setiap skema eksternal menjelaskan bagian dari database yang diminati oleh kelompok pengguna tertentu dan menyembunyikan sisa database darikelompok pengguna tersebut.

2.8 INDEPENDENSI DATA

Kemampuan untuk mengubah skema pada satu tingkat sistem Database tanpa harus mengubah skema pada tingkat berikutnya yang lebih tinggi disebut "Independensi Data". Ada dua jenis independensi data:

1. **Independensi Data Fisik:** Independensi data fisik adalah kemampuan untuk mengubah skema internal tanpa harus mengubah skema konseptual. *Contoh* : Dengan membuat struktur akses tambahan untuk

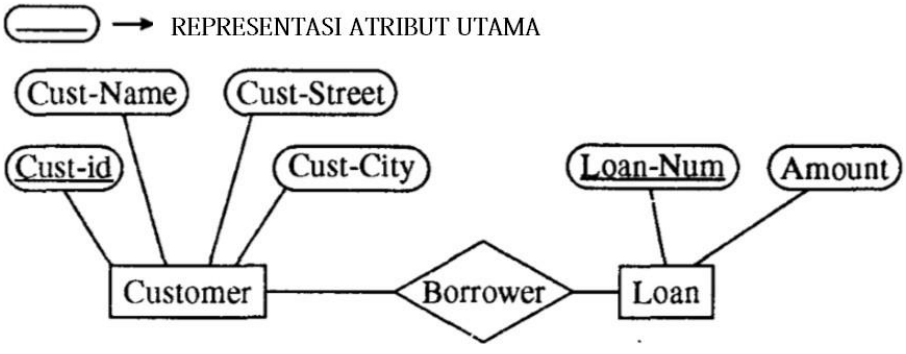
meningkatkan kinerja pengambilan atau pembaruan.

2. **Logical Data Independence :** Logical Data Independence adalah kemampuan untuk mengubah skema konseptual tanpa harus mengubah program aplikasi (skema eksternal). *Contoh* : Kita dapat mengubah skema konseptual untuk memperluas database dengan menambahkan tipe record atau item data. ATAU untuk mengurangi database dengan menghapus item data.

Model Entity-Relationship (E-R)

Model data E-R didasarkan pada persepsi dunia nyata yang terdiri dari kumpulan objek dasar yang disebut entitas dan hubungan di antara objek-objek tersebut. Keseluruhan struktur logika database dapat direpresentasikan secara grafis dengan diagram ER. DiagramE-R dibangun dari komponen-komponen berikut.

1. **Rectangle**: yang mewakili himpunan entitas
2. **Elips**: yang mewakili atribut
3. **Diamond**: yang mewakili hubungan antara himpunan entitas.
4. **Line**: yang menghubungkan atribut ke kumpulan entitas dan kumpulan entitas kehubungan.
5. **Double Ellips** : yang mewakili atribut multinilai.
6. **Dashed Ellipses** : yang menunjukkan atribut turunan.
7. **Double Line :** yang mewakili total partisipasi suatu entitas dalam himpunanhubungan. Dobel
8. **Rectangle** : yang mewakili himpunan entitas yang lemah.



Gambar 2.5 Representasi Atribut Utama

Kelebihan

1. **Representasi relasional lurus ke depan:** Setelah merancang diagram ER untuk aplikasi database, representasi relasional dari model database menjadi relatif lurus ke depan.
2. **Konversi Mudah untuk ER ke model data lain:** Konversi dari diagram ER ke jaringan atau model data hierarkis dapat dengan mudah dilakukan.
3. **Representasi grafis** untuk pemahaman yang lebih baik.

Kekurangan

1. **Tidak ada standar industri untuk notasi:** Tidak ada notasi standar industri untukmengembangkan diagram ER.
2. **Populer untuk desain tingkat tinggi:** Model data ER adalah desain database tingkattinggi.

Model Relasional

Model relasional menggunakan kumpulan tabel untuk mewakili data dan hubungan di antara data tersebut. Tabel adalah kumpulan baris dan kolom. Setiap kolom memiliki nama yang unik. Setiap baris dalam tabel mewakili kumpulan nilai data terkait. Dalam model relasional, baris disebut tuple, header kolom disebut atribut dan tabel disebut relasi.

Kelebihan Model Relasional

1. Kesederhanaan: Model data relasional bahkan lebih sederhana dari model hierarkis dan jaringan. Ini membebaskan desainer dari detail penyimpanan data fisik yang sebenarnya, sehingga memungkinkan mereka untuk berkonsentrasi pada tampilan logis dari database.
2. Independensi struktural : Model data relasional tidak bergantung pada sistem akses data navigasi. Perubahan struktur database tidak mempengaruhi akses data.
3. Kemudahan desain, implementasi, pemeliharaan, dan penggunaan.
4. Kemampuan kueri yang fleksibel dan kuat.

Kekurangan

1. Overhead perangkat keras: Model data relasional membutuhkan perangkat keras komputasi dan perangkat penyimpanan data yang lebih kuat untuk melakukan tugas yang ditetapkan RDBMS.
2. Mudah untuk merancang kemampuan yang mengarah ke desain yang buruk.

   Model Data Berorientasi Objek

   Model data berorientasi objek IS didasarkan pada paradigma bahasa pemrograman berorientasi objek. Paradigma berorientasi objek didasarkan pada Enkapsulasi data dan kode yang terkait dengan suatu objek menjadi satu kesatuan, pewarisan dan identitas objek.

   Contoh:

   Kelas karyawan

   {

   nama string; alamat string; int gaji;

   int gaji tahunan ( );

   };

Kelebihan Model Data Berorientasi Objek

1. Akses data yang ditingkatkan: Model data berorientasi objek mewakili hubungan secara eksplisit, mendukung akses navigasi dan asosiatif ke informasi. Ini meningkatkan kinerja akses data.
2. Peningkatan produktivitas: Ini menyediakan fitur-fitur canggih seperti pewarisan, polimorfisme, dan pengikatan dinamis yang memungkinkan pengguna untuk membuat objek dan memberikan solusi tanpa menulis kode khusus objek. Fitur ini meningkatkan produktivitas para pengembang aplikasi database secara signifikan.
3. Menggabungkan pemrograman berorientasi objek dengan teknologi database.
4. Mampu menangani berbagai macam tipe data.

Kekurangan

1. Tidak ada definisi yang tepat: Sulit untuk memberikan definisi yang tepat tentang apa yang terdapat pada DBMS berorientasi objek.

2. Sulit untuk dipertahankan : Definisi objek perlu diubah secara berkala dan migrasi database yang ada agar sesuai dengan definisi objek baru dengan perubahan kebutuhan informasi organisasi.
3. Tidak cocok untuk semua aplikasi: Model data berorientasi objek digunakan jika ada kebutuhan untuk mengelola hubungan kompleks di antara objek data.
4. Overhead perangkat keras: Model data relasional membutuhkan perangkat keras komputasi dan penyimpanan data yang lebih kuat.

Model Data Obyek-Relasional

Model data relasional objek, menggabungkan fitur model rasional dan berorientasi objek. Model ini menyediakan sistem tipe kaya database berorientasi objek, menggabungkan dengan hubungan sebagai dasar untuk penyimpanan data. Sistem Database relasional objek menyediakan jalur migrasi yang mulus bagi pengguna Database relasional yang ingin menggunakan fitur berorientasi objek.

Model Hirarki

Model hierarki menyediakan dua konsep penataan data utama.

- Catatan
- Hubungan orang tua-anak.

  Catatan : Catatan adalah kumpulan nilai bidang yang memberikan informasi tentang entitas atau instance hubungan. Relasi Parent-child : Tipe relasi parent-child adalah relasi 1 : N antara dua tipe record.
- Tipe record pada sisi 1 disebut tipe parent reecprd dan tipe record pada sisi N disebut tipe record anak tipe PCR
- Instance tipe PCR terdiri dari satu record dari tipe record induk dan sejumlah record (nol atau lebih) dari tipe record anak.

Keuntungan Model Data Hirarki

Berikut adalah keuntungan dari model data hierarkis.

1. Kesederhanaan: Hubungan antara berbagai lapisan secara logis sederhana dan desain Database hierarkis sederhana.
2. Berbagi data : Karena semua data disimpan dalam database yang sama, berbagi data menjadi praktis. .

3. Keamanan data: Ini adalah model database pertama yang menawarkan keamanan data yang disediakan dan diterapkan oleh DBMS.
4. Integritas data: Dalam hubungan induk/anak, selalu ada tautan antara segmen induk dan segmen turunannya di bawahnya.
5. Efficiency: Model ini sangat efisien ketika database berisi volume data yang besar dalam hubungan one-lo-many (1:m) dan ketika pengguna membutuhkan sejumlah besar transaksi.

Independensi data: DBMS menciptakan lingkungan di mana independensi data dapat dipertahankan.

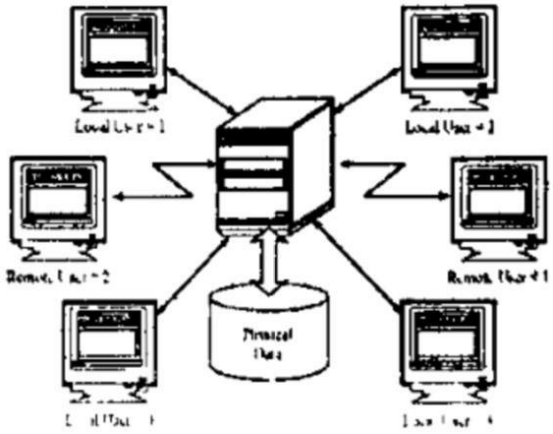
Kekurangan

1. Kompleksitas implementasi: Meskipun Database hierarkis secara konseptual sederhana, mudah dirancang dan tidak ada masalah ketergantungan data, namun cukup rumit untuk diterapkan.
2. Batasan implementasi : Banyak dari hubungan umum tidak mengkonfirmasi format hubungan satu-ke-banyak yang diperlukan oleh model Database hierarkis.
3. Tidak fleksibel: Database hierarkis tidak memiliki fleksibilitas. Perubahan dalam relasi atau segmen baru sering kali menghasilkan tugas manajemen sistem yang sangat kompleks.
4. Kurangnya independensi struktural.
5. Kompleksitas pemrograman aplikasi.

Model Data Jaringan

Data dalam model Jaringan mewakili kumpulan catatan dan hubungan antar data diwakili oleh tautan, yang dapat dilihat sebagai petunjuk. Catatan dalam database diatur sebagai kumpulan grafik arbitrer.

Sebagai contoh,



Gambar 2.6 Model Data Jaringan

Keuntungan Model Data Jaringan

- Kesederhanaan: Mirip dengan model data hierarkis, model jaringan juga sederhana dan mudah dirancang.
- Akses data superior: Akses dan fleksibilitas data lebih unggul daripada yang ditemukan dalam model data hierarkis.
- Memfasilitasi lebih banyak jenis hubungan : Model jaringan memfasilitasi dalam menangani hubungan satu-ke-banyak (1:m) dan banyak-ke-banyak (n:m), yang membantu dalam pemodelan situasi kehidupan nyata.
- Integritas Database: Model jaringan menegakkan integritas Database dan tidak mengizinkan anggota ada tanpa pemilik
- Independensi data: Model data jaringan menyediakan independensi data yang cukup dengan setidaknya mengisolasi sebagian program dari detail penyimpanan fisik yang kompleks.

Kekurangan

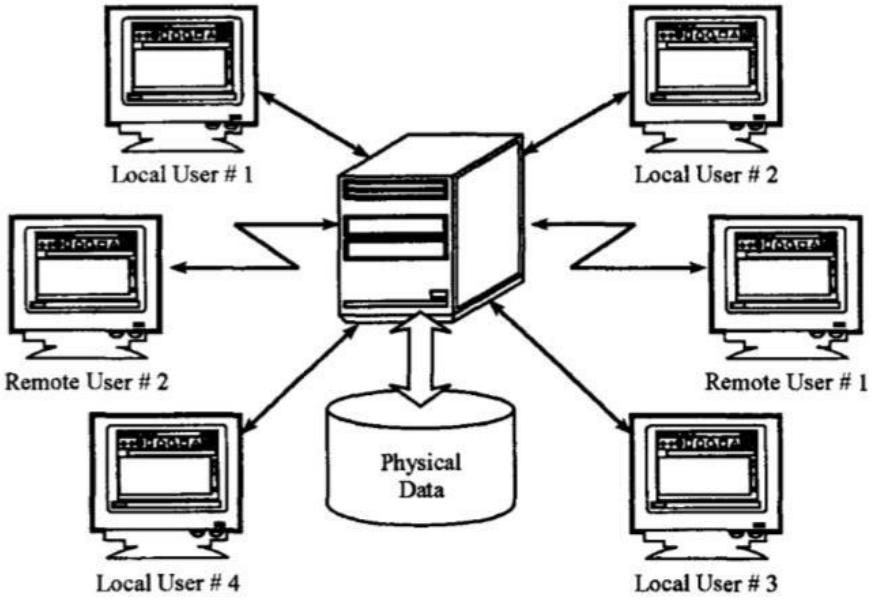
- Kompleksitas sistem: Karena model jaringan menyediakan mekanisme akses navigasi ke data di mana data mengakses satu catatan pada satu waktu. Mekanisme ini membuat implementasi sistem menjadi sangat kompleks.
- Tidak adanya independensi struktural: Sulit untuk membuat perubahan dalam database jaringan.
- Tidak ramah pengguna: Model data jaringan bukanlah desain untuk sistem yang mudah digunakan dan merupakan sistem yang sangat berorientasi pada keterampilan.

- Penggunaan pointer mengarah ke struktur yang kompleks, yang membuat pemetaan data terkait menjadi sangat sulit.

2.9 JENIS – JENIS SISTEM DATABASE

Database Manajemen Sistem dapat diklasifikasikan menurut jumlah pengguna, lokasi database antara lain :

1. Berdasarkan jumlah pengguna
   - Database Manajemen Sistem Tunggal
   - Database Manajemen Sistem Multi Pengguna
2. Berdasarkan Lokasi :
   - Database Manajemen Sistem Terpusat
   - Database Manajemen Sistem Terdistribusi
   - Database Manajemen Sistem paralel
   - Database Manajemen Sistem Client/Server



Gambar 2.7 Database Manajemen Sistem

Sistem Database Terpusat

Sistem database terpusat terdiri dari satu prosesor bersama dengan perangkat penyimpanan data terkait dan periferal lainnya. Secara fisik terbatas pada satu lokasi. Pengelolaan sistem dan datanya dikendalikan secara terpusat dari siapa pun atau situs pusat. Sebuah DBMS terpusat jika data disimpan di satu situs komputer. Sebuah DBMS

terpusat dapat mendukung banyak pengguna, tetapi DBMS dan database itu sendiri berada sepenuhnya di satu situs komputer.

Kelebihan Sistem Database Terpusat :

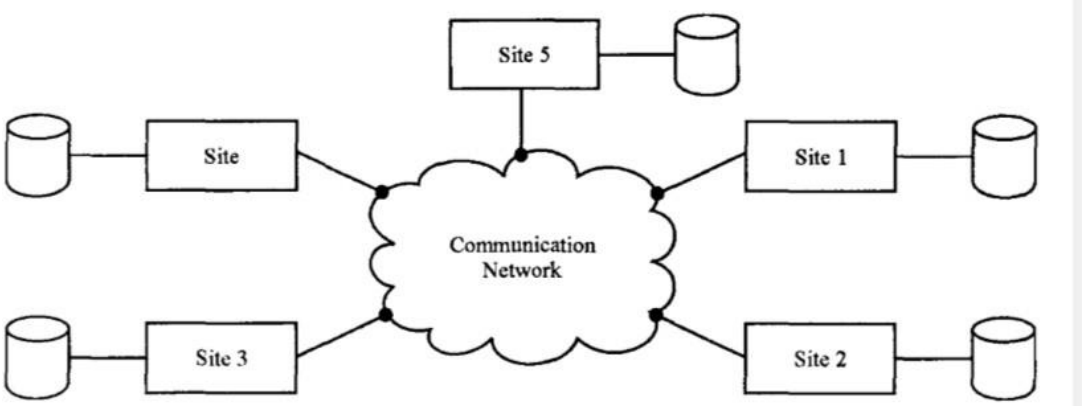
- Sebagian besar fungsi seperti pembaruan, pencadangan, kueri, akses kontrol, dan sebagainya, lebih mudah dilakukan dalam sistem Database terpusat.
- Ukuran Database dan komputer tempat ia berada tidak perlu menentukan apakah Database terletak di pusat.

Kekurangan Sistem Database Terpusat :

- Ketika komputer situs pusat atau sistem Database mati, maka setiap pengguna diblokir dari menggunakan sistem hingga sistem kembali.
- Biaya komunikasi dari terminal ke lokasi pusat bisa mahal.

Sistem Database Terdistribusi

Database terdistribusi adalah kumpulan beberapa database yang saling berhubungan secara logis yang didistribusikan melalui Jaringan Komputer. Sistem manajemen Database terdistribusi adalah sistem perangkat lunak yang mengelola Database terdistribusi sambil membuat distribusi transparan kepada pengguna. Dalam sistem database terdistribusi, data didistribusikan di berbagai database yang berbeda.



Gambar 2.8 Jaringan Sistem Database Terdistribusi

Sistem Database Terdistribusi ini dikelola oleh berbagai perangkat lunak DBMS yang berbeda yang berjalan pada berbagai mesin komputasi yang berbeda yang didukung oleh berbagai sistem operasi yang berbeda. Mesin-mesin ini didistribusikan secara geografis dan dihubungkan bersama oleh berbagai jaringan komunikasi. Dalam sistem

database terdistribusi, satu aplikasi dapat beroperasi pada data yang didistribusikan secara geografis pada mesin yang berbeda. Jadi, dalam sistem Database terdistribusi, data mungkin didistribusikan pada komputer yang berbeda sedemikian rupa sehingga data untuk satu bagian disimpan di satu komputer dan data untuk bagian lain disimpan di komputer lain. Setiap mesin dapat memiliki data dan aplikasinya sendiri. Namun, pengguna di satu komputer dapat mengakses data yang disimpan di serveral komputer lain. Oleh karena itu, setiap mesin akan bertindak sebagai server untuk beberapa pengguna dan klien untuk yang lain. Detil lebih lanjut tentang sistem database terdistribusi diberikan di Unit-So.

Keuntungan Sistem Database Terdistribusi

1. Pengelolaan data terdistribusi dengan tingkat transparansi yang berbeda.
2. Peningkatan Keandalan dan Ketersediaan.
3. Pemrosesan kueri terdistribusi.
4. Peningkatan kinerja.
5. Peningkatan skalabilitas
6. Evaluasi paralel.
7. Pemulihan Database terdistribusi.
8. Manajemen Data yang Direplikasi
9. Keamanan
10. Transparansi jaringan.
11. Ini memberikan efisiensi yang lebih besar dan kinerja yang lebih baik.
12. Transparansi replikasi.

Kekurangan Sistem Database Terdistribusi

1. Masalah teknis menghubungkan mesin yang berbeda.
2. Biaya dan kompleksitas perangkat lunak.
3. Kesulitan dalam kontrol integritas data.
4. Memproses overhead.
5. Kegagalan jaringan komunikasi.
6. Pemulihan dari kegagalan lebih kompleks.

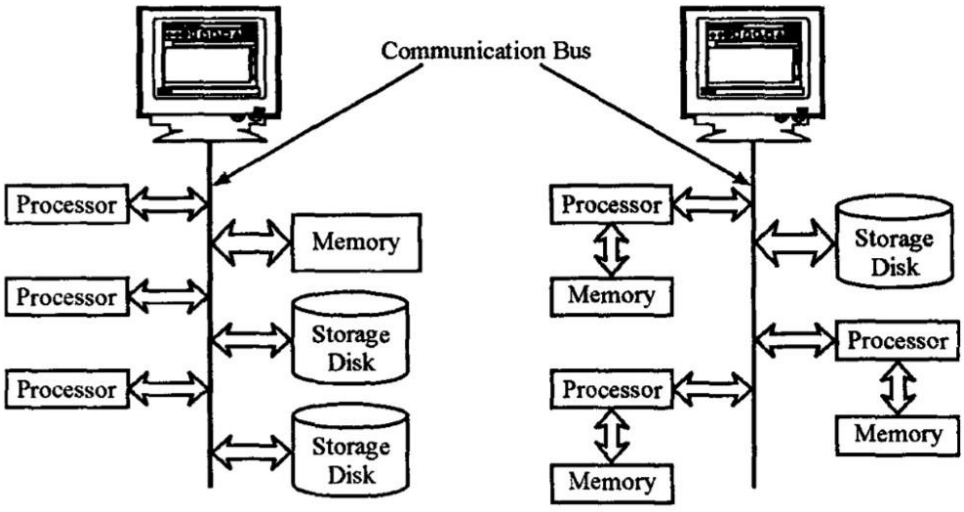
Sistem Database Paralel

Arsitektur sistem database paralel terdiri dari beberapa CPU dan disk penyimpanan data secara paralel. Oleh karena itu, mereka meningkatkan kecepatan pemrosesan dan input/output. Sistem Database paralel digunakan dalam aplikasi yang harus menanyakan Database yang sangat besar atau yang harus memproses transaksi dalam jumlah yang sangat besar per detik.

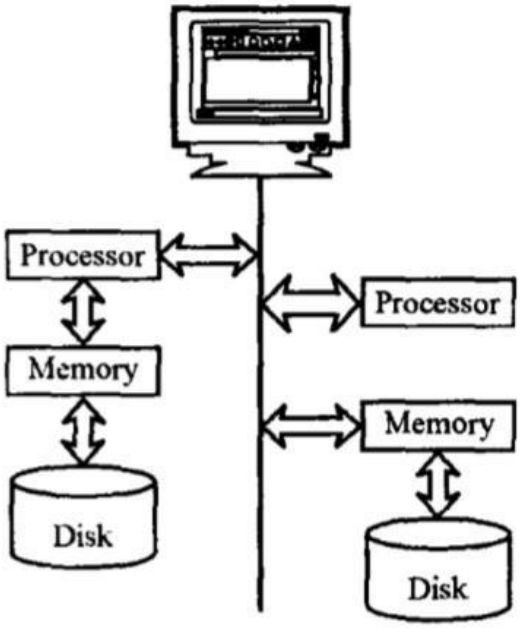
Beberapa arsitektur yang berbeda dapat digunakan untuk sistem database paralel, yaitu sebagai berikut:

- Memori Bersama : Semua prosesor berbagi memori yang sama.
- Disk penyimpanan data bersama : Semua prosesor berbagi satu set disk yang sama. Sistem disk bersama kadang-kadang disebut cluster.
- Sumber Daya Independen : Prosesor tidak berbagi memori atau disk umum.
- Hirarki : Model ini adalah hibrida dari tiga arsitektur sebelumnya.

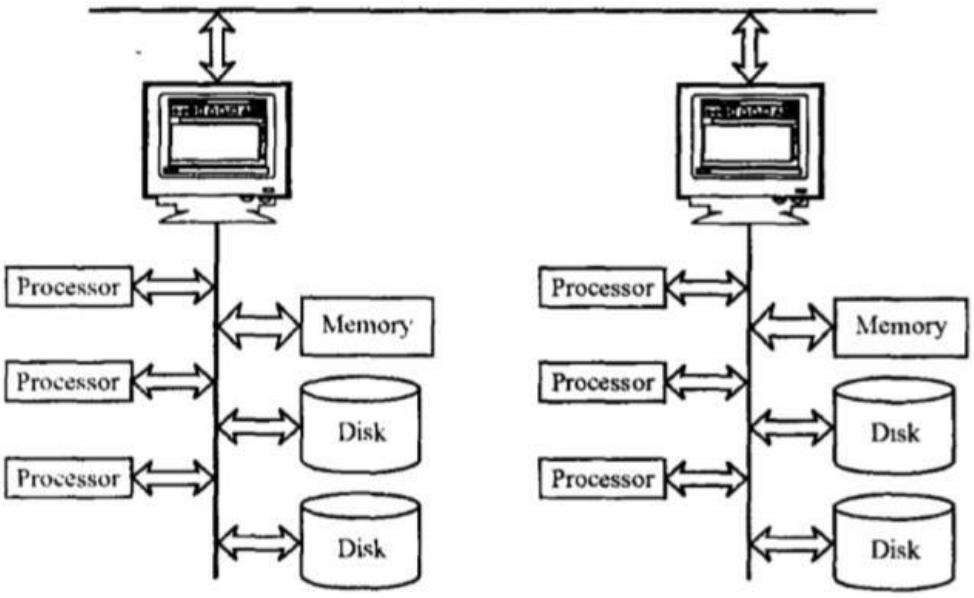
Gambar menggambarkan arsitektur yang berbeda dari sistem database paralel. Dalam disk penyimpanan data bersama, semua prosesor berbagi disk (atau kumpulan disk) yang sama, seperti yang ditunjukkan pada Gambar. (a). Dalam arsitektur memori bersama, semua prosesor berbagi memori yang sama, seperti yang ditunjukkan pada Gambar. (b). Dalam arsitektur sumber daya independen, prosesor tidak berbagi memori umum maupun disk umum. Mereka memiliki sumber daya independen mereka sendiri seperti yang ditunjukkan pada Gambar. (c). Arsitektur hierarkis adalah hibrida dari ketiga arsitektur sebelumnya, seperti yang ditunjukkan pada Gambar. (d)



Gambar 2.9 Database Paralel memori bersama



Gambar 2.10 Database Paralel Independen



Gambar 2.11 Database Paralel Hierarki

Keuntungan Sistem Database Paralel

- Sistem Database paralel sangat berguna untuk aplikasi yang harus meminta Database yang sangat besar atau yang harus memproses transaksi dalam jumlah yang sangat besar per detik.

- Teknik ini digunakan untuk mempercepat proses transaksi pada sistem data-server.
- Dalam sistem database paralel, throughput (yaitu, jumlah tugas yang dapat diselesaikan dalam interval waktu tertentu) dan waktu respons (yaitu, jumlah waktu yang diperlukan untuk menyelesaikan satu tugas dari waktu diajukan) sangat tinggi.

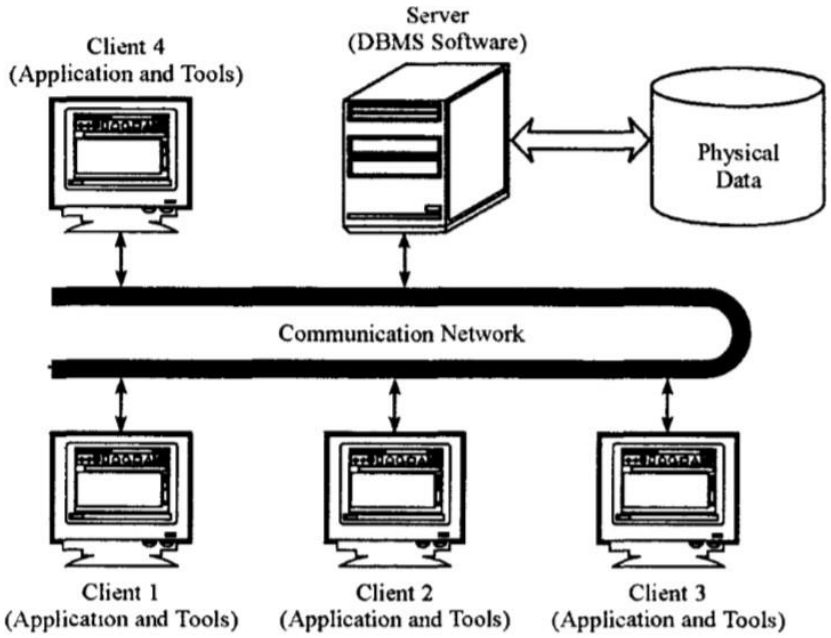
Kekurangan Sistem Database Paralel

- Dalam sistem database paralel, ada biaya startup yang terkait dengan memulai satu proses dan waktu startup dapat menutupi waktu pemrosesan yang sebenarnya, mempengaruhi percepatan secara merugikan.
- Karena proses yang dijalankan dalam sistem paralel sering mengakses sumber daya bersama, perlambatan dapat terjadi akibat interferensi dari setiap proses baru karena proses tersebut bersaing dengan proses yang ada untuk sumber daya yang dimiliki bersama, seperti disk penyimpanan data bersama, bus sistem, dan sebagainya.

Sistem Database Client/Server

Arsitektur Client/Server dari sistem database memiliki dua komponen logis yaitu client dan server. Client umumnya komputer pribadi atau workstation sedangkan server adalah workstation besar. Aplikasi dan alat DBMS berjalan pada satu atau lebih platform klien, sedangkan perangkat lunak DBMS berada di server. Komputer server disebut backend dan komputer klien disebut frontend. Komputer server dan klien ini terhubung melalui jaringan komputer. Aplikasi dan alat bertindak sebagai klien DBMS, membuat permintaan untuk layanannya.

Arsitektur client / server adalah bagian dari arsitektur sistem terbuka di mana semua perangkat keras komputasi, sistem operasi, protokol jaringan, dan perangkat lunak lainnya saling berhubungan sebagai jaringan dan bekerja bersama untuk mencapai tujuan pengguna. Ini sangat cocok untuk pemrosesan transaksi online dan aplikasi pendukung keputusan, yang cenderung menghasilkan sejumlah transaksi yang relatif singkat dan membutuhkan tingkat konkurensi yang tinggi.



Gambar 2.12 Model Database Client Server

Keuntungan Client Server

- Lingkungan Client/Server memfasilitasi pekerjaan yang lebih produktif oleh pengguna dan memanfaatkan data yang ada dengan lebih baik.
- Sistem Database Client/Server lebih fleksibel dibandingkan dengan sistem terpusat.
- Sebuah database tunggal (di server) dapat dibagi di beberapa sistem klien (aplikasi) yang berbeda.
- Arsitektur Client/Server memberikan kinerja DBMS yang lebih baik.
- Sistem Client/Server memiliki platform yang lebih murah untuk mendukung aplikasi yang sebelumnya hanya berjalan pada komputer mainframe.

Keuntungan Client Server

- Biaya tenaga kerja atau pemrograman tinggi di lingkungan Client/Server, terutama pada fase awal.
- Ada kekurangan alat manajemen untuk pemantauan kinerja dan penyetelan dan kontrol keamanan, untuk DBMS, klien dan sistem operasi dan lingkungan jaringan.

2.10 PEMAHAMAN BAHASA DATABASE

Bahasa definisi data digunakan untuk menentukan skema database. misalnya, Contoh DOL adalah membuat, mengubah, dan menjatuhkan tabel. Seperti Buat tabel siswa :

(Nomor guling (10), Nama char (15), Jenis kelamin char (2), Alamat varchar (15));

Eksekusi pernyataan DOL di atas membuat tabel siswa. Ini memperbarui satu set tabel khusus yang disebut 'Kamus Data'. Kamus data berisi meta data artinya data tentang data. Skema (desain) sebuah tabel adalah contoh dari meta data.

Contoh :

- CREATE : Untuk membuat objek dalam database.
- ALTER: Mengubah struktur database.
- DROP: Menghapus objek dari data.
- TRUNCATE: Hapus semua catatan dari tabel, termasuk semua ruang yang dialokasikan untuk catatan dihapus
- COMMENT: Menambahkan komentar ke kamus data.

Data-Manipulation Language (DMl)

Data Manipulation Berarti :

- Pengambilan data dari database.
- Penghapusan data dari database
- Penyisipan data baru ke dalam database
- Modifikasi data dalam database.

Data Manipulation Language (DML adalah bahasa yang memungkinkan pengguna untuk mengakses atau memodifikasi data dari database. Artinya DML digunakan untuk mengakses atau memanipulasi data dari database. DML pada dasarnya adalah dua jenis:

- DML Prosedural
- DML non prosedural

DML Prosedural : DML prosedural mengharuskan pengguna untuk menentukan Data apa yang dibutuhkan dan bagaimana cara mendapatkan data tersebut. misalnya, PL/SOL. DML Non Prosedural : DML non prosedural mengharuskan pengguna untuk menentukan

data apa yang dibutuhkan tanpa menentukan cara mendapatkan data tersebut. misalnya, SQL

Contoh: Contoh DML (NDML)

- Pilih Gulungan, Nama, Alamat dari Siswa Dimana Gulungan = 3;
- Pilih * dari siswa: Misalnya,
- INSERT: Menyisipkan data ke dalam tabel
- UPDATE: Memperbarui data yang ada dalam tabel
- DELETE: Menghapus semua record dari tabel, ruang untuk record tetap ada.
- LOCK TABLE: Kontrol konkurensi.

Data Control Language (DCl)

Ini adalah komponen pernyataan SQL yang mengontrol akses ke data dan ke database. Sebagai contoh,

- COMMIT: Simpan pekerjaan yang sudah selesai.
- ROLL-BACK: Kembalikan database ke aslinya sejak komit terakhir.
- SAVE POINT: Identifikasi titik dalam transaksi yang nantinya dapat Anda putar kembali
- GRANTT/REVOKE: Memberikan atau mengambil kembali izin ke atau dari pengguna oracle.
- SET TRANSACTION: Ubah atau ambil kembali izin ke atau dari pengguna oracle.

Data Query Language (DQL)

Ini adalah komponen pernyataan SQL yang memungkinkan mendapatkan data dari database dan memaksakan pemesanan padanya. SQL:

- DDL (membuat tabel)
- DML (memanipulasi tabel)

Catatan: DDL dan DML bukan dua bahasa yang berbeda, ini adalah bagian dari SQL.

Antarmuka yang mudah digunakan yang disediakan oleh DBMS dapat mencakup hal- hal berikut:

- Antarmuka Berbasis Menu untuk Penjelajahan: Mereka sering digunakan dalam antarmuka penjelajahan, yang memungkinkan pengguna untuk melihat isi database dengan cara yang eksploratif dan tidak terstruktur.

- Antarmuka Berbasis Formulir: Antarmuka berbasis formulir menampilkan formulir untuk setiap pengguna.
- Antarmuka Pengguna Grafis: Antarmuka grafis menampilkan skema kepada pengguna dalam bentuk diagram.
- Antarmuka Bahasa Alami: Ini memiliki "skema" sendiri yang mirip dengan skema konseptual Database.
- Antarmuka untuk DBA
- Antarmuka untuk Pengguna

DATABASE USER DAN ADMINISTRATOR

Orang-orang yang bekerja dengan database dapat dikategorikan sebagai administrator database dan pengguna database.

Database Administrator (DBA)

Seseorang yang memiliki kendali pusat baik data maupun program yang mengakses data tersebut melalui sistem disebut administrator Database.

Fungsi DBA

Berikut ini adalah fungsi-fungsi DBA.

1. Mendefinisikan Skema Konseptual: DBA membuat skema konseptual (menggunakan bahasa definisi data) yang sesuai dengan desain database tingkat abstrak yang dibuat oleh administrator data. DBA membuat skema database asli dan struktur database.
2. Mendefinisikan Skema Fisik: DBA membuat skema fisik (menggunakan DDL) yang sesuai dengan desain database tingkat abstrak yang dibuat oleh administrator data.
3. Skema dan Modifikasi Organisasi Fisik: DBA melakukan perubahan atau modifikasi deskripsi database atau hubungannya dengan organisasi fisik database untuk mencerminkan perubahan kebutuhan organisasi atau untuk mengubah organisasi fisik untuk meningkatkan kinerja
4. Pemberian Otorisasi untuk Akses Data : DBA memberikan berbagai jenis otorisasi untuk menggunakan database kepada penggunanya. Ini mengatur penggunaan bagian-bagian tertentu dari database oleh berbagai pengguna. Informasi otorisasi disimpan dalam struktur sistem khusus yang dikonsultasikan oleh sistem Database setiap kali seseorang mencoba

mengakses data dalam sistem. DBA membantu pengguna dengan definisi masalah dan resolusinya.

5. Struktur Penyimpanan dan Metode Akses Definisi: DBA memutuskan bagaimana data akan direpresentasikan dalam database yang disimpan, proses yang disebut desain database fisik. Database administrator mendefinisikan struktur penyimpanan database menggunakan bahasa definisi data dan metode akses data dari database.
6. Pemeliharaan Rutin: DBA memelihara backup database secara berkala, baik ke hard disk, compact disk atau ke server jarak jauh, untuk mencegah hilangnya data jika terjadi bencana. Ini memastikan bahwa ruang penyimpanan kosong yang cukup tersedia untuk operasi normal dan meningkatkan ruang disk sesuai kebutuhan. DBA juga bertanggung jawab untuk memperbaiki kerusakan database karena misue atau kegagalan perangkat lunak dan perangkat keras. DBA mendefinisikan dan menerapkan mekanisme kontrol kerusakan yang sesuai yang melibatkan pembongkaran Database secara berkala ke perangkat penyimpanan cadangan dan memuat ulang Database dari dump terbaru kapan pun diperlukan.
7. Ketersediaan, Pencadangan, dan Pemulihan: Tugas terpenting DBA adalah ketersediaan, pencadangan, dan pemulihan data. Berdasarkan nilai yang ditempatkan pada data elektronik, database harus dilindungi dari segala bentuk kegagalan seperti perangkat keras, perangkat lunak dan manusia. DBA mempertahankan informasi tentang kebutuhan organisasi agar berhasil. Availability artinya data harus tersedia bagi semua orang yang membutuhkannya pada saat mereka membutuhkannya. Jika data tidak tersedia, bisnis berhenti berfungsi. Karena tidak semua kegagalan dapat diprediksi, DBA perlu menerapkan prosedur pemulihan yang akan mengurangi waktu henti yang terkait dengan kegagalan.
8. Performance Monitoring and Thning: DBA harus memastikan database cepat dan responsif. Database respons lambat biasanya menunjukkan kinerja sistem yang buruk, ada sesuatu yang salah di suatu tempat. DBA memantau keadaan database untuk kinerja opsional dan log kesalahan atau log peristiwa juga dipantau untuk kesalahan database. Database yang disetel dengan buruk membuat frustrasi untuk digunakan - mereka cenderung menambah lebih

banyak tekanan daripada nilai. Pemantauan sangat penting untuk mengakses status database dan menyesuaikannya.

9. Implementasi dan Desain Database: Tugas penting DBA adalah merancang Database untuk kinerja, skalabilitas, fleksibilitas, dan keandalan yang maksimal. Sebuah database yang dirancang dan diimplementasikan dengan baik membenarkan investasi database. DBA bertanggung jawab untuk menginstal DBMS baru dan memutakhirkan DBMS yang ada. DBA harus fasih dengan masalah instalasi dan peningkatan, yaitu masalah, persyaratan, dll.
10. Kontrol migrasi program, perubahan Database, referensi perubahan Database, dan perubahan menu melalui siklus hidup pengembangan.
11. Membuat dan memelihara semua database: yang diperlukan untuk pengembangan, pengujian, dan penggunaan produksi.
12. DBA memberlakukan dan memelihara batasan untuk memastikan integritas database.

Keterampilan DBA

DBA harus memiliki keterampilan berikut:

1. Pengetahuan yang baik tentang sistem operasi.
2. Pengetahuan yang baik tentang desain Database fisik.
3. Kemampuan untuk melakukan baik database dan juga pemantauan kinerja sistem operasi dan penyesuaian yang diperlukan.
4. Mampu memberikan arahan database strategis bagi organisasi.
5. Pengetahuan yang sangat baik tentang cadangan Database dan skenario pemulihan.
6. Keterampilan yang baik dalam semua alat database.
7. Pengetahuan yang baik tentang manajemen keamanan Database.
8. Pengetahuan yang baik tentang bagaimana database memperoleh dan mengelola sumber daya.
9. Pengetahuan yang baik tentang aplikasi di situs Anda.
10. Pengalaman dan pengetahuan dalam memigrasikan kode, perubahan Database, data, dan menu melalui berbagai tahap siklus hidup pengembangan.

11. Pengetahuan yang baik tentang cara database menegakkan integritas data.
12. Pengetahuan yang baik tentang penyetelan kinerja Database dan kode program.
13. DBA harus memiliki keterampilan komunikasi yang baik dengan manajemen, tim pengembangan, vendor, administrator sistem, dan penyedia layanan terkait lainnya.

User Database

Ada empat jenis pengguna sistem Database yang berbeda.

1. Pengguna Pemula : Pengguna pemula adalah pengguna tidak canggih yang berinteraksi dengan sistem dengan menjalankan salah satu program aplikasi yang telah ditulis sebelumnya. misalnya, Teller Bank yang perlu mentransfer Rp. 500 dari rekening A ke rekening B menjalankan program yang disebut transfer. Program ini meminta teller untuk jumlah uang yang akan ditransfer. Seorang pengguna ATM termasuk dalam kategori ini.
2. Pemrogram Aplikasi : Pemrogram aplikasi adalah profesional komputer yang menulis program aplikasi. Pemrogram aplikasi dapat memilih dari banyak alat untuk mengembangkan antarmuka pengguna. Contoh: Alat Rapid Application Development (RAD) adalah alat yang memungkinkan pemrogram aplikasi untuk membuat formulir dan laporan tanpa menulis program.
3. Pengguna Canggih **:** Pengguna canggih berinteraksi dengan sistem tanpa menulis program. Sebagai gantinya, mereka membentuk permintaan mereka dalam bahasa kueri database. Mereka mengirimkan setiap kueri tersebut ke prosesor kueri, yang fungsinya untuk memecah pernyataan DML menjadi instruksi yang dipahami oleh manajer penyimpanan. Analis yang mengirimkan kueri untuk mengeksplorasi data dalam database termasuk dalam kategori ini. (4) Pengguna khusus: Pengguna khusus adalah pengguna canggih yang menulis aplikasi database khusus yang tidak sesuai dengan kerangka pemrosesan data tradisional.

STRUKTUR DATABASE

Sistem Database dibagi menjadi modul-modul yang menangani masing-masing tanggung jawab sistem secara keseluruhan. Beberapa fungsi sistem database mungkin disediakan oleh sistem operasi komputer. Komponen fungsional dari sistem database dapat dibagi menjadi:

- Manajer penyimpanan
- Pemroses kueri

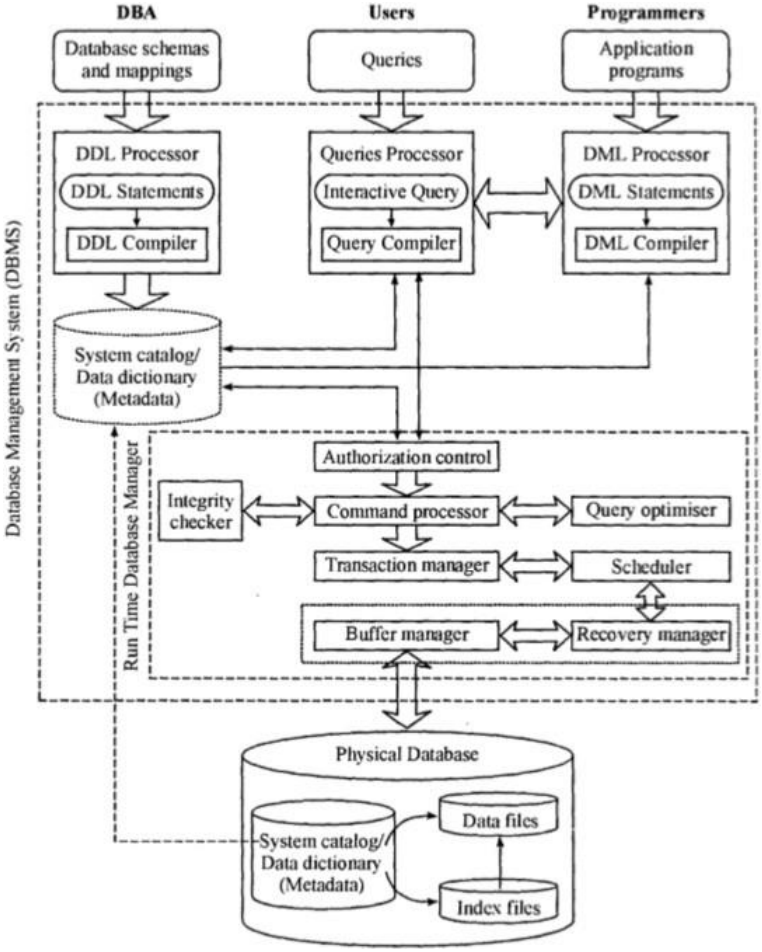
Manajer Penyimpanan

Manajer penyimpanan adalah modul program yang menyediakan antarmuka antara data tingkat rendah yang disimpan dalam database dan program aplikasi dan kueri yang dikirimkan ke sistem.

- Manajer penyimpanan bertanggung jawab atas interaksi dengan manajer pengisian.
- Manajer penyimpanan penting karena Database membutuhkan ruang penyimpanan yang besar.

Komponen berikut busur ini termasuk dalam manajer penyimpanan.

- Manajer Otorisasi dan Integritas: Ini menguji kepuasan batasan integritas dan memeriksa otoritas pengguna untuk mengakses data.
- Manajer Transaksi: Ini memastikan bahwa database tetap dalam keadaan konsisten meskipun ada kegagalan sistem dan eksekusi transaksi saat ini tanpa konflik.
- File Manager: Ini mengelola alokasi ruang pada penyimpanan disk dan struktur data yang digunakan untuk mewakili informasi yang disimpan pada disk.
- Buffer Manager: Bertanggung jawab untuk mengambil data dari penyimpanan disk ke memori utama dan memutuskan data apa yang akan diambil di memori utama.



Gambar 2.13 Struktur Database Manajemen Sistem

Berikut ini adalah struktur data yang diperlukan sebagai bagian dari implementasi sistem fisik.

1. File Data : Ini menyimpan database itu sendiri.
2. Kamus Data: Ini menyimpan metadata tentang struktur database, khususnya, skema database.
3. Indeks: Ini menyediakan akses cepat ke item data yang memiliki nilai tertentu
4. Data Statistik : Ini menyimpan informasi statistik tentang data dalam database

Pemroses Kueri Berikut adalah komponen-komponen berikut:

1. Kompilator DML: Ini menerjemahkan pernyataan DML ke dalam bahasa kueri ke dalam instruksi tingkat rendah yang dipahami oleh mesin evaluasi kueri.
2. Precompiler DML Tertanam: Ini mengubah pernyataan DML yang disematkan dalam program aplikasi ke panggilan prosedur normal dalam bahasa host.

3. Precompiler berinteraksi dengan compiler DML untuk menghasilkan kode yang sesuai.
4. DDL Interpreter : Ini menafsirkan pernyataan DDL dan mencatat definisi dalam kamus data.
5. Mesin Evaluasi Kueri: yang mengeksekusi instruksi tingkat rendah yang dihasilkan oleh kompiler DML.

Bahasa Generasi Ke-empat (4GL)

4GL adalah bahasa pemrograman yang ringkas, efisien dan non-prosedural yang digunakan untuk meningkatkan produktivitas DBMS. Dalam 4GL, pengguna menentukan apa yang harus dilakukan dan bukan bagaimana melakukannya. 4GL bergantung pada alat 4GL tingkat yang lebih tinggi, yang digunakan oleh pengguna untuk menentukan parameter guna menghasilkan program aplikasi. 4GL memiliki komponen-komponen berikut di dalamnya:

- Bahasa kueri
- Pembuat laporan
- Spreadsheet
- Bahasa Database
- Generator aplikasi untuk operasi deflne seperti menyisipkan, mengambil dan memperbarui data dari database untuk membangun aplikasi.
- Bahasa tingkat tinggi untuk menghasilkan program aplikasi.
- Structured Query Language (SQL) dan query by example (QBE) adalah contoh dari 4GL.

Konsep *Entity Relationship*

Diagram ER dapat mengekspresikan keseluruhan struktur logis dari database secara grafis. Ini adalah elemen-elemen berikut yang terdiri dari model E-R.

1. Himpunan entitas
2. Himpunan relasi
3. Atribut

Entitas

Entitas adalah "benda" atau objek di dunia nyata yang dapat dibedakan dari objek lain. misalnya, Setiap orang adalah suatu entitas dan rekening bank dapat dianggap sebagai entitas. Himpunan Entitas : Himpunan entitas adalah sekumpulan entitas

dengan tipe yang sama yang memiliki properti atau atribut yang sama. misalnya, Himpunan" dari semua orang yang menjadi pelanggan di bank tertentu. Itu adalah pelanggan adalah himpunan entitas Atribut

Setiap entitas diwakili oleh satu set properti yang disebut atribut. Atribut Kunci : Tipe entitas biasanya memiliki atribut yang nilainya berbeda untuk setiap entitas individu dalam koleksi. Atribut seperti itu disebut atribut kunci dan nilainya dapat digunakan untuk mengidentifikasi setiap entitas secara unik.

Jenis Atribut

Atribut Sederhana : Atribut yang tidak dapat dibagi menjadi sub bagian disebut atribut sederhana. misalnya, nomor Roll adalah contoh atribut sederhana.

Atribut Komposit : Atribut yang dapat dibagi menjadi pendukung disebut atribut komposit. misalnya, Tanggal Lahir adalah contoh atribut komposit, karena dapat dibagi menjadi "Hari lahir", "Bulan lahir" dan "Tahun lahir". OR Name juga merupakan contoh dari atribut komposit.

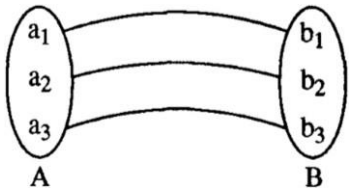
- Atribut Bernilai Tunggal : Atribut yang dapat mengasumsikan satu dan hanya satu nilai disebut "Atribut bernilai tunggal". Misalnya: ''Usia seseorang adalah contoh Atribusi bernilai tunggal. Atribut Bernilai Banyak: Atribut yang dapat mengasumsikan sekumpulan nilai disebut "Atribut Bernilai Banyak". misalnya, Entitas Karyawan yang ditetapkan dengan atribut nomor telepon adalah contoh atribut multinilai karena seorang karyawan mungkin memiliki nol, satu atau banyak nomor telepon.
- Atribut Null: Nilai nol digunakan ketika entitas tidak memiliki nilai untuk atribut. misalnya, Atribut gelar perguruan tinggi hanya berlaku untuk orang dengan gelar sarjana.
- Derived Attributes : Nilai dari tipe atribut ini dapat diturunkan dari atribut atau entitas lain yang terkait. misalnya, Dalam sistem penggajian, HRA dan PE diturunkan dari atribut gaji.

Pemetaan Kardinalitas

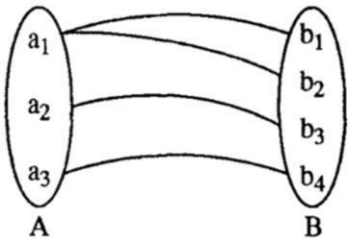
Kardinalitas pemetaan atau rasio kardinalitas, menyatakan jumlah entitas yang entitas lain dapat dikaitkan melalui hubungan st:1l: kardinalitas pemetaan paling berguna dalam menggambarkan himpunan hubungan biner. Untuk himpunan relasi

biner R antara himpunan entitas A dan B, kardinalitas pemetaan harus salah satu dari berikut ini :

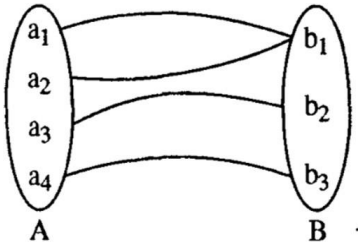
1. *One to One:* Entitas di A diasosiasikan dengan tepat satu entitas di B. Dan satu entitas di B diasosiasikan dengan tepat satu entitas di A.



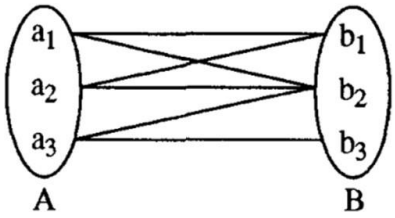
2. Satu ke banyak: Sebuah entitas di A diasosiasikan dengan sejumlah entitas di B. Sebuah entitas di B dapat diasosiasikan dengan paling banyak satu entitas di A.



3. Banyak ke Satu: Sebuah entitas di A dikaitkan dengan paling banyak satu entitas di B. Sebuah entitas di B, dapat dikaitkan dengan sejumlah entitas di A.



4. Banyak ke Banyak: Entitas di A dikaitkan dengan sejumlah entitas di B, dan entitas di B dikaitkan dengan sejumlah entitas di A.

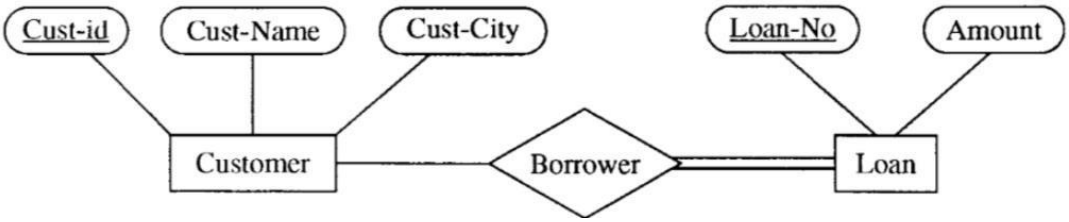


Batasan partisipasi menentukan apakah keberadaan himpunan entitas bergantung pada keterkaitannya dengan entitas lain

Batasan partisipasi:

1. Total
2. Sebagian

Total Participate : Partisipasi himpunan entitas E dalam himpunan relasi R dikatakan total jika setiap entitas pada E berpartisipasi dalam setidaknya satu hubungan dalam R. Contoh : Kita mengharapkan setiap entitas pinjaman berelasi dengan setidaknya satu pelanggan melalui hubungan peminjam. Oleh karena itu, partisipasi pinjaman dalam



himpunan hubungan peminjam adalah total.

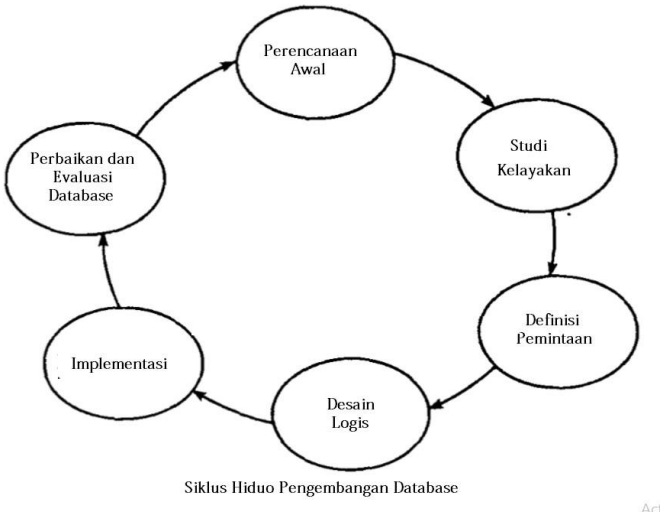
Gambar 2.14 Total Participal

Partiall Participate : Jika hanya beberapa entitas di E yang berpartisipasi dalam relasi di R. Partisipasi himpunan entitas E dalam relasi R dikatakan partisipasi parsial. Contoh: Seseorang dapat menjadi nasabah bank baik ia memiliki pinjaman di bank atau tidak. Oleh karena itu partisipasi nasabah dalam hubungan peminjam diatur secara parsial.

Tahapan *Database Development Life Cycle*

Database Development Life Cycle adalah proses untuk merancang, mengimplementasikan dan memelihara sistem Database. Ini terdiri dari enam tahap:

1. Desain awal
2. Desain kelayakan
3. Definisi persyaratan
4. Desain konseptual
5. Implementasi
6. Evaluasi dan pemeliharaan database.



Gambar 2.15 Siklus Hidup Pengembangan Database

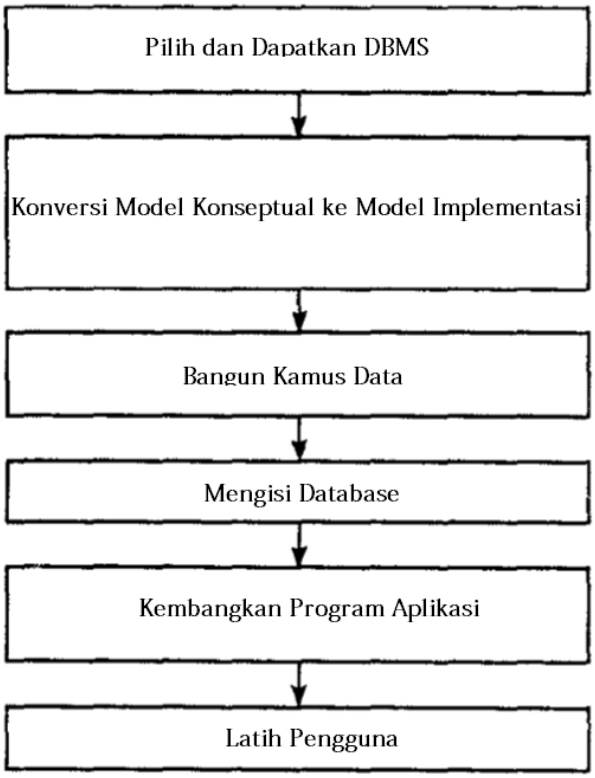
1. **Perencanaan awal** adalah sistem Database khusus yang terjadi selama proyek perencanaan Database strategis. Setelah proyek implementasi Database dimulai, model informasi umum yang dihasilkan selama perencanaan Database ditinjau dan ditingkatkan jika diperlukan. Selama proses ini, perusahaan mengumpulkan informasi untuk menjawab pertanyaan-pertanyaan berikut:
   a. Berapa banyak program aplikasi yang digunakan, dan fungsi apa yang mereka lakukan?
   b. File apa yang terkait dengan masing-masing aplikasi ini?
   c. Aplikasi dan file baru apa yang sedang dikembangkan?

   Informasi ini dapat digunakan untuk membangun hubungan antara aplikasi saat ini dan untuk mengidentifikasi penggunaan informasi aplikasi. Hal ini juga membantu untuk mengidentifikasi kebutuhan sistem masa depan dan untuk menilai manfaat ekonomi dari sistem database.
2. **Studi Kelayakan** melibatkan penyusunan laporan tentang isu-isu berikut:
   a. Kelayakan teknologi: Apakah teknologi yang sesuai tersedia untuk mendukung pengembangan database?
   b. Kelayakan operasional : Apakah perusahaan memiliki personel, anggaran dan keahlian internal untuk membuat sistem database berhasil?

c. Kelayakan ekonomi: Dapatkah manfaat diidentifikasi? Apakah sistem yang diinginkan akan menguntungkan dari segi biaya? Dapatkah biaya dan manfaat diukur?

3. **Definisi Persyaratan** Ini melibatkan pendefinisian ruang lingkup database yang mengidentifikasi manajemen dan persyaratan informasi area fungsional dan menetapkan persyaratan perangkat keras/lunak. Persyaratan informasi ditentukan dari tanggapan kuesioner, wawancara dengan manajer dan pengguna klerikal serta laporan dan formulir yang saat ini digunakan.
4. **Desain Konseptual** adalah tahap desain konseptual menciptakan skema konseptual untuk database. Spesifikasi dikembangkan ke titik di mana implementasi dapat dimulai. Selama tahap ini, model tampilan pengguna yang terperinci dibuat dan diintegrasikan ke dalam model data konseptual yang merekam semua elemen data perusahaan untuk dipelihara dalam database.
5. **Implementasi** yaitu Selama implementasi Database, DBMS dipilih dan diperoleh. Kemudian model konseptual rinci diubah menjadi model implementasi DBMS, kamus data dibangun, database terisi, program aplikasi dikembangkan dan pengguna dilatih.
6. **Evaluasi dan Pemeliharaan Database** melibatkan mewawancarai pengguna untuk menentukan apakah ada kebutuhan data yang tidak terpenuhi. Perubahan dilakukan sesuai kebutuhan. Seiring waktu sistem dipertahankan melalui pengenalan perangkat tambahan dan penambahan program baru dan elemen data sebagai kebutuhan bisnis berubah dan berkembang.

Arsitektur Database Tiga Tingkat.

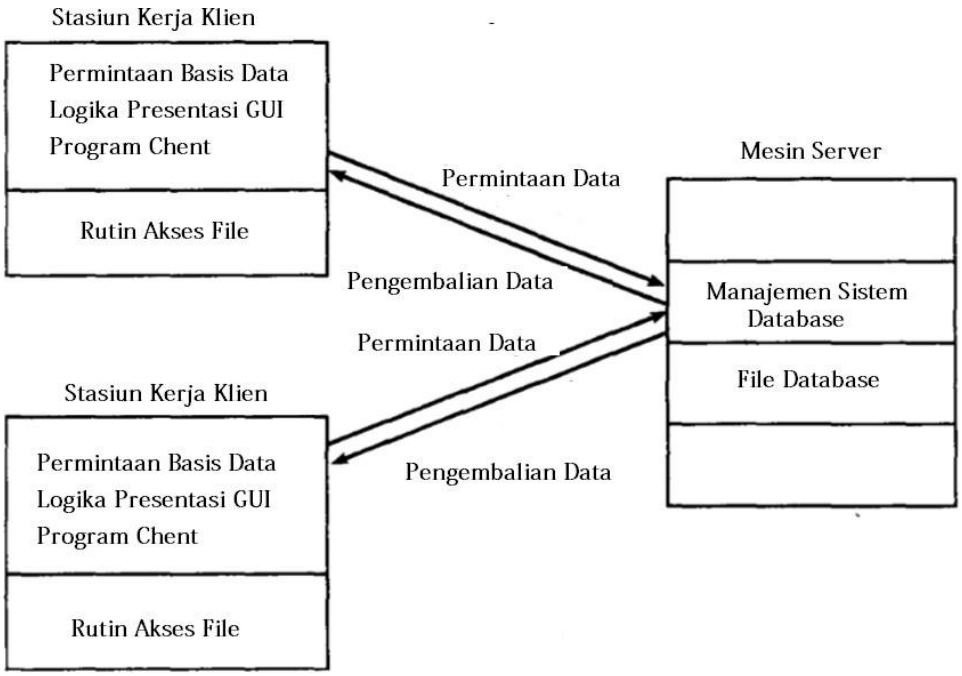


Gambar 2.16 Struktur Database Standar

1. **Tingkat Konseptual:** Ini adalah tingkat di mana desain database konseptual dilakukan. Desain Database konseptual melibatkan analisis kebutuhan informasi pengguna dan definisi item data yang diperlukan untuk memenuhinya. Hasil desain konseptual adalah skema konseptual, deskripsi tunggal dan logis dari semua elemen data dan hubungannya.
2. **Tingkat Eksternal:** Ini terdiri dari pandangan pengguna dari database. Setiap grup pengguna yang ditentukan akan memiliki tampilan databasenya sendiri. Masing- masing tampilan ini memberikan deskripsi berorientasi pengguna dari elemen data dan hubungan yang terdiri dari tampilan tersebut. Hal ini dapat langsung diturunkan dari skema konseptual.
3. **Tingkat internal:** Tingkat internal menyediakan tampilan fisik dari database-disk drive, alamat fisik, indeks, pointer dan sebagainya. Level ini merupakan tanggung jawab perancang Database fisik, yang memutuskan

perangkat fisik mana yang akan berisi data, metode akses apa yang akan digunakan untuk mengambil dan memperbarui data, dan tindakan apa yang akan diambil untuk mempertahankan atau meningkatkan kinerja database.

Arsitektur Two-Tier dan Three-Tier dari Database Manajemen Sistem

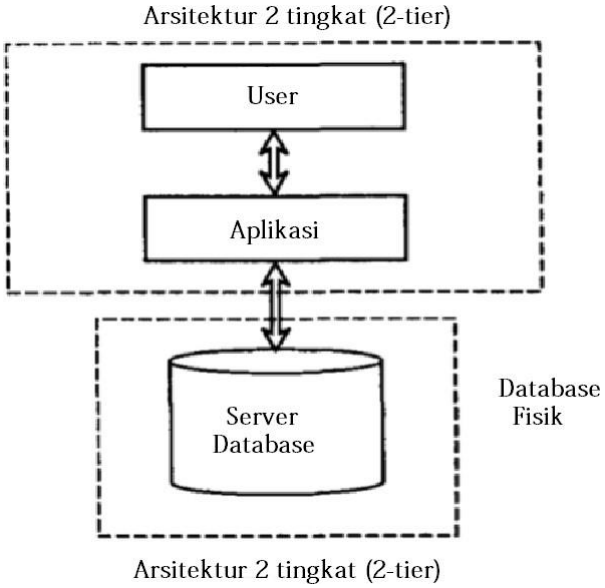


Gambar 2.17 Arsitektur Two-Tier dan Three-Tier dari DBMS

1. Arsitektur client/server Two-Tier untuk DBMS: Dalam arsitektur dua tingkat, aplikasi dipartisi menjadi komponen yang berada di mesin klien, yang membangkitkan fungsionalitas sistem Database pada mesin server melalui pernyataan bahasa kueri. Standar antarmuka program aplikasi digunakan untuk interaksi antara klien dan server. Arsitektur dua tingkat Client/Server memiliki dua komponen penting:
    a. PC klien dan
    b. Server database
2. Pertimbangan Tingkat
    a. Program klien mengakses database secara langsung:
        - Memerlukan perubahan kode untuk port ke database yang berbeda.
        - Volume lalu lintas yang tinggi karena pengiriman data.

b. Program klien mengeksekusi logika aplikasi:

- Dibatasi oleh kemampuan pemrosesan workstation klien (memori, CPU).
- Membutuhkan kode aplikasi untuk didistribusikan ke setiap workstation klien.
- SQL menyediakan bahasa standar untuk RDBMS. Ini menciptakan titik pemisah logis antara klien dan server. Oleh karena itu, fungsi kueri dan transaksi tetap berada di sisi server. Dalam arsitektur seperti itu, server sering disebut server kueri atau server transaksi, karena menyediakan dua hal ini:



Gambar 2.18 Arsitektur 2 tingkat

Keuntungan Arsitektur 2 Tingkat

1. Struktur sederhana
2. Mudah diatur dan dipelihara
3. Performa yang memadai untuk lingkungan dengan volume rendah hingga sedang.
4. Logika bisnis dan database secara fisik dekat, yang memberikan kinerja lebih tinggi.
5. Ini adalah kompatibilitas yang sederhana dan mulus dengan sistem yang ada.

Kekurangan:

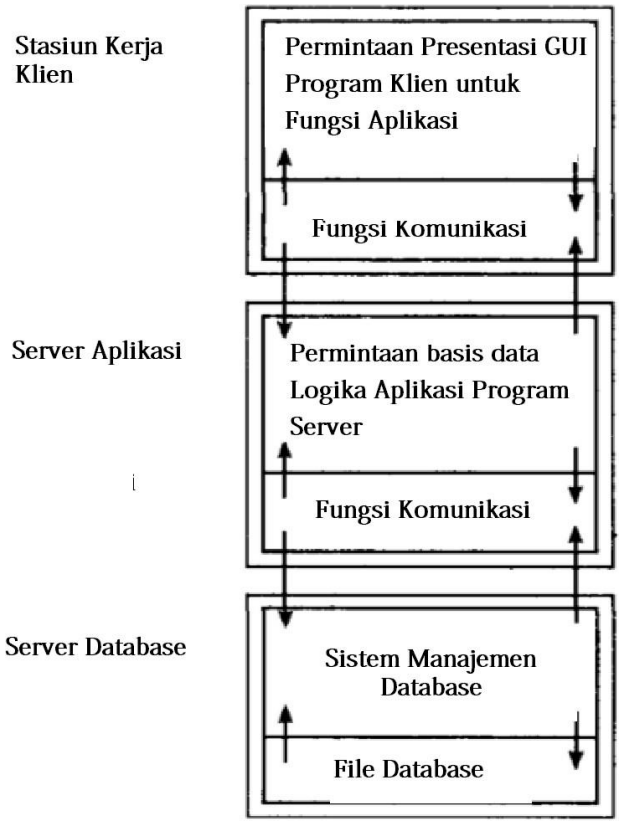
1. Aturan aplikasi yang kompleks sulit untuk diterapkan dalam database, server membutuhkan lebih banyak kode untuk klien.
2. Aturan aplikasi yang kompleks sulit diterapkan di klien dan memiliki kinerja yang buruk.
3. Perubahan logika bisnis tidak secara otomatis diberlakukan oleh server-hanges memerlukan perangkat lunak sisi klien baru untuk didistribusikan dan diinstal.
4. Tidak portabel untuk platform server database lainnya.
5. Performa yang tidak memadai untuk lingkungan volume sedang hingga tinggi, karena server database diperlukan untuk menjalankan logika bisnis. Ini memperlambat operasi database di server database.

Arsitektur Client/Server 3-Tier untuk DBMS

Arsitektur tiga tingkat, yang menambahkan lapisan perantara antara klien dan server database. Lapisan perantara atau tingkat menengah ini disebut server aplikasi. Server ini berperan sebagai perantara dengan menyimpan aturan bisnis (prosedur atau kendala) yang digunakan untuk mengakses data dari server database. Itu juga dapat meningkatkan keamanan database dengan memeriksa kredensial klien sebelum meneruskan permintaan ke server database. Klien berisi antarmuka GUI dan beberapa aturan bisnis khusus aplikasi tambahan. Server perantara menerima permintaan dari klien, memproses permintaan dan mengirimkan perintah database ke server database, dan kemudian bertindak sebagai saluran untuk meneruskan data yang diproses dari server database ke klien, di mana dapat diproses lebih lanjut dan disaring dan disajikan ke pengguna dalam format GUI. Dengan demikian, antarmuka pengguna, aturan aplikasi, dan akses data bertindak sebagai tiga tingkatan.

Arsitektur server klien 3-tier memiliki tiga komponen penting:

- PC klien
- Server Aplikasi
- Server Database



Gambar 2.19 Arsitektur 3 tingkat

3. Pertimbangan Arsitektur Tier
    a. Program klien hanya berisi logika presentasi:
        - Lebih sedikit sumber daya yang dibutuhkan untuk workstation klien.
        - Tidak ada modifikasi klien jika lokasi database berubah.
        - Lebih sedikit kode untuk didistribusikan ke cleint workstation.
    b. Satu server menangani banyak permintaan klien :
        - Tersedia lebih banyak sumber daya untuk program server.
        - Mengurangi lalu lintas data di jaringan.

Kelebihan:

1. Aturan aplikasi yang kompleks mudah diimplementasikan di server aplikasi.
2. Logika bisnis diturunkan dari server database dan klien, yang meningkatkan kinerja.
3. Perubahan logika bisnis secara otomatis diberlakukan oleh server-perubahan hanya memerlukan perangkat lunak server aplikasi baru yang akan diinstal.

4. Logika server aplikasi portabel ke platform server database lain berdasarkan perangkat lunak aplikasi.
5. Performa superior untuk lingkungan volume sedang hingga tinggi.
6. Teknologi enkripsi dan dekripsi membuatnya lebih aman untuk mentransfer data sensitif dari server ke klien dalam bentuk terenkripsi, di mana ia akan didekripsi.
7. Berbagai teknologi untuk kompresi data membantu dalam mentransfer sejumlah besar data dari server ke klien.

Kekurangan :

1. Struktur lebih kompleks.
2. Lebih sulit untuk diatur dan dipelihara.
3. Masalah keamanan jaringan.
4. Pemisahan fisik server aplikasi yang mencakup fungsi logika bisnis dan server Database yang berisi Database mungkin sedikit memengaruhi kinerja.

BAB 3

PEMOGRAMAN DINAMIS

Pemrograman Dinamis

Pemrograman dinamis adalah teknik desain algoritma yang dapat digunakan untuk menemukan solusi optimal untuk masalah dan menghitung jumlah solusi. Bab ini adalah pengantar pemrograman dinamis, dan teknik ini akan digunakan berkali-kali nanti dalam buku ini ketika merancang algoritma. Bagian pertama membahas elemen dasar pemrograman dinamis dalam konteks masalah penukaran koin. Dalam masalah ini kita diberikan satu set nilai koin dan tugas kita adalah membuat sejumlah uang dengan menggunakan koin sesedikit mungkin. Ada algoritma serakah sederhana untuk masalah ini, tetapi seperti yang akan kita lihat, itu tidak selalu menghasilkan solusi yang optimal. Namun, dengan menggunakan pemrograman dinamis, kita dapat membuat algoritma yang efisien yang selalu menemukan solusi optimal. Bagian selanjutnyamenyajikan pilihan masalah yang menunjukkan beberapa kemungkinan pemrograman dinamis. Soal-soal tersebut meliputi menentukan barisan kenaikan terpanjang dalam suatu larik, menemukan jalur optimal dalam kisi dua dimensi, dan menghasilkan semua jumlah bobot yang mungkin dalam masalah knapsack.

3.1 Konsep Dasar

Pada bagian ini, kita membahas konsep dasar pemrograman dinamis dalam konteks masalah perubahan koin. Pertama-tama kita menyajikan algoritma yang disepakati untuk masalah tersebut, yang tidak selalu menghasilkan solusi optimal. Setelah ini, kami menunjukkan bagaimana masalah dapat diselesaikan secara efisien menggunakan pemrograman dinamis.

Misalkan kita diberi satu set nilai koin coin ={ c1,c2,...,ck} dan jumlah uang n yang ditargetkan, dan kita diminta untuk membuat jumlah n menggunakan koin sesedikit mungkin. Tidak ada batasan berapa kali kita dapat menggunakan setiap nilai koin. Misalnya, jika koin={ 1,2,5}dan n =12, solusi optimalnya adalah 5+5+2=12, yang membutuhkan tiga koin. Ada algoritma serakah alami untuk memecahkan masalah: selalu pilih koin terbesar yang mungkin sehingga jumlah nilai koin tidak melebihi jumlah target. Misalnya, jika n = 12, pertama-tama kita memilih dua koin bernilai 5, dan

kemudian satu koin bernilai 2, yang melengkapi penyelesaian. Ini terlihat seperti strategi yang masuk akal, tetapi apakah selalu optimal?

Ternyata strategi ini tidak selalu berhasil. Misalnya, jika koin = {1,3,4} dan n = 6, solusi optimal hanya memiliki dua koin (3+3 = 6) tetapi strategi serakah menghasilkan solusi dengan tiga koin (4+1+1 = 6). Contoh tandingan sederhana ini menunjukkan bahwa algoritma serakah tidak benar.7 Bagaimana kita bisa menyelesaikan masalah? Tentu saja, kita dapat mencoba menemukan algoritma serakah lain, tetapi tidak ada strategi jelas lain yang dapat kita pertimbangkan. Kemungkinan lain adalah membuat algoritma brute force yang melewati semua cara yang mungkin untuk memilih koin. Algoritma seperti itu pasti akan memberikan hasil yang benar, tetapi akan sangat lambat pada input yang besar. Namun, dengan menggunakan pemrograman dinamis, kita dapat membuat algoritma yang hampir mirip dengan algoritma brute force tetapi juga efisien. Dengan demikian, kita berdua dapat yakin bahwa algoritme itu benar dan menggunakannya untuk memproses input yang besar. Selanjutnya, kita dapat menggunakan teknik yang sama untuk memecahkan sejumlah besar masalah lain.

3.1.1 Menemukan Solusi Optimal

Untuk menggunakan pemrograman dinamis, kita harus merumuskan masalah secara rekursif sehingga solusi masalah dapat dihitung dari solusi ke sub masalah yang lebih kecil. Dalam masalah koin, masalah rekursif alami adalah menghitung nilai suatu fungsi. solve (x): berapa jumlah minimum koin yang diperlukan untuk membentuk jumlah x? Jelas, nilai fungsi bergantung pada nilai koin. Misalnya, If coin = {1,3,4}, nilai pertama dari fungsi tersebut adalah sebagai berikut:

solve(0) = 0

solve(1) = 1

solve(2) = 2

solve(3) = 1

solve(4) = 1

solve(5) = 2

solve(6) = 2

solve(7) = 2

solve(8) = 2

solve(9) = 3

solve(10) = 3

Misalnya, solve(10) = 3, karena paling sedikit diperlukan 3 koin untuk membentuk jumlah 10. Solusi optimalnya adalah 3+3+4=10. Sifat esensial dari solve adalah bahwa nilainya dapat dihitung secara rekursif dari nilai yang lebih kecil. Idenya adalah untuk fokus pada koin pertama yang kita pilih untuk dijumlahkan. Sebagai contoh, dalam skenario di atas, koin pertama dapat berupa 1, 3 atau 4. Jika pertama kita memilih koin 1, tugas yang tersisa adalah membentuk jumlah 9 menggunakan jumlah koin minimum, yang merupakan submasalah dari koin asli. masalah. Tentu saja, hal yang sama berlaku untuk koin 3 dan 4. Jadi, kita dapat menggunakan rumus rekursif berikut untuk menghitung jumlah minimum koin:

solve(x) = min(solve(x −1)+1,

solve(x −3)+1,

solve(x −4)+1).

Kasus dasar dari rekursi adalah solve(0) = 0, karena tidak diperlukan koin untuk membentuk jumlah kosong. Sebagai contoh,

solve(10) =solve(7)+1=solve(4)+2=solve(0)+3=3.

Sekarang kita siap untuk memberikan fungsi rekursif umum yang menghitung jumlah minimum koin yang diperlukan untuk membentuk jumlah x:

$$solve(x) = \begin{cases} \infty & x < 0 \\ 0 & x = 0 \\ \min_{c \in coins} solve(x-c) + 1 & x > 0 \end{cases}$$

Pertama, jika x < 0, nilainya tak berhingga, karena tidak mungkin membentuk jumlah uang negatif. Kemudian, jika x =0,nilainya nol, karena uang logam diperlukan untuk membentuk jumlah kosong. Akhirnya, jika x > 0, variabel c melewati semua kemungkinan bagaimana memilih koin pertama dari jumlah tersebut. Setelah fungsi rekursif yang memecahkan masalah telah ditemukan, kita dapat langsung mengimplementasikan solusi dalam C++ (konstanta INF menunjukkan tak terhingga):

```
int solve(int x) {
if (x < 0)return INF; if (x == 0) return 0; int best = INF;
for (auto c : coins) {
best = min(best, solve(x-c)+1);
}
return best;
```

}

Namun, fungsi ini tidak efisien, karena mungkin ada banyak cara untuk membuat jumlah dan fungsi memeriksa semuanya. Untungnya, ternyata ada cara sederhana untuk membuat fungsi tersebut menjadi efisien.

Memoisasi

Ide kunci dalam pemrograman dinamis adalah memoisasi, yang berarti bahwa kita menyimpan setiap nilai fungsi dalam array secara langsung setelah menghitungnya. Kemudian, ketika nilai tersebut dibutuhkan lagi, nilai tersebut dapat diambil dari array tanpa panggilan rekursif. Untuk melakukan ini, kami membuat array

```
bool ready[N];
int value[N];
```

di mana ready[x] menunjukkan apakah nilai solve(x) telah dihitung, dan jika ya, nilai[x] berisi nilai ini. Konstanta N telah dipilih sehingga semua nilai yang dibutuhkan sesuai dengan array. Setelah ini, fungsi tersebut dapat diimplementasikan secara efisien sebagai berikut:

```
int solve(int x) {
if (x < 0)return INF;
if (x == 0) return 0;
if (ready[x]) return value[x];
int best = INF;
for (auto c : coins) {
best = min(best, solve(x-c)+1);
}
ready[x] = true; value[x] = best; return best;
}
```

Fungsi menangani kasus dasar x < 0 dan x = 0 seperti sebelumnya. Kemudian memeriksa dari ready[x]jika solve(x) telah disimpan dalam nilai[x], dan jika ya, fungsi langsung mengembalikannya. Jika tidak, fungsi menghitung nilai solve(x) secara rekursif dan menyimpannya dalam nilai[x]. Fungsi ini bekerja dengan efisien, karena jawaban untuk setiap parameter x dihitung secara rekursif hanya satu kali. Setelah nilai penyelesaian (x) telah disimpan dalam nilai [x], fungsi tersebut dapat diambil secara efisien setiap kali fungsi dipanggil kembali dengan parameter x. Kompleksitas waktu dari

algoritma adalah O(nk), di mana n adalah jumlah target dan k adalah jumlah koin.
Implementasi Berulang

Perhatikan bahwa kita juga dapat membangun nilai array secara iteratif menggunakan loop sebagai berikut:

```
value[0] = 0;
for (int x = 1; x <= n; x++) {
value[x] = INF;
 for (auto c : coins) {
if (x-c >= 0) {
value[x] = min(value[x], value[x-c]+1);
}
}
}
```

Faktanya, sebagian besar programmer kompetitif lebih menyukai implementasi ini, karena lebih pendek dan memiliki faktor konstan yang lebih kecil. Mulai sekarang, kami juga menggunakan implementasi berulang dalam contoh kami. Namun, seringkali lebih mudah untuk memikirkan solusi pemrograman dinamis dalam hal fungsi rekursif.
Membangun Solusi

Kadang-kadang kita diminta untuk menemukan nilai solusi optimal dan memberikan contoh bagaimana solusi tersebut dapat dibangun. Untuk membangun solusi optimal dalam masalah koin, kita dapat mendeklarasikan array baru yang menunjukkan untuk setiap jumlah uang koin pertama dalam solusi optimal:

```
int first[N];
```

Kemudian, kita dapat memodifikasi algoritma sebagai berikut:

```
value[0] = 0;
for (int x = 1; x <= n; x++) {
value[x] = INF;
for (auto c : coins) {
if (x-c >= 0 && value[x-c]+1 < value[x]) { value[x] = value[x-c]+1; first[x] = c;
}
}
}
```

Setelah ini, kode berikut mencetak koin yang muncul dalam solusi optimal untuk jumlah n:

6.1.3 Menghitung Solusi

Sekarang mari kita pertimbangkan varian lain dari masalah koin di mana tugas kita adalah menghitung jumlah total cara untuk menghasilkan jumlah x menggunakan koin. Misalnya, jika coin={ 1,3,4}dan x =5, ada total 6 cara:

- 1+1+1+1+1
- 1+1+3
- 1+3+1
- 3+1+1
- 1+4
- 4+1

Sekali lagi, kita dapat menyelesaikan masalah secara rekursif. Biarkan Memecahkan(x) menunjukkan jumlah cara kita dapat membentuk jumlah x. Misalnya, jika coin={ 1,3,4}, maka solve(5) =6 dan rumus rekursifnya adalah

solve(x) =solve(x −1)+ solve(x −3)+ solve(x −4).

Maka, fungsi rekursif umum adalah sebagai berikut:

$$solve(x) = \begin{cases} 0 & x < 0 \\ 1 & x = 0 \\ \sum \in c\ oins\ solve(x - c) + 1 & x > 0 \end{cases}$$

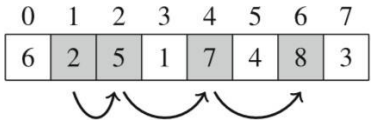
Jika x < 0, nilainya nol, karena tidak ada solusi. Jika x = 0, nilainya satu, karena hanya ada satu cara untuk membentuk jumlah kosong. Jika tidak, kita menghitung jumlah semua nilai dari bentuk solve(x c) di mana c dalam koin. Kode berikut menyusun jumlah larik sedemikian rupa sehingga count[x] sama dengan nilai solve(x) untuk 0≤ x n:

```
count[0] = 1;
for (int x = 1; x <= n; x++) {
for (auto c : coins) {
if (x-c >= 0) {
count[x] += count[x-c];
}
}
}
```

Seringkali jumlah solusi sangat besar sehingga tidak diperlukan untuk menghitung jumlah pastinya tetapi cukup untuk memberikan jawaban modulo m dimana, misalnya, m = 109 +7. Hal ini dapat dilakukan dengan mengubah kode sehingga semua perhitungan dilakukan modulo m. Pada kode di atas, cukup tambahkan baris
count[x] %= m; setelah garis count[x] += count[x-c];

3.2 Contoh Pemrograman Dinamis

Setelah membahas konsep dasar pemrograman dinamis, sekarang kita siap untuk membahas serangkaian masalah yang dapat diselesaikan secara efisien menggunakan pemrograman dinamis. Seperti yang akan kita lihat, pemrograman dinamis adalah teknik serbaguna yang memiliki banyak aplikasi dalam desain algoritma.



Gambar 3.1 Urutan kenaikan terpanjang dari larik ini adalah [2,5,7,8]

3.2.1 Urutan Peningkatan Terpanjang

Barisan kenaikan terpanjang dalam larik n elemen adalah barisan panjang maksimum elemen larik yang bergerak dari kiri ke kanan, dan setiap elemen dalam barisan lebih besar dari elemen sebelumnya. Misalnya, Gambar 6.1 menunjukkan barisan naik terpanjang dalam larik delapan elemen. Kita dapat secara efisien menemukan peningkatan suburutan terpanjang dalam sebuah array menggunakan pemrograman dinamis. Misalkan length(k) menyatakan panjang dari barisan kenaikan terpanjang yang berakhir pada posisi k. Kemudian, jika kita menghitung semua nilai length (k) di mana $0 \leq k \leq n-1$, kita akan menemukan panjang dari barisan yang bertambah panjang. Nilai fungsi untuk array contoh kita adalah sebagai berikut:

length(0) = 1

length(1) = 1

length(2) = 2

length(3) = 1

length(4) = 3

length(5) = 2

length(6) = 4

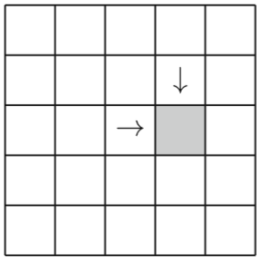
length(7) = 2

Misalnya, panjang(6) = 4, karena barisan kenaikan terpanjang yang berakhir pada posisi 6 terdiri dari 4 elemen. Untuk menghitung nilai length (k), kita harus mencari posisi i < k dimana array[i] < array[k]dan panjang(i) sebesar mungkin. Kemudian kita mengetahui bahwa length (k) = length (i)+1, karena ini adalah cara optimal untuk menambahkan sinar[k] ke suatu barisan. Namun, jika tidak ada posisi i seperti itu, maka length (k) = 1, yang berarti bahwa turunannya hanya berisi array[k]. Karena semua nilai fungsi dapat dihitung dari nilai yang lebih kecil, kita dapat menggunakan pemrograman dinamis untuk menghitung nilainya. Dalam kode berikut, nilai fungsi akan disimpan dalam length array.

```
for (int k = 0; k < n; k++) { length[k] = 1;
for (int i = 0; i < k; i++) {
if (array[i] < array[k]) {
length[k] = max(length[k],length[i]+1);
}
}
}
```

3	7	9	2	7
9	8	3	5	5
1	7	9	8	5
3	8	6	4	10
6	3	9	7	8

Gambar 3.2 Jalur optimal dari sudut kiri atas ke sudut kanan bawah



Gambar 3.3 Dua cara yang mungkin untuk mencapai persegi di jalan

3.2.2 Jalur dalam Grid

Masalah kita selanjutnya adalah menemukan jalur dari sudut kiri atas ke sudut kanan bawah dari kisi nxn, dengan batasan bahwa kita hanya dapat bergerak ke bawah dan ke kanan. Setiap kotak berisi bilangan bulat, dan jalurnya harus dibuat sedemikian rupa sehingga jumlah nilai sepanjang jalur tersebut sebesar mungkin. Sebagai contoh, Gambar 3.2 menunjukkan jalur optimal dalam kisi 5x5. Jumlah nilai pada lintasan adalah 67, dan ini adalah jumlah terbesar yang mungkin pada lintasan dari sudut kiri atas ke sudut kanan bawah. Asumsikan bahwa baris dan kolom kisi diberi nomor dari 1 hingga n, dan value[y][x]sama dengan nilai kuadrat (y,x). Misalkan sum(y,x) menunjukkan jumlah maksimum pada lintasan dari sudut kiri atas ke persegi (y,x). Kemudian, sum(n,n) memberi tahu kita jumlah maksimum dari sudut kiri atas ke sudut kanan bawah. Misalnya, pada kisi di atas, jumlah(5,5) =67. Sekarang kita bisa menggunakan rumus

sum(y,x) =max(sum(y,x −1),sum(y−1,x))+value[y][x],

yang berdasarkan pengamatan bahwa lintasan yang berakhir pada bujur sangkar (y,x) dapat berasal dari square (y,x −1) atau dari square(y−1,x) (Gambar 6.3). Jadi, kami memilih arah yang memaksimalkan jumlah. Kami berasumsi bahwa sum(y,x) = 0 ify = 0 atau x =0, sehingga rumus rekursif juga berfungsi untuk kuadrat paling kiri dan paling atas. Karena jumlah fungsi memiliki dua parameter, array pemrograman dinamis juga memiliki dua dimensi. Misalnya, kita dapat menggunakan array

```
int sum[N][N];
```

dan hitung jumlahnya sebagai berikut:

```
for (int y = 1; y <= n; y++) {
for (int x = 1; x <= n; x++) {
sum[y][x] = max(sum[y][x-1],sum[y-1][x])+value[y][x];
}
}
```

Kompleksitas waktu dari algoritma adalah O(n2).

3.2.3 Masalah Ransel

Istilah knapsack mengacu pada masalah di mana satu set objek diberikan, dan himpunan bagian dengan beberapa properti harus ditemukan. Masalah knapsack seringkali dapat diselesaikan dengan menggunakan pemrograman dinamis. Pada bagian ini, kita fokus pada masalah berikut: Diberikan daftar bobot [w1,w2,...,wn], tentukan semua jumlah

yang dapat dibangun dengan menggunakan bobot. Misalnya, Gambar 6.4 menunjukkan jumlah yang mungkin untuk bobot [1,3,3,5]. Dalam hal ini, semua jumlah antara 0...12 dimungkinkan, kecuali 2 dan 10. Misalnya, jumlah 7 dimungkinkan karena kita dapat memilih bobot[1,3,3]. Untuk memecahkan masalah, kami fokus pada submasalah di mana kami hanya menggunakan k bobot pertama untuk membangun jumlah. Misalkan possible(x,k) =benar jika kita dapat membuat jumlah x menggunakan k bobot pertama, dan jika tidak, possible(x,k)=false. Nilai fungsi dapat dihitung secara rekursif menggunakan rumus

possible(x,k) =possible(x −wk,k−1) or possible(x,k−1),

yang didasarkan pada fakta bahwa kita dapat menggunakan atau tidak menggunakan bobot wk dalam penjumlahan. Jika kita menggunakan wk, tugas yang tersisa adalah membentuk jumlah x−wk menggunakan bobot k−1 pertama, dan jika kita tidak menggunakan wk, tugas yang tersisa adalah membentuk jumlah x menggunakan bobot k−1 pertama. Kasus dasarnya adalah

possible(

x0)

$true \quad x = 0$

= {

$false \quad x \neq 0$

karena jika tidak ada bobot yang digunakan, kita hanya dapat membentuk penjumlahan0. Akhirnya, possible(x,n) memberi tahu kita apakah kita dapat membuat jumlah x menggunakan semua bobot.

0	1	2	3	4	5	6	7	8	9	10	11	12
✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓

Gambar 3.4 Membuat penjumlahan menggunakan bobot[1,3,3,5]

	0	1	2	3	4	5	6	7	8	9	10	11	12
$k=0$	✓												
$k=1$	✓	✓											
$k=2$	✓	✓		✓	✓								
$k=3$	✓	✓		✓	✓		✓	✓					
$k=4$	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓

Gambar 3.5 Memecahkan masalah knapsack untuk bobot [1,3,3,5]menggunakan pemrograman dinamis

Gambar 3.5 menunjukkan semua nilai fungsi untuk bobot [1, 3, 3, 5] (simbol " √" menunjukkan nilai sebenarnya). Misalnya, baris k = 2 memberi tahu kita bahwa kita dapat membuat sum[0,1,3,4] menggunakan bobot[1,3]. Biarkan m menunjukkan jumlah total bobot. Solusi pemrograman dinamis waktu O(nm) berikut sesuai dengan fungsi rekursif:

```
possible[0][0] = true;
for (int k = 1; k <= n; k++) {
for (int x = 0; x <= m; x++) {
if (x-w[k] >= 0) { possible[x][k] |=
possible[x-w[k]][k-1];
}
possible[x][k] |= possible[x][k-1];
}
}
```

Ternyata ada juga cara yang lebih ringkas untuk mengimplementasikan perhitungan pemrograman dinamis, dengan hanya menggunakan array satu dimensi yang memungkinkan[x] yang menunjukkan apakah kita dapat membuat subset dengan jumlah x. Triknya adalah memperbarui array dari kanan ke kiri untuk setiap bobot baru:

```
possible[0] = true;
for (int k = 1; k <= n; k++) {
for (int x = m-w[k]; x >= 0; x--) { possible[x+w[k]] |= possible[x];
}
}
```

Perhatikan bahwa ide pemrograman dinamis umum yang disajikan di bagian ini juga dapat digunakan dalam masalah knapsack lainnya, seperti dalam situasi di mana objek memiliki bobot dan nilai dan kita harus menemukan subset nilai maksimum yang bobotnya tidak melebihi batas yang diberikan.

3.2.4 Dari Permutasi ke Subset

Menggunakan pemrograman dinamis, seringkali dimungkinkan untuk mengubah sebuah iterasi pada permutasi menjadi sebuah iterasi pada himpunan bagian. Manfaatnya adalah bahwa n!, jumlah permutasi, jauh lebih besar dari 2n, jumlah himpunan bagian. Sebagai contoh, jika n =20, n!≈2.4x1018 dan 2n ≈ 106. Jadi, untuk nilai n tertentu, kita dapat melewati himpunan bagian secara efisien tetapi tidak melalui permutasi. Sebagai contoh, perhatikan masalah berikut: Ada lift dengan berat maksimum x, dan n orang yang ingin naik dari lantai dasar ke lantai atas. Orang-orang tersebut diberi nomor 0,1,...,n−1, dan berat orang i adalah weight[i]. Berapa jumlah minimum perjalanan yang diperlukan untuk membawa semua orang ke lantai atas? Misalnya, misalkan x =12, n =5, dan bobotnya adalah sebagai berikut:

- weight[0]=2
- weight[1]=3
- weight[2]=4
- weight[3]=5
- weight[4]=9

Dalam skenario ini, jumlah minimum perjalanan adalah dua. Salah satu solusi optimal adalah sebagai berikut: pertama, orang 0, 2, dan 3 naik lift (berat total 11), dan kemudian, orang 1 dan 4 naik lift (total berat 12). Soal dapat dengan mudah diselesaikan dalam waktu O(n!n) dengan menguji semua kemungkinan permutasi dari n orang. Namun, kita dapat menggunakan pemrograman dinamis untuk membuat algoritma waktu O(2nn) yang lebih efisien. Idenya adalah untuk menghitung untuk setiap subset orang dua nilai: jumlah minimum perjalanan yang dibutuhkan dan berat minimum orang yang naik dalam kelompok terakhir. Biarkan rides(S) menunjukkan jumlah minimum perjalanan untuk subset S, dan biarkan last(S) menunjukkan berat minimum perjalanan terakhir dalam solusi di mana jumlah perjalanan minimum.

Misalnya, dalam skenario di atas

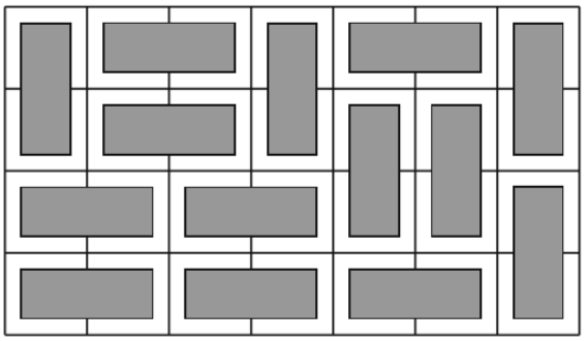
rides({3,4}) =2 and last({3,4}) =5,

karena cara optimal bagi orang 3 dan 4 untuk sampai ke lantai atas adalah mereka mengambil dua jalan terpisah dan orang 4 pergi pertama, yang meminimalkan berat perjalanan kedua. Tentu saja, tujuan akhir kita adalah menghitung nilai rides({0...n−1}). Kita dapat menghitung nilai fungsi secara rekursif dan kemudian menerapkan pemrograman dinamis. Untuk menghitung nilai subset S, kami menelusuri semua orang

yang termasuk dalam S dan secara optimal memilih orang terakhir p yang memasuki elevator. Setiap pilihan seperti itu menghasilkan submasalah untuk sebagian kecil orang. Jika terakhir(S\p)+ weight[p]≤x,kita dapat menambahkan p ke jalan terakhir. Jika tidak, kita harus menyimpan wahana baru yang hanya berisi p. Cara mudah untuk mengimplementasikan perhitungan pemrograman dinamis adalah dengan menggunakan operasi bit. Pertama, kita mendeklarasikan sebuah array pair<int,int> best[1<<N];

yang berisi untuk setiap subset S sepasang (naik(S),terakhir(S)). Untuk subset kosong, tidak ada perjalanan yang diperlukan:

best[0] = {0,0};



Gambar 3.6 Salah satu cara untuk mengisi kisi 4×7 menggunakan ubin 1×2 dan 2×1

Kemudian, kita dapat mengisi array sebagai berikut:

```
for (int s = 1; s < (1<<n); s++) {
//initial value : n+1rides are needed best[s] = {n+1,0};
for (int p = 0; p < n; p++) {
if (s&(1<<p)) {
auto option = best[s^(1<<p)];
if (option.second+weight[p] <= x) {
//addp to a nexist in gride option.second += weight[p];
} else {
//reserve a new ride for p option.first++; option.second = weight[p];
}
best[s] = min(best[s], option);
}
}
}
```

Perhatikan bahwa loop di atas menjamin bahwa untuk setiap dua himpunan bagian S1 dan S2 sedemikian rupa sehingga S1 ⊂ S2, kita memproses S1 sebelum S2. Dengan demikian, nilai pemrograman dinamis dihitung dalam urutan yang benar.

3.2.5 Menghitung kotak

Terkadang keadaan solusi pemrograman dinamis lebih kompleks daripada kombinasi nilai yang tetap. Sebagai contoh, perhatikan masalah menghitung jumlah cara yang berbeda untuk mengisi kisi nxm menggunakan ubin ukuran 1x2 dan 2x1. Misalnya, ada total 781 cara untuk mengisi kisi 4x7, salah satunya adalah solusi yang ditunjukkan pada Gambar 6.6. Masalah tersebut dapat diselesaikan dengan menggunakan pemrograman dinamis dengan menelusuri grid baris demi baris. Setiap baris dalam solusi dapat direpresentasikan sebagai string yang berisi m karakter dari himpunan {⊓,
M, ⊏, ⊐}. Misalnya, solusi pada Gambar 6.6 terdiri dari empat baris yang sesuai dengan string berikut:

- ⊓⊏⊐⊓⊏⊐⊓
- M⊏⊐M⊓⊓M
- ⊏⊐⊏⊐MM⊓
- ⊏⊐⊏⊐⊏⊐M

Misalkan baris grid diindeks dari 1 sampai n. Misalkan count(k,x) menyatakan banyaknya cara untuk membangun solusi untuk baris 1...k sedemikian rupa sehingga string x sesuai dengan baris k. Dimungkinkan untuk menggunakan pemrograman dinamis di sini, karena status baris hanya dibatasi oleh status baris sebelumnya. Solusi valid jika baris 1 tidak berisi karakter , baris tidak berisi karakter , dan semua baris berurutan kompatibel. Misalnya, baris M⊏⊐M⊓⊓M dan ⊏⊐⊏⊐MM⊓ kompatibel, sedangkan baris ⊓⊏⊐⊓⊏⊐⊓ dan ⊏⊐⊏⊐⊏⊐M tidak kompatibel. Karena satu baris terdiri dari beberapa karakter dan ada empat pilihan untuk setiap karakter, jumlah baris yang berbeda paling banyak 4m.

Kita dapat melalui O(4m) keadaan yang mungkin untuk setiap baris, dan untuk setiap keadaan, ada O(4m) keadaan yang mungkin untuk baris sebelumnya, sehingga kompleksitas waktu dari penyelesaiannya adalah O(n42m). Dalam praktiknya, ide yang baik untuk memutar kisi-kisi sehingga sisi terpendek memiliki panjang m, karena faktor 42m mendominasi kompleksitas waktu. Dimungkinkan untuk membuat solusi lebih efisien dengan menggunakan representasi baris yang lebih ringkas. Ternyata cukup

untuk mengetahui kolom mana dari baris sebelumnya yang berisi bujur sangkar atas ubin vertikal. Dengan demikian, kita dapat merepresentasikan baris hanya dengan menggunakan karakter ⊓ dan , Dimana merupakan kombinasi dari karakter M, ⊏ dan ⊐ . Menggunakan representasi ini, hanya ada 2m baris yang berbeda, dan kompleksitas waktu adalah O(n22m). Sebagai catatan terakhir, ada juga rumus langsung untuk menghitung jumlah ubin:

n [2]

m [2]

G G 4 x ($cos2\ \pi a$

n + 1

+ $cos2\ \pi b$)

m + 1

a=1 b=1

Rumus ini sangat efisien, karena menghitung jumlah ubin dalam waktu O(nm), tetapi karena jawabannya adalah perkalian bilangan real, masalah saat menggunakan rumus adalah bagaimana menyimpan hasil antara secara akurat.

Banyak masalah pemrograman dapat diselesaikan dengan mempertimbangkan situasi sebagai graf dan menggunakan algoritma graf yang sesuai. Dalam bab ini, kita akan mempelajari dasar-dasar grafik dan pilihan algoritma graf yang penting. Bagian pertama membahas terminologi grafik dan struktur data yang dapat digunakan untuk mewakili grafik dalam algoritma. Bagian selanjutnya memperkenalkan dua algoritma traversal graf fundamental. Pencarian Depth First adalah cara sederhana untuk mengunjungi semua node yang dapat dicapai dari node awal, dan pencarian Breadth First mengunjungi node dalam urutan jarak yang meningkat dari node awal. Bagian selanjutnya menyajikan algoritma untuk menemukan jalur terpendek dalam grafik berbobot.

Algoritma Bellman-Ford adalah algoritma sederhana yang menemukan jalur terpendek dari node awal ke semua node lainnya. Algoritma Dijkstra merupakan algoritma yang lebih efisien yang mensyaratkan bahwa bobot genap tidak negatif. Algoritma Floyd-Warshall menentukan jalur terpendek antara semua pasangan simpul

dari suatu graf. Bagian tengah mengeksplorasi sifat khusus dari grafik asiklik berarah. Kita akan belajar bagaimana membangun semacam topologi dan bagaimana menggunakan pemrograman dinamis untuk secara efisien memproses grafik tersebut. Bagian selanjutnya berfokus pada grafik sukses di mana setiap node memiliki suksesor unik. Kami akan membahas cara yang efisien untuk menemukan penerus node dan algoritma Floyd untuk deteksi siklus. Bagian akhir menyajikan algoritma Kruskal dan Prim untuk membangun pohon merentang minimum. Algoritma Kruskal didasarkan pada struktur union-find yang efisien yang juga memiliki kegunaan lain dalam desain algoritma.

3.2.6 Pemrograman Dinamis

Dengan menggunakan pemrograman dinamis, kita dapat secara efisien menjawab banyak pertanyaan tentang jalur dalam grafik asiklik berarah. Contoh pertanyaan seperti itu adalah:

- Apa jalur terpendek/terpanjang dari simpul a ke simpul b?
- Ada berapa banyak jalan yang berbeda?
- Berapa jumlah tepi minimum/maksimum dalam suatu jalur?
- Node mana yang muncul di setiap jalur yang mungkin?

Perhatikan bahwa banyak dari masalah di atas sulit dipecahkan atau tidak terdefinisi dengan baik untuk graf umum. Sebagai contoh, pertimbangkan masalah menghitung jumlah jalur dari node a ke node b. Misalkan path (x) menyatakan jumlah path dari node a ke node x. Sebagai kasus dasar, path(a) =1. Kemudian, untuk menghitung nilai lain dari path (x), kita dapat menggunakan rumus rekursif

paths(x) =paths(s1)+paths(s2)+⋯+paths(sk),

dimana s1,s2,...,sk adalah node dari mana ada tepi ke x. Karena grafiknya asiklik, nilai jalur dapat dihitung dalam urutan topologi. Gambar7.29 menunjukkan nilai jalur dalam skenario contoh di mana kita ingin menghitung jumlah jalur dari node 1 ke node 6. Misalnya,

paths(6) =paths(2)+paths(3),

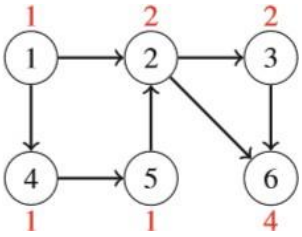
karena edge yang berakhir pada node 6 adalah 2→6 dan 3→ 6. Karena path(2) = 2 dan path(3) =2, kita simpulkan bahwa path(6) =4. Jalan-jalannya adalah sebagai berikut:

- 1→2→3→6
- 1→2→6

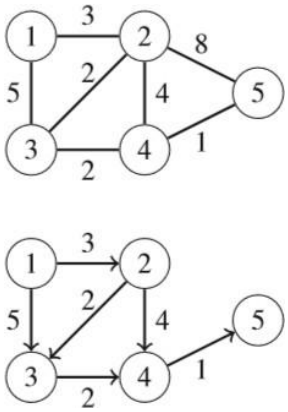
- 1→4→5→2→3→6
- 1→4→5→2→6

Memproses Jalur Terpendek

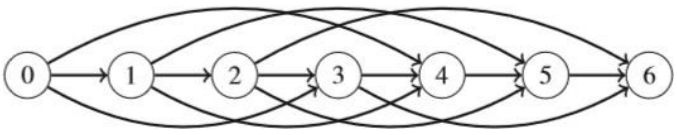
Pemrograman dinamis juga dapat digunakan untuk menjawab pertanyaan tentang jalur terpendek pada grafik umum (tidak harus asiklik). Yaitu, jika kita mengetahui jarak minimum dari node awal ke node lain (misalnya, setelah menggunakan algoritma Dijkstra), kita dapat dengan mudah membuat grafik jalur terpendek asiklik berarah yang menunjukkan untuk setiap node cara yang mungkin untuk mencapai node menggunakan jalur terpendek dari node awal.



Gambar 3.7 Menghitung jumlah jalur dari node 1 ke node 6



Gambar 3.8 Sebuah graf dan graf jalur terpendeknya



Gambar 3.9 Masalah koin sebagai grafik asiklik berarah

Faktanya, setiap masalah pemrograman dinamis dapat direpresentasikan sebagai grafik asiklik terarah di mana setiap node sesuai dengan keadaan pemrograman dinamis dan tepinya menunjukkan bagaimana keadaan bergantung satu sama lain. Misalnya, pertimbangkan masalah pembentukan jumlah uang n menggunakan koin

{c1,c2,...,ck} Dalam skenario ini, kita dapat membuat grafik di mana setiap simpul sesuai dengan jumlah uang, dan ujung-ujungnya menunjukkan bagaimana koin dapat dipilih. Sebagai contoh, Gambar 3.9 menunjukkan grafik untuk koin {1,3,4} dan n = 6. Dengan menggunakan representasi ini, jalur terpendek dari node 0 ke node n sesuai dengan solusi dengan jumlah koin minimum, dan jumlah total jalur dari simpul 0 ke simpul n sama dengan jumlah total solusi.

BAB 4

PENGENALAN PEMROGRAMAN VISUAL STUDIO (VB.NET)

4.1 Pengantar

1. Perangkat lunak/software visual studio (VB.NET) merupakan software untuk pemrograman.
2. Dalam pemrograman ada istilah “Bahasa pemrograman” yang merupakan perintah yang ditujukan untuk komputer supaya mengeksekusi/melakukan tugas dari perintah tersebut.
3. VB.Net mulai di kembangkan tahun 1991 oleh perusahaan/vendor microsoft.
4. VB.NET adalah pengembangan dati bahasan BASIC (Beginners All Purpose Symbolic Intruction Code) yang merupakan bahasa awal dari VB.NET
5. Beginners All Purpose Symbolic Intruction Code dibuat oleh Professor John Kemeny dan Thomas Kurtz dari Kampus Darmouth pada pertengahan tahun 1960-an (Deitel & Deitel, 1999)
6. Graphical User Interface (GUI) dibuat dengan berbagai cara yang disebut sebagai VISUAL.
7. Kode baris tidak perlu dituliskan saat kita hendak melakukan instruksi pemrograman.
8. Objek yang akan kita gunakan hanya perlu di “drag” lalu “drop”
9. Bahasa Visual Basic merupakan satu dari berbagai bahasa pemrograman komputer yang sudah mendukung object (Pemrograman Berorientasi Objek /Object Oriented Programming = OOP)

4.2 Konsep Pemrograman Berbasis Visual

Program berbasis visual memakai konsep event – driven dimana Kode program tidak mengikuti alur yang ditetapkan awal dan Eksekusi program dapat berlainan sesuai event yang diberikan. Urutan event menentukan urutan kode yang dieksekusi, jadi alur jalannya program bisa berbeda untuk setiap eksekusi program. Penulisan program banyak dilakukan dengan berbagai editor, misal: Notepad. Dengan menggunakan IDE, Programmer dapat membuat user interface, melakukan koding, melakukan testing dan debugging serta mengkompilasi program menjadi executable.

Visual Basic merupakan bahasa pemrograman yang sangat mudah dipelajari, dengan teknik pemrograman visual yang memungkinkan penggunanya untuk berkreasi lebih baik dalam menghasilkan suatu program aplikasi. Ini terlihat dari dasar pembuatan dalam visual basic adalah FORM, dimana pengguna dapat mengatur tampilan form kemudian dijalankan dalam script yang sangat mudah. Hingga saat ini, Visual Basic sudah hadir dalam 10 versi.

Berikut peluncuran dari masingmasing versi.

1. Pada tahun 1991 => Microsoft Visual Basic Versi 1.0
2. Pada tahun 1992 => Microsoft Visual Basic Versi 2.0
3. Pada tahun 1993 => Microsoft Visual Basic Versi 3.0
4. Pada tahun 1996 => Microsoft Visual Basic Versi 4.0
5. Pada tahun 1997 => Microsoft Visual Basic Versi 5.0
6. Pada tahun 1998 => Microsoft Visual Basic Versi 6.0
7. Pada tahun 2003 => Microsoft Visual Basic Versi 7.0
8. Pada tahun 2005 => Microsoft Visual Basic Versi 8.0
9. Pada tahun 2008 => Microsoft Visual Basic Versi 9.0
10. Pada tahun 2010 => Microsoft Visual Basic Versi 10.0

4.3 Visual Basic .NET

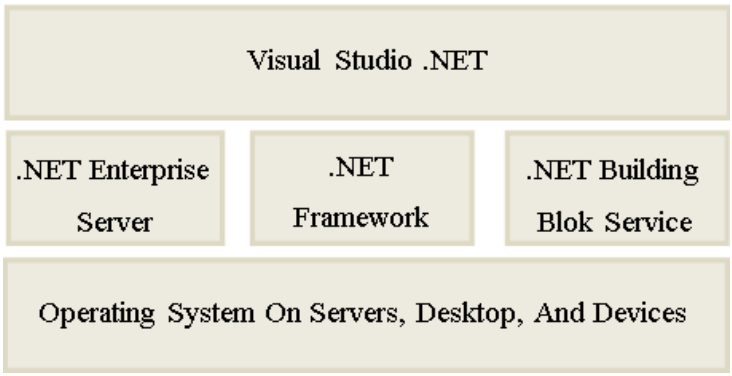
VB .Net adalah salah satu bahasa pemrograman dalam .Net framework. Cikal bakal dari VB.Net adalah bahasa BASIC (Beginer All-Purpose Symbolic Instruction Code) yang diciptakan tahun 1964 oleh professor John Kemeny dan Thomas Kurtz

Microsoft .Net : yang awalnya disebut Next Generation Windows Services (NGWS) adalah suatu platform untuk membangun dan menjalankan generasi penerus aplikasi-aplikasi. Microsoft.NET merupakan framework (kerangka) pengembangan yang menyediakan antarmuka pemrograman baru untuk layanan Windows dan API (Application Programming Interface)

Microsoft .NET merupakan strategi Microsoft untuk menghubungkan sistem, informasi, dan alat (device), sehingga orang dapat berkomunikasi serta berkolaborasi dengan lebih efektif.

Teknologi .NET terintegrasi penuh melalui produk-produk Microsoft, dan menyediakan kemampuan untuk mengembangkan solusi dengan menggunakan Web service.

Platform Microsoft .NET terdiri dari lima komponen utama yang tersusun dalam tiga lapisan (layer). Lapisan paling bawah adalah sistem operasi; lapisan kedua terdiri dari tiga komponen; lapisan teratas adalah Visual Studio .NET.

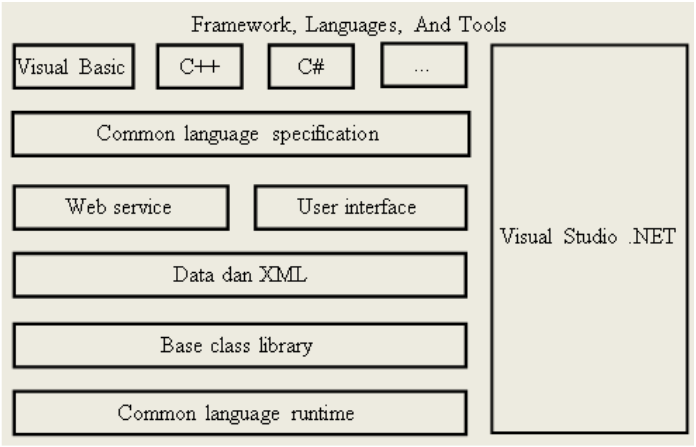


Gambar 4.1 Platform Microsoft .NET

Beberapa keuntungan ketika menggunakan .Net, adalah sebagai berikut:

1. Multi Language

Arsitektur .NET bersifat terbuka, sehingga memungkinkan berbagai bahasa pemrograman mengakses CLR dengan mulus. Banyak kalangan menyebut .NET sebagai "open source" versi Microsoft. Saat ini .NET dapat diprogram menggunakan Visual Basic.NET, C++.NET, Visual C#, Jscript, dan J#.



Gambar 4.2 Arsitektur Visual Basic .Net

2. No DLL Hell

DLL merupakan blok atau modul-modul obyek dari sebuah aplikasi. Peranannya sangat penting, sekaligus memusingkan. Sering terjadi dalam dunia windows, kompatibilitas dan registrasi DLL dimasing-masing Workstation menjadi isu besar dalam deployment aplikasi

3. Strong Typing dan Type Safety

.NET menyediakan strong typing, dimana setiap variabel wajib didefiniskan scope dan tipe datanya. Demikian pula dengan fasilitas type safety yang sangat bermanfaat untuk membantu dalam coding pemrograman, terutama fasilitas intellisense yang membimbing pemrogram dalam menentukan property, method, maupun function yang akan dipakai.

4. Cross Platform Possibility

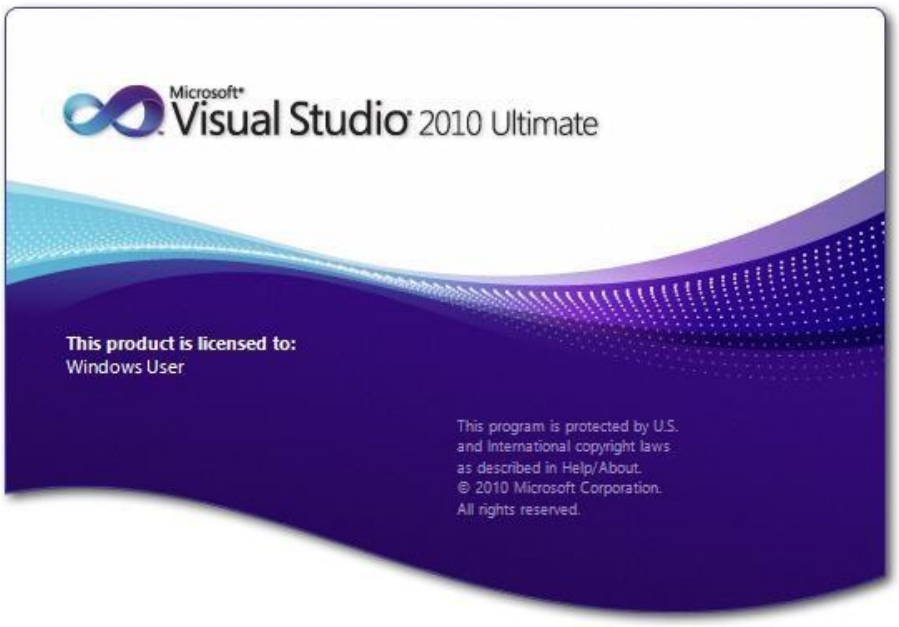
.Net menyimpan dan mengirim data dalam bentuk XML yang merupakan format data universal di internet. Dengan demikian integrasi data antar platform lebih mudah dilakukan, selama platform tersebut mendukung XML. Manipulasi format data dalam bentuk XML, .txt, maupun .rtf merupakan sesuatu yang menantang para programmer untuk membuat aplikasi lintas platform.

5. Code Once, More Application

Interface pemrograman bersifat konsisten, dengan object model yang sama pada setiap bahasa yang digunakan. Suatu object baik berbentuk class, library, maupun web services dapat diakses dengan mudah oleh berbagai aplikasi windows maupun web.

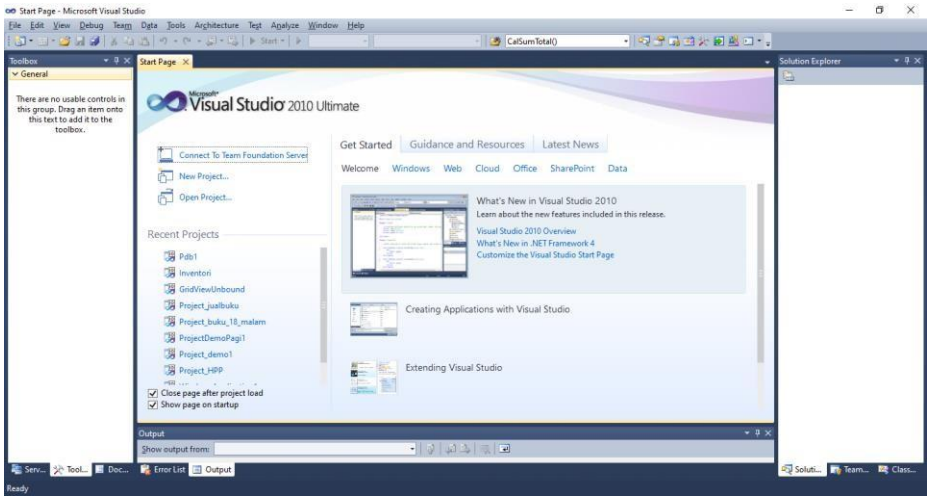
4.3.1 Memulai Pemrograman .NET

Tampilan aplikasi visual studio 2010 sebagai berikut :



Gambar 4.3 Splash Screen Visual Studio 2010

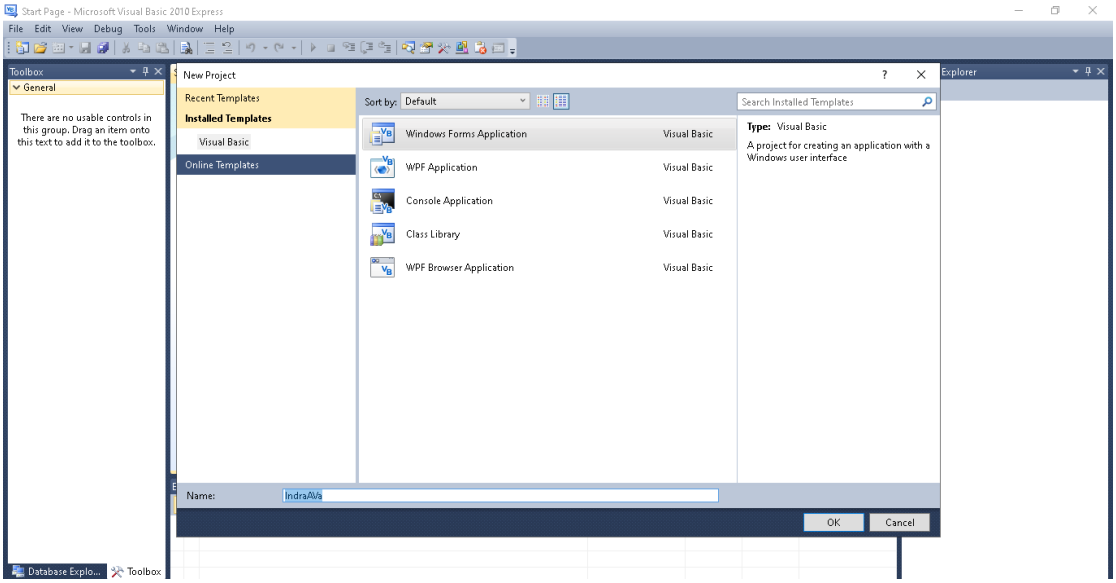
Setelah tampilan splash screen selanjutnya akan ditampilkan halaman Start Page Visual Studio 2010 sebagai berikut:



Gambar 4.4 Halaman Start Page Visual Studio 2010

Project merupakan kerangka dasar aplikasi yang menentukan jenis aplikasi yang akan dibuat. Langkah-langkah pembuatan project adalah sebagai berikut:

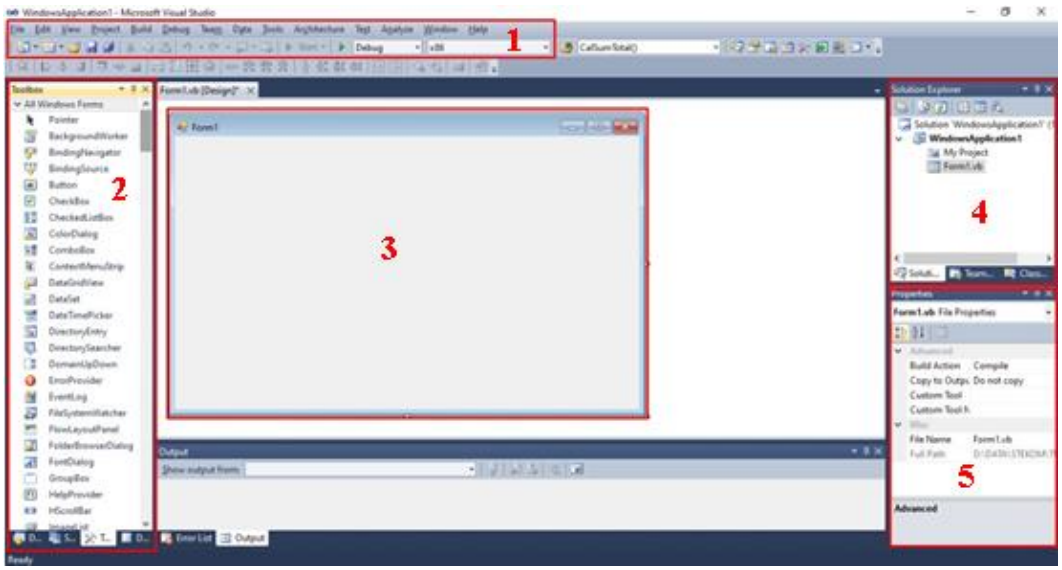
1. Jalankan Visual Basic 2010
2. Pada bagian Start Page pilih New Project
3. Pada bagian New Project pilih bahasa Visual Basic, Project Types: Windows dan Templates: Windows From Application
4. Klik OK



Gambar 4.5 Template Project Visual Studio 2010

	Jenis	Macam Template
	Windows	▪ Windows Form Application ▪ Class Library ▪ Windows Service ▪ dll
	Web	▪ ASP .NET Application ▪ ASP .NET Server Control ▪ WCF Service Application ▪ dll
	Smart Device	▪ Smart Device Project ▪ dll
	Database	▪ SQL Server Project

Dalam upaya memudahkan penggunaan IDE, maka diperlukan pemahaman yang baik mengenai IDE Visual Basic 2010.



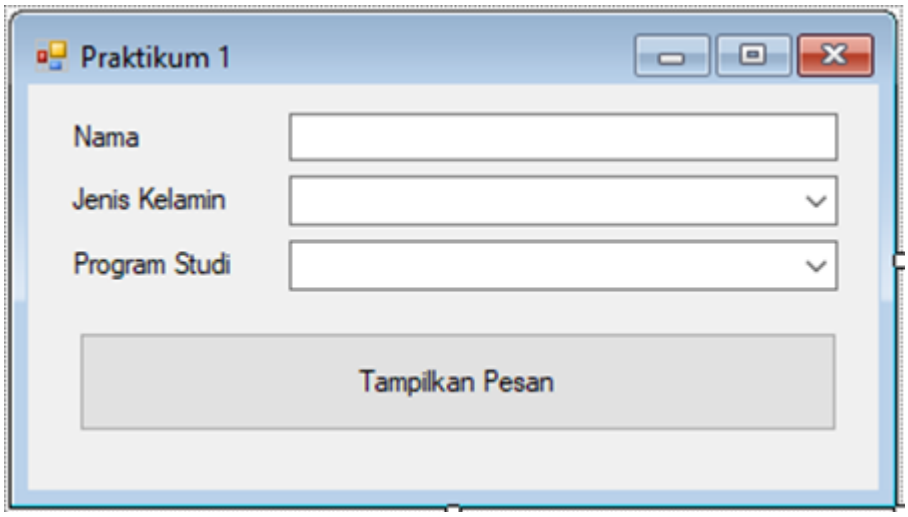
Gambar 4.6 Lembar Kerja Visual Studio 2010

Keterangan:

1. Menu Bar (menu standar visual basic)
2. Toolbox (daftar control yang ditambahkan ke dalam program sebagai interface)
3. Form Design (digunakan untuk mengedit tampilan form serta mengatur posisi control pada form)
4. Solution Explorer (digunakan untuk mengolah file dan project)
5. Properties (digunakan untuk mengedit dari form dan control yang sedang diedit)

Contoh Latihan

1. Buatlah sebuah Project Baru dengan nama Praktikum1_vbnet
2. Desainlah form sebagai berikut:



Gambar 4.7 Desain contoh latihan

Pengaturan Properti

No	Object	Property	Nilai
1	Form	Name Text	Frmpraktikum1 Praktikum 1
2	Label	Name Text	Label1 Nama
3	Label	Name Text	Label2 Jenis Kelamin

4	Label	Name Text	Label3 Program Studi
5	Textboxt	Name	Txtnama
6	Combobox	Name	Cbjeniskelamin
		Items DropDownStyle	Laki-laki Perempuan DropDownList
7	Combobox	Name Items DropDownStyle	Cbprogramstudi S1 - Teknik Informatika S1 - Sistem Informasi D4 - Komputerisasi Akuntansi D3 - Komputerisasi Akuntansi D4 - Manajemen Informatika D4 - Sistem Komputer DropDownList
8	Button	Name Text	Bttampilkan Tampilkan Pesan

Untuk Penulisan Kodingnya sebagai berikut :

```
Public Class Frmpraktikum1
    Private Sub Bttampilkan_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        MessageBox.Show(txtnama.Text & vbCrLf & cbjeniskelamin.Text & vbCrLf & cbprogramstudi.Text,
                        "Hasil Pengisian", MessageBoxButtons.OK, MessageBoxIcon.Information)
    End Sub
End Class
```

Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > Start Debugging

4.3.1.1 Variabel

Variabel adalah nama atau simbol yang digunakan untuk mewakili suatu nilai.Suatu variabel mempunyai nama dan menyimpan tipe data yang merupakan jenis data variabel.

Aturan penamaan variabel adalah sebagai berikut:

1. Harus dimulai dengan sebuah huruf
2. Tidak lebih dari 255 karakter
3. Tidak boleh sama dengan nama statement, fungsi, metode, objek, dan sebagainya yang merupakan bahasa dari Visual BASIC.
4. Tidak boleh ada spasi, tanda titik(.), tanda seru(!), atau karakter @, &, $, dan #.

Deklarasi variabel dapat dituliskan dengan urutan sebagai berikut:

Public <nama_variabel> As <Tipe_Data>

Atau

Dim <nama_variabel> As <Tipe_Data>

Contoh :

Public Angka1 As Integer Dim Nama As String

Tipe data adalah jenis data yang disimpan dalam variabel. Tipe data untuk Visual BASIC adalah sebagai berikut:

1. Tipe Data Numerik : digunakan untuk menyimpan data numerik, terdiri dari:

Tipe Data	Ukuran	Range
Byte	1 byte	0 sampai 255
Integer	2 byte	-32.768 sampai 32.767
Long	4 byte	-2.147.483.648 sampai 2.147.483.647
Single	4 byte	-3,402823E38 sampai -1,401298E-45; 1,401298E-45 sampai 3,402823E38

Double	8 byte	-1.79769313486232E308 sampai -4,94065645841247E-324; 4,94065645841247E-324 sampai 1.79769313486232E308
Currency	8 byte	-922.337.203.685.477,5808 sampai 922.337.203.685.477,5807

2. Tipe Data String: digunakan untuk menyimpan data berbentuk karakter.Panjang maksimal karakter yang dapat disimpan adalah 65.400 karakter. Penulisan data dengan tipe ini diawali dan diakhiri dengan tanda petik dua (“).

   Contoh:

   Dim Nama As String Nama = “Dewi”

3. Tipe Data Logika (Boolean): melakukan pengetesan logika. Data dengan tipe data ini hanya dapat bernilai benar (True) atau salah (False).

   Contoh:

   Dim Baru As Boolean Baru = True

4.3.1.2 Konstanta

Konstanta adalah suatu nilai konstan yang tidak berubah. Seperti halnya variabel, konstanta dapat diberi nama dimana aturan penamaannya sama dengan variabel.

Contoh:

Const A = 10

4.3.1.3 Operator

1. Operator Nilai Pemberi

   Deklarasi pemberian nilai pada Visual BASIC = Bahasa BASIC yaitu menggunakan operator sama dengan (=).

   Contoh :

   a = 24

   nama = “Fery Updi”

2. Operator Aritmatika

Operator	Operasi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
\	Pembagian dengan hasil bilangan bulat
Mod	Sisa pembagian (Modulus)

3. Operator Boolean

Operator	Operasi
Not	**Negasi**
And	**Logika and**
Or	**Logika or**
Xor	**Logika xor**

4. Operator Pembanding

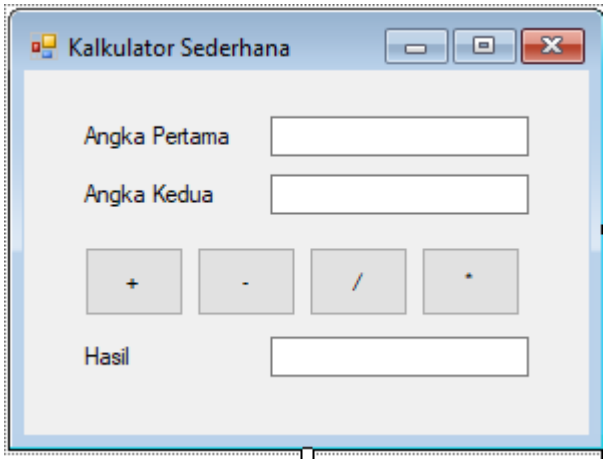
Operator	Operasi
=	Sama dengan
<>	Tidak sama dengan
<	Kurang dari
>	Lebih dari
<=	Kurang dari atau sama dengan
>=	Lebih dari atau sama dengan

5. Derajat Operator

Operator	Operasi
Not	Tertinggi
* / \ mod and	
+ - or xor	
= <> <= >=	Terendah

Latihan Praktikum

1. Buatlah Form baru dengan nama frmkalkulator
2. Buatlah sebuah kalkulator sederhana yang bisa melakukan operasi penambahan, pengurangan, pembagian dan perkalian antara dua buah bilangan yang diinputkan oleh user.
3. Desain form kalkulator sederhana ini kurang lebih sebagai berikut:



Gambar 4.8 Program Kalkulator Sederhana

Pengaturan Properti form

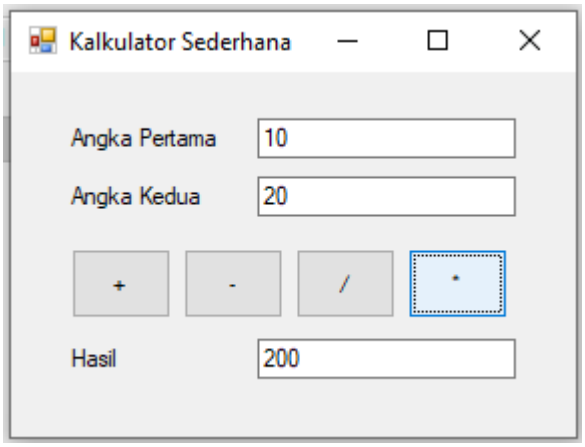
No	Object	Property	Nilai
1	Form	Name Text	Frmkalkulator Kalkulator Sederhana
2	Label	Name Text	Label1 Angka Pertama
3	Label	Name Text	Label2 Angka Kedua
4	Label	Name Text	Label3 Hasil
5	Textboxt	Name	Txtangka1
6	Textboxt	Name	Txtangka2
7	Textboxt	Name	Txthasil
8	Button	Name Text	btjumlah +

9	Button	Name Text	btkurang -
10	Button	Name Text	btbagi /
11	Button	Name Text	btkali *

Tuliskan kode programnya sebagai berikut :

```
Public Class Frmkalkulator

    Private Sub btjumlah_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        Dim angka1, angka2, hasil As Double
        angka1 = Val(txtangka1.Text)
        angka2 = Val(txtangka2.Text)
        hasil = angka1 + angka2
        txthasil.Text = hasil
    End Sub

    Private Sub btkurang_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        Dim angka1, angka2, hasil As Double
        angka1 = Val(txtangka1.Text)
        angka2 = Val(txtangka2.Text)
        hasil = angka1 - angka2
        txthasil.Text = hasil
    End Sub

    Private Sub btbagi_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        Dim angka1, angka2, hasil As Double
        angka1 = Val(txtangka1.Text)
        angka2 = Val(txtangka2.Text)
        hasil = angka1 / angka2
        txthasil.Text = hasil
    End Sub

    Private Sub btkali_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        Dim angka1, angka2, hasil As Double
        angka1 = Val(txtangka1.Text)
        angka2 = Val(txtangka2.Text)
        hasil = angka1 * angka2
        txthasil.Text = hasil
    End Sub
End Class
```

Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > Start Debugging



Gambar 4.9 Hasil Latihan Praktikum

4.3.1.4 Fungsi Percabangan

Pada beberapa kasus terkadang kita menginginkan komputer melakukan suatu pernyataan tertentu bila suatu kondisi terpenuhi. Dalam Visual Basic .NET perintah percabangan/pemilihan keputusan dapat dilakukan dengan statemen If...Then dan Select Case.

Beberapa pernyataan If … Then

1. Statemen If...Then

Statemen ini digunakan untuk melakukan aksi setelah melakukan pengujian terhadap suatu kondisi. Pernyataan dalam blok statemen hanya akan dilaksanakan ketika kondisi pengetesan/pengujian bernilai benar. Statement If...Then memiliki beberapa sintaks/cara penulisan sesuai dengan jumlah pernyataan yang akan dieksekusi.

a. If...Then dengan Kondisi dan Pernyataan Tunggal

If <kondisi> Then <Pernyataan>

Contoh:

If Nilai >= 60 Then Keterangan = “Lulus”

b. If...Then dengan Pernyataan Jamak

If <Kondisi> Then

<Pernyataan_1>

<Pernyataan_2>

```
..
<Pernyataan_n>
End If
Contoh :
If Nilai >= 60 Then
    Keterangan = “Lulus”
    Ucapan = “Selamat”
End If
```

c. If... Then dengan 2 kondisi.

```
If <Kondisi> Then
    <Pernyataan_Jika_Kondisi_Benar>
Else
    <Pernyataan_Jika_Kondisi_Salah>
End If
Contoh:

If Nilai >= 60 Then
    Keterangan = “Lulus”
    Ucapan = “Selamat”
Else
    Keterangan = “Tidak Lulus”
    Ucapan = “Jangan Bersedih”
End If
```

d. If...Then dengan kondisi jamak.

```
If <Kondisi_1> Then
        <Pernyataan>
ElseIf <Kondisi_2>
Then
        <Pernyataan>
        ...
ElseIf <Kondisi_n> Then
        <Pernyataan>
Else
        <Pernyataan>
End If
```

Contoh:

```
If Nilai >= 85 Then
        Keterangan = “Lulus, Sangat Memuaskan”ElseIf
Nilai >= 70 Then
        Keterangan = “Lulus, Memuaskan”ElseIf
Nilai >=60 Then
        Keterangan = “Lulus, Cukup Memuaskan”
        Keterangan = “Tidak Lulus, Silahkan
        Mengulang”
End If
```

2. Statement Select ... Case

Sama halnya seperti statemen **If...Then**, **Select Case** juga mengerjakan suatu blok statemen berdasarkan uji nilai ekspresi. Perbedaannya adalah pada tata cara penulisan dan pengelompokan nilai dari variabel/kondisi.

Select Case <Variabel penguji>
Case <Nilai_1>

<Pernyataan_1>
Case <Nilai_2>

<Pernyataan_2>
Case Else

<Pernyataan_n>
End Select

Contoh:

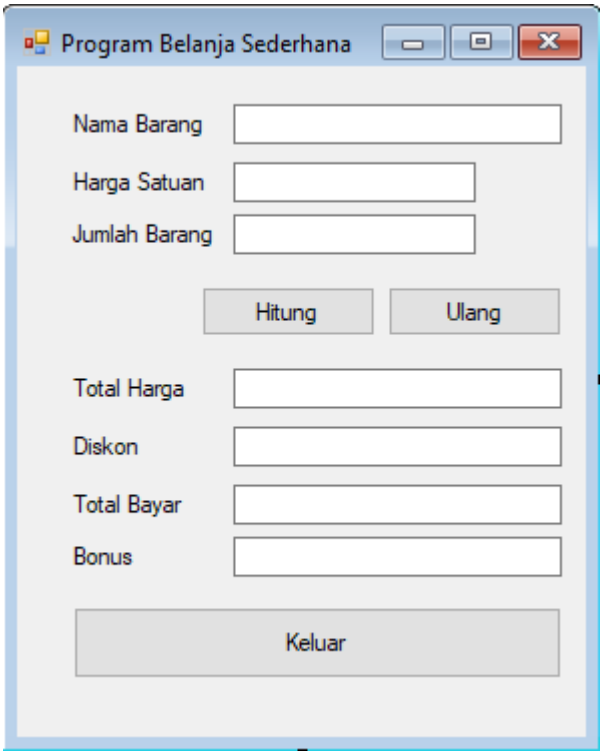
```
Select Case
        Nilai
  Case “A”
```

***Sintaks**:*

```
        Keterangan = “Sangat Memuaskan”Case “B”
        Keterangan = “Memuaskan”
 Case “C”
        Keterangan = “Cukup”
 Case Else
        Keterangan = “Kurang”
End Select
```

Latihan Praktikum

Buatlah sebuah form baru pada Visual BASIC .NET, desain tampilan form sehingga didapat tampilan seperti berikut:



Gambar 4.10 Desain form praktikum percabangan

Pengaturan pada form properti sebagai berikut :

No	Object	Property	Nilai
1	Form	Name Text	Frmbelanja Program Belanja Sederhana
2	Label	Name Text	Label1 Nama Barang
3	Label	Name Text	Label2 Harga Satua
4	Label	Name Text	Label3 Jumlah Barang
5	Label	Name Text	Label4 Total Harga

6	Label	Name Text	Label5 Diskon
7	Label	Name Text	Label6 Total Bayar
8	Label	Name Text	Label7 Bonus
9	Textbox	Name	Txtnabar
10	Textbox	Name	Txtharga
11	Textbox	Name	Txtjumlah
12	Textbox	Name	Txttotalharga
13	Textbox	Name	Txtdiskon
14	Textbox	Name	Txttotalbayar
15	Textbox	Name	Txtbonus
16	Button	Name Text	Bthitung Hitung
17	Button	Name	Btulang
		Text	Ulang
18	Button	Name	Btkeluar
		Text	Keluar

Pada program diatas perhitungan dilakukan dengan kriteria sebagai berikut:

Total Harga = Harga Satuan x Jumlah Barang Diskon dan Bonus, didapat dengan ketentuan:

Total Harga	Diskon	Bonus
>=500rb	20%	Tas Pinggang
200rb – 500rb	15%	Payung
100rb – 200rb	10%	Kaos
50rb – 100rb	5%	Pena
<50rb	0	Tiidak Ada

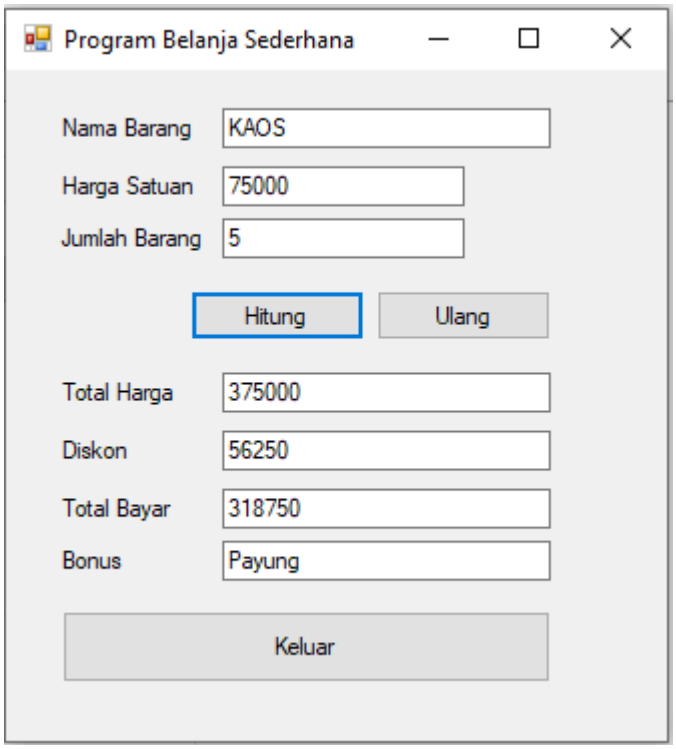
Total Bayar = Total Harga – Diskon

Tuliskan kode berikut ini:

```
Private Sub btulang_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    'membersihkan isi textbox
    txtnabar.Clear()
    txtharga.Clear()
    txtjumlah.Clear()
    txttotalharga.Clear()
    txtdiskon.Clear()
    txttotalbayar.Clear()
    txtbonus.Clear()
End Sub
```

```
Private Sub bthitung_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    'deklarasi variable
    Dim harga, jumlah As Integer
    Dim total, diskon, bayar As Double
    Dim bonus As String
    harga = txtharga.Text
    jumlah = txtjumlah.Text

    total = harga * jumlah
    'penentuan diskon
    If total >= 500000 Then
        diskon = 0.2 * total
        bonus = "Tas Pinggang"
    ElseIf total >= 200000 Then
        diskon = 0.15 * total
        bonus = "Payung"
    ElseIf total >= 100000 Then
        diskon = 0.1 * total
        bonus = "Kaos"
    ElseIf total >= 50000 Then
        diskon = 0.05 * total
        bonus = "Pena"
    Else
        diskon = 0
        bonus = "Tidak Ada"
    End If
    'hitung total bayar
    bayar = total - diskon
    'tampilkan total harga, diskon, total bayar, dan bonus
    txttotalharga.Text = total
    txtdiskon.Text = diskon
    txttotalbayar.Text = bayar
    txtbonus.Text = bonus
End Sub
Private Sub btkeluar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    End
End Sub
```

Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > StartDebugging



Gambar 4.11 Hasil Program

4.3.1.5 Fungsi Perulangan

Proses perulangan dalam pemrograman dilakukan untuk mengerjakan suatu proses operasi secara bertahap demi tahap dengan nilai variabel yang menaik atau menurun. Dalam Visual Basic .NET proses perulangan dapat dilakukan dengan beberapa statemen, diantaranya adalah statemen For...Next dan Do...Loop.

1. For...Next

Statemen ini akan mengulangi suatu blok pernyataan sebanyak jumlah yang ditentukan. Statemen ini digunakan jika banyaknya jumlah perulangan sudah diketahui.

Sintaks:

```
For <Variabel_Pengulang> = NilaiAwal To NilaiAkhir [Step Tingkat]
<Pernyataan_1>
...
<Pernyataan_n>
Next <Variabel_Pengulang>
```

Statemen ini digunakan untuk kondisi yang mempunyai nilai berurutan dan variable yang mempunyai nilai numerik. Default untuk Step adalah 1, jadi untuk perulangan dengan urutan menaik 1, nilai step tidak perlu ditulis. Sedangkan untuk perulangan menurun (Ni lai awal > Nilai Akhir), nilai step diawali dengan tanda minus(-).

Misalnya :

For i = 10 To 1 Step -1.

Contoh:

Untuk mencetak angka 1 sampai 10 secara berurutan pada objek ListBox dapat dilakukan dengan memberi listing program sebagai berikut:

```
For i = 1 To 10
List1.Additem  i
Next i
```

2. Do ... Loop

Statemen ini mengulang blok statemen bila kondisi benar atau sampai kondisi menjadi benar. Bila tidak ada perintah keluar, proses perulangan (loop) akan terus berlangsung. Statemen ini digunakan untuk kondisi yang mempunyai nilai tidak pasti dan tidak berurutan.

Statemen ini memiliki dua buah bentuk logika:

a. Statemen Do...Loop...While

Statemen ini akan mengerjakan pernyataan dalam blok statemen ketika kondisi bernilai benar, dan akan berhenti ketika kondisi sudah bernilai salah.

Sintaks:

```
Do While <Kondisi>                    Do
        <Pernyataan_1>                        <Pernyataan_1>
        ...                  Atau             ...
        <Pernyataan_n>                        <Pernyataan_n>
                                      Loop While <Kondisi>
Loop
```

Contoh: Untuk mencetak angka 1 sampai 10 secara berurutan pada ListBox dapat dilakukan dengan memberi listing program sebagai berikut:

```
i = 1
Do While i <= 10
    List1.Additem i
    i = i + 1
Loop
```

b. Statement Do…Loop…Until

Statemen ini akan mengerjakan pernyataan dalam blok statemen ketika kondisi bernilai salah, dan akan berhenti ketika kondisi mencapai nilai benar.

Sintaks:

```
Do Until <Kondisi>
        <Pernyataan_1>
        ...
        <Pernyataan_n>
Loop
```

Atau

```
Do
        <Pernyataan_1>
        ...
        <Pernyataan_n>
Loop Until <Kondisi>
```

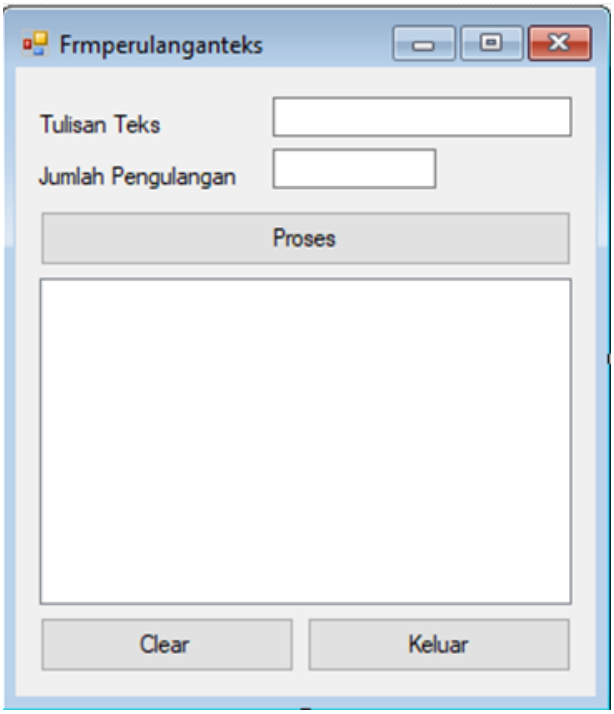
Contoh:

Untuk mencetak angka 1 sampai 10 secara berurutan pada objek ListBox dapat dilakukan dengan memberi listing program sebagai berikut:

```
i = 1
Do
    List1.Additem i
    i = i + 1
    Loop Until i > 10
```

Latihan Praktikum

Buatlah sebuah form baru pada Visual BASIC .NET, desain tampilan form sehingga didapat tampilan seperti pada gambar berikut :



Gambar 4.12 Desain Form Perulangan

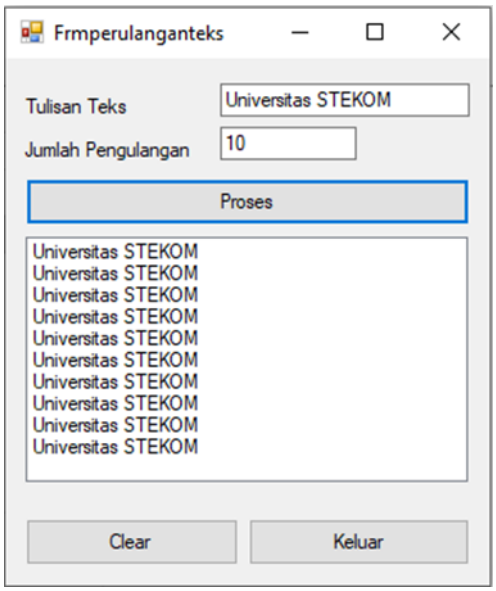
Aturlah properti object sebagai berikut:

No	Object	Property	Nilai
1	Form	Name Text	Frmpengulanganteks Frmpengulanganteks
2	Label	Name Teks	Label1 Tulisan Teks
3	Label	Name Teks	Label2 Jumlah pengulangan
4	Textboxt	Name	Txtteks
5	Textboxt	Name	Txtjumlah
6	Listbox	Name	Listbox1
7	Button	Name Text	Bproses Proses
8	Button	Name Text	Bclear Clear
9	Button	Name Text	Bkeluar Keluar

Untuk penulisan koding sebagai berikut :

```
Private Sub bkeluar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    End
End Sub

Private Sub bclear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) H
    txtteks.Clear()
    txtjumlah.Clear()
    ListBox1.Items.Clear()
End Sub

Private Sub bproses_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Dim teks As String
    Dim jumlah As Integer
    teks = txtteks.Text
    jumlah = txtjumlah.Text
    For i = 1 To jumlah
        ListBox1.Items.Add(teks)
    Next
End Sub
```

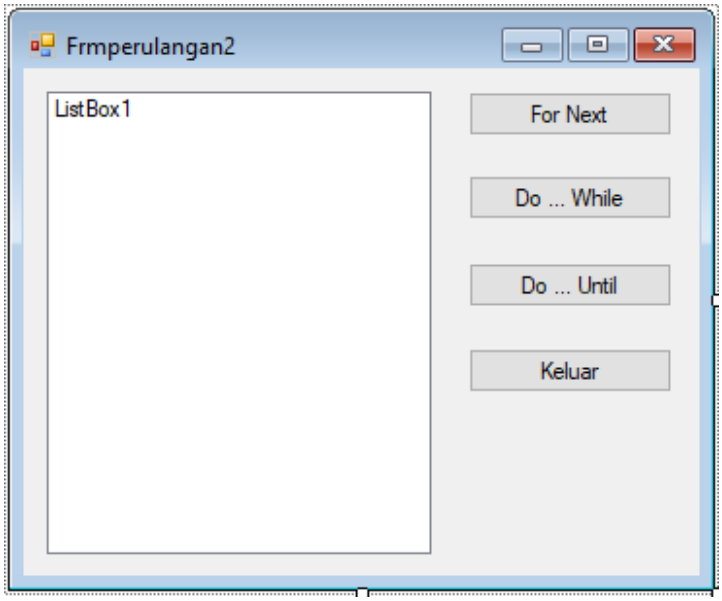
Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > Start Debugging



Gambar 4.13 Hasil Form Perulangan

Latihan Praktikum perulangan 2

Buatlah sebuah form baru pada Visual BASIC .NET, desain tampilan form sehingga didapat tampilan seperti pada gambar



Gambar 4.14 Desain form perulangan 2

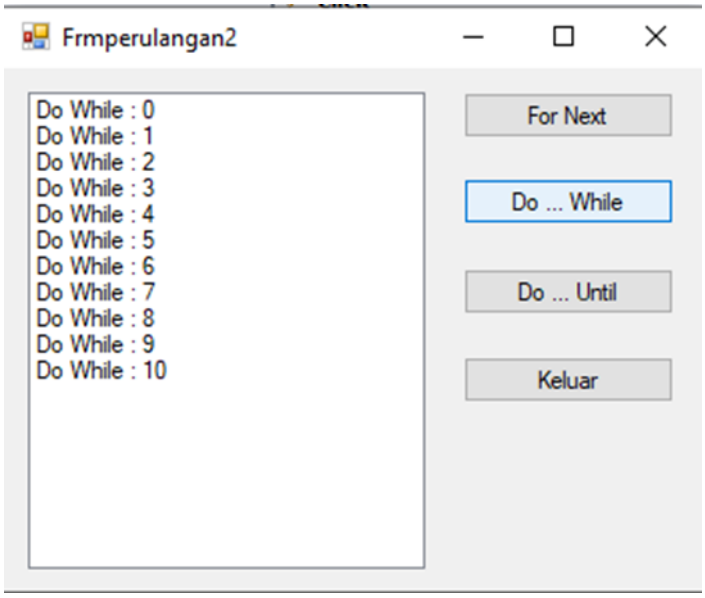
Aturlah properti object sebagai berikut :

No	Object	Property	Nilai
1	Form	Name Teks	Frmperulangan2 Frmperulangan2
2	Listbox	Name	Listbox1
3	Button	Name Text	Bfor For … Next
4	Button	Name Text	Bwhile Do … While
5	Button	Name Text	Buntil Do … Until
6	Button	Name Text	Bkeluar Keluar

Tuliskan Koding program sebagai berikut :

```
Private Sub bfor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) H
    ListBox1.Items.Clear()
    For i = 1 To 10
        ListBox1.Items.Add("For Next : " & i)
    Next
End Sub

Private Sub bwhile_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ListBox1.Items.Clear()
    Dim i As Integer = 0
    Do While i <= 10
        ListBox1.Items.Add("Do While : " & i)
        i = i + 1
    Loop
End Sub

Private Sub buntil_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) H
    ListBox1.Items.Clear()
    Dim i As Integer = 0
    Do Until i > 10
        ListBox1.Items.Add("Do Until : " & i)
        i = i + 1
    Loop
End Sub

Private Sub bkeluar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    End
End Sub
```

Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > Start Debugging



Gambar 4.15 Hasil Program Form Perulangan 2

4.3.1.6 Array

Array adalah sekumpulan data yang memiliki tipe yang sama, sejumlah tetap, serta disusun secara terstruktur dan disimpan dalam satu variabel yang sama, dan diurutkan dengan index

1. Array Satu Dimensi

Suatu array yang nilai dan ukurannya yang sudah ditentukan terlebih dahulu, dan memiliki satu dimensi

Bentuk umum

Dim Array[Indeks] As Tipe_Data

Contoh

Dim Arr(2) As String Arr(0) = TxtNim.Text Arr(1) = TxtNama.Text Arr(2) = TxtProdi.Text

2. Array Multi Dimensi

Suatu array yang fungsinya hampir sama dengan array satu dimensi hanya saja pada array multi dimensi ini mewakili nilai table yang terdiri dari informasi yang diatur dalam ba ris dan kolom. Untuk mendefenisikan elemen table tertentu, kita harus menentukan dua indeks, pertama mengidentifikasi elemen baris dan yang mengidentifikasi elemen kolom. Array multidimensi memiliki lebih dari dua dimensi.

Bentuk umum

Dim Array[indeks,Indeks] As Tipe_Data

Dim Array[indeks,indeks,indeks] As Tipe_Data

Contoh

Dim Arr(3, 1) As String

Arr(0, 0) = "NIM"

Arr(0, 1) = "NAMA"

Arr(1, 0) = "JENIS KELAMIN" Arr(1, 1) = "PRODI"

Arr(2, 0) = "Laki-laki" Arr(2, 1) = "Perempuan"

Arr(3, 0) = "Manajemen Informatika" Arr(3, 1) = "Teknik Informatika"

4.3.1.7 Procedure, Function dan Module

1. Procedure

Sejauh ini, anda telah menuliskan sub procedure yang dibuat secara otomatis pada saat menggunakan event . Sub procedure merupakan blok kode yang mempunyai nama dan berisi perintah yang dapat dipanggil suatu waktu. Artinya, hanya sekali dibuat yang kemudian di panggil berkali-kali sesuai dengan kebutuhan.

Bentuk Umum

[Private|Public] Sub subname[(argumentlist)]

Statements

End sub

Sub Procedure dapat dipanggil dengan perintah :

[call] subname[(argumentlist)]

Keterangan:

Sub procedure (Procedure) sebaiknya menggunakan kata kerja (verb) Pemanggilan procedure yang masih dalam satu kelas (class) atau modul (module) dapat dilakukan dengan kata kunci me.

Procedure dapat menerima argument melalui nilai (by value) a tau juga reference (by reference). Jika menggunakan by value, nilai pada argument tidak berubah. Sedangkan jika menggunakan by reference, nilai argument yang digunakan dapat berubah sesuai dengan perintah dalam procedure.

Secara default, argument dimasukkan melalui nilai (by value) dapat menggunakan kata kunci ByVal dan ByRef untuk By Reference.

Syntax argument:

[ByVal|ByRef] variabelname as type

Contoh:

```
Sub HitungVal(byVal as Decimal)
        A+=1
        textAkhir.text=a
end sub
```

a. Procedure dengan argument di passing sebagai value

b. Procedure dengan argument di passing sebagai value

```
Sub HitungRef(byRef as Decimal)
      A+=1
      textAkhir.text=a
        end sub
```

c. Pemanggilan procedure

```
HitungVal(textAwal.Text)
```

2. Function (Fungsi)

Fungsi mempunyai banyak kemiripan dengan sub procedure.

Bedanya, fungsi selalu mengembalikan nilai (return value)

Sintaks fungsi dituliskan sebagai berikut :

```
[Private|Public] function functionname[(argumentlist)] [As
    type]Statement
    Functionname==expression | {return expression}
          End function
```

Keterangan :

- Perbedaan antara fungsi dengan procedure hanyalah pada pengembalian nilai saja (return value). Sebagai contoh :

Menggunakan argument berupa nilai (by value) dan reference (by reference)

- Pengembalian nilai dapat dilakukan dengan menggunakan operator assignment "=" pada fungsi atau biasanya dengan perintah return.

```
Private Function cekData() as Boolean
        if IsNumeric(TextAwal.text)
        thenreturn true
        else
        return false
        endif
end function
```

Contoh:

```
if cekData()=true then
        HitungRef(textAwal.text)
else
        messageBox.Show("Data yang dimasukkan bukan
        angka", "Kesalahan data", MessageBoxButtons.OK,
        MessageBoxIcons.Error)
endif
```

Pemanggilan Fungsi:

3. Modul *(Module)*

Modul merupakan bagian yang sengaja dipisahkan untuk memudahkan pemrograman. Dalam module dapat dimasukkan procedure dan fungsi dan kemudian digunakan oleh beberapa form.

a. Umumnya modul dideklarasikan secara public (public) agar dapat digunakan di dalam kode dimanapun
b. Modul dapat diisi dengan main procedure yang akan dijala nkan pertama kali saat program dimulai
c. Modul merupakan salah satu argumentasi code reuse yang bertujuan agar program lebih efesien.

```
Module moduleName
        Statement

End Module
```

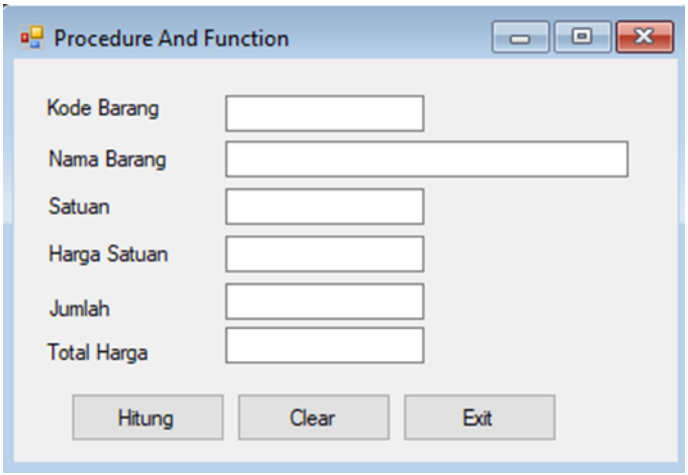
Sintaks pembuatan module sebagai berikut :

```
Module Module
        Public intResponce, RoleId, Msg As Integer
        Public Username, Password, Nama As String
End Module
```

Contoh

Latihan Praktikum

Buatlah sebuah form baru pada Visual BASIC .NET, desain tampilan form sehingga didapat tampilan seperti pada gambar



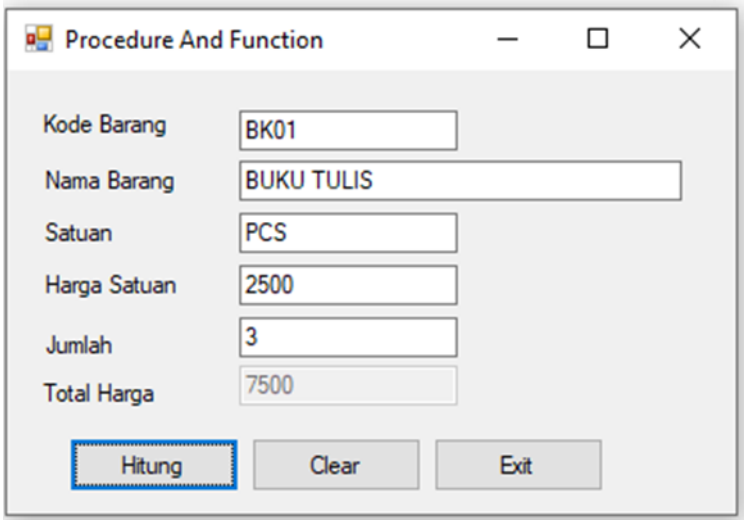
Gambar 4.16 Desain Praktikum Modul

Aturlah Properti object sebagai berikut :

No	Object	Property	Nilai
1	Form	Name Text	Frmprocedurefunction Procedure And Function
2	Label1	Text	Kode barang
	Label2	Text	Nama barang

	Label3	Text	Satuan
	Label4	Text	Harga satuan
	Label5	Text	Jumlah
	Label6	Text	Total harga
	Textbox1	Name	Txtkobar
	Textbox2	Name	Txtkobar
	Textbox3	Name	Txtnabar
	Textbox4	Name	Txtsatuan
	Textbox5	Name	Txtjumlah
	Textbox6	Name	Txttotharga
	Button1	Name	Bhitung
		Text	Hitung
	Button2	Name	Bclear
		Text	Clear
	Button3	Name	Bexit
		Text	Exit

Jalankan aplikasi dengan menekan tombol F5 atau melalui ikon Start Debugging di toolbar, atau melalui menu Debug > Start Debugging



Gambar 4.17 Hasil Program Procedure and Function

Bab 5

Implementasi Pemrograman-1

5.1 Normalisasi

a. Unnormalisasi

kode_produksi	kode_barang_masuk	kode_akun
tgl_produksi	Faktur	nama_akun
jumlahProduksi	tgl_masuk	kateg_perkiraan
satuan	kode_supplier	No_transaksi
JenisProduksi	keterangan	tgl_transaksi
deskripsi	no_transaksi	kode_akun_debit

hasil_produksi	kode_barang_masuk	kode_akun_kredit
tgl_selesai_produksi	kode_bahan	keterangan
input_petugas	qty	saldo
tgl_input	harga	input_user
kode_request	kode_produksi	tgl_input
tgl_request	jenis_produk	id_petugas
kode_produksi	qty_produksi	Nama_petugas
keterangan	tgl_selesai	Jabatan
input_petugas	harga_jual	Password
tgl_input	input_user	user_entry
kode_request	tgl_input	tgl_entry
Seq	no_penjualan	Kode_supplier
kode_bahan	tgl_penjualan	Nama_supplier
qty	kode_customer	Alamat
Kode_pengeluaran	totalPenjualan	kota
tgl_pengeluaran	totalPembayaran	Telp
kode_request	totalPembayaran	input_petugas
keterangan	tgl_input	tanggal_input
input_petugas	no_transaksi	Kode_customer
tgl_input	no_penjualan	Nama_customer
kode_pengeluaran	kode_produk	Alamat
kode_seq	qty_jual	kota
kode_bahan	hargasatuan	Telp
qtyKeluar	kode_kategori	

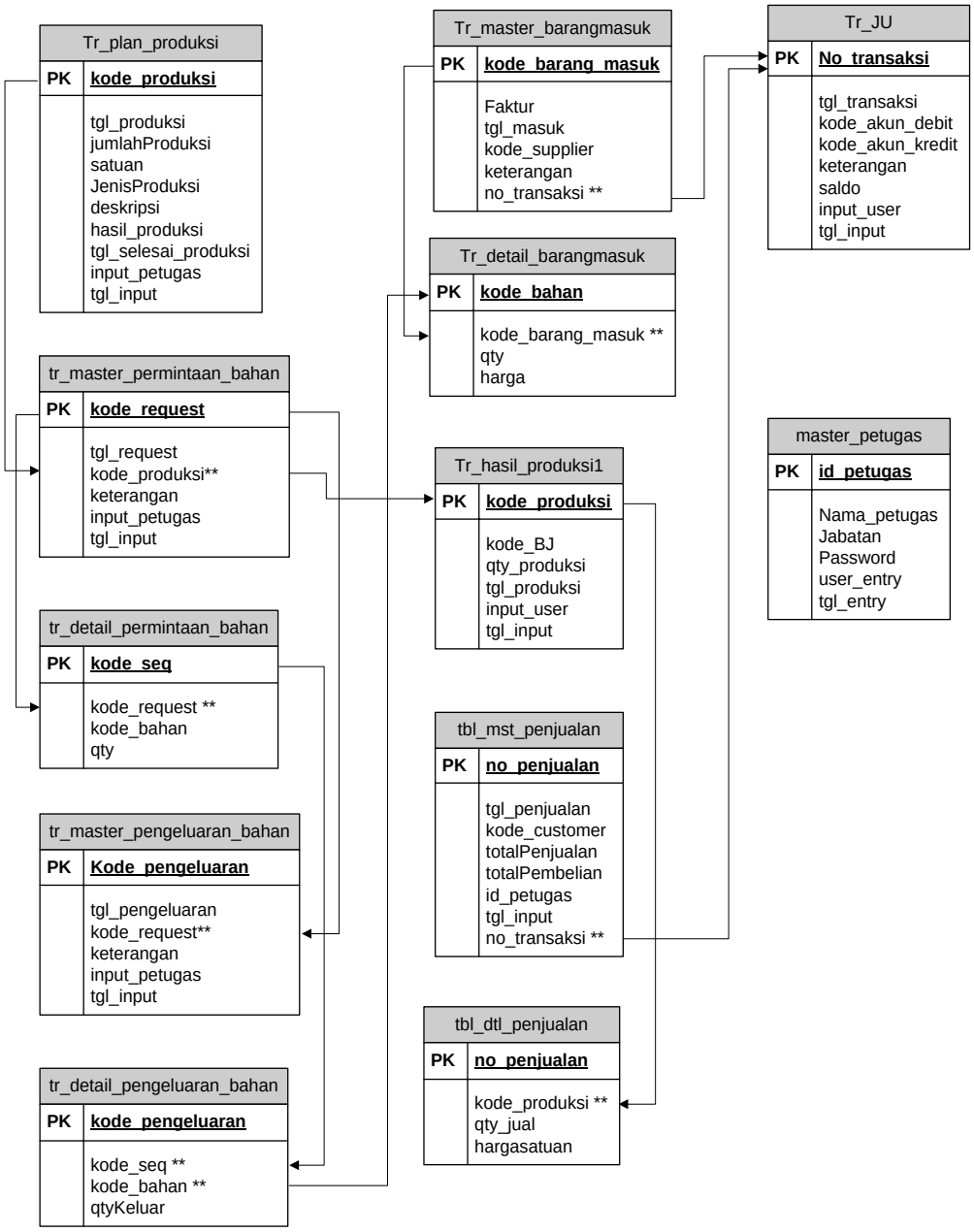
Gambar 5.1 Bentuk Unnormalisasi

b. Normalisasi 1NF

Tr_JU	
PK	**No_transaksi**
	tgl_transaksi kode_akun_debit kode_akun_kredit keterangan saldo input_user tgl_input

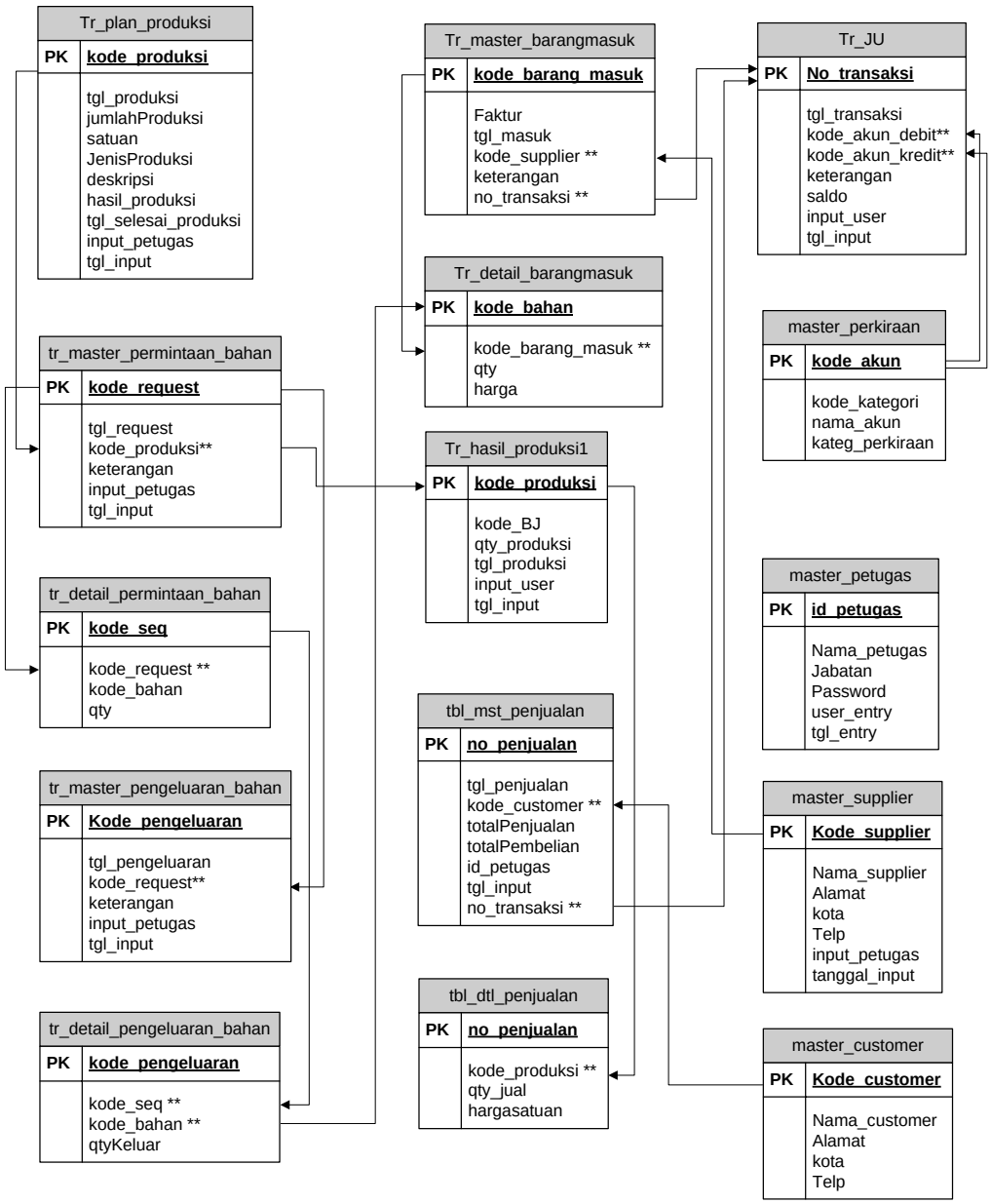
Gambar 5. 2 Bentuk Normalisasi 1NF

c. Normalisasi 2NF



Gambar 5.3 Bentuk Normalisasi 2NF

d. Normalisasi 3NF



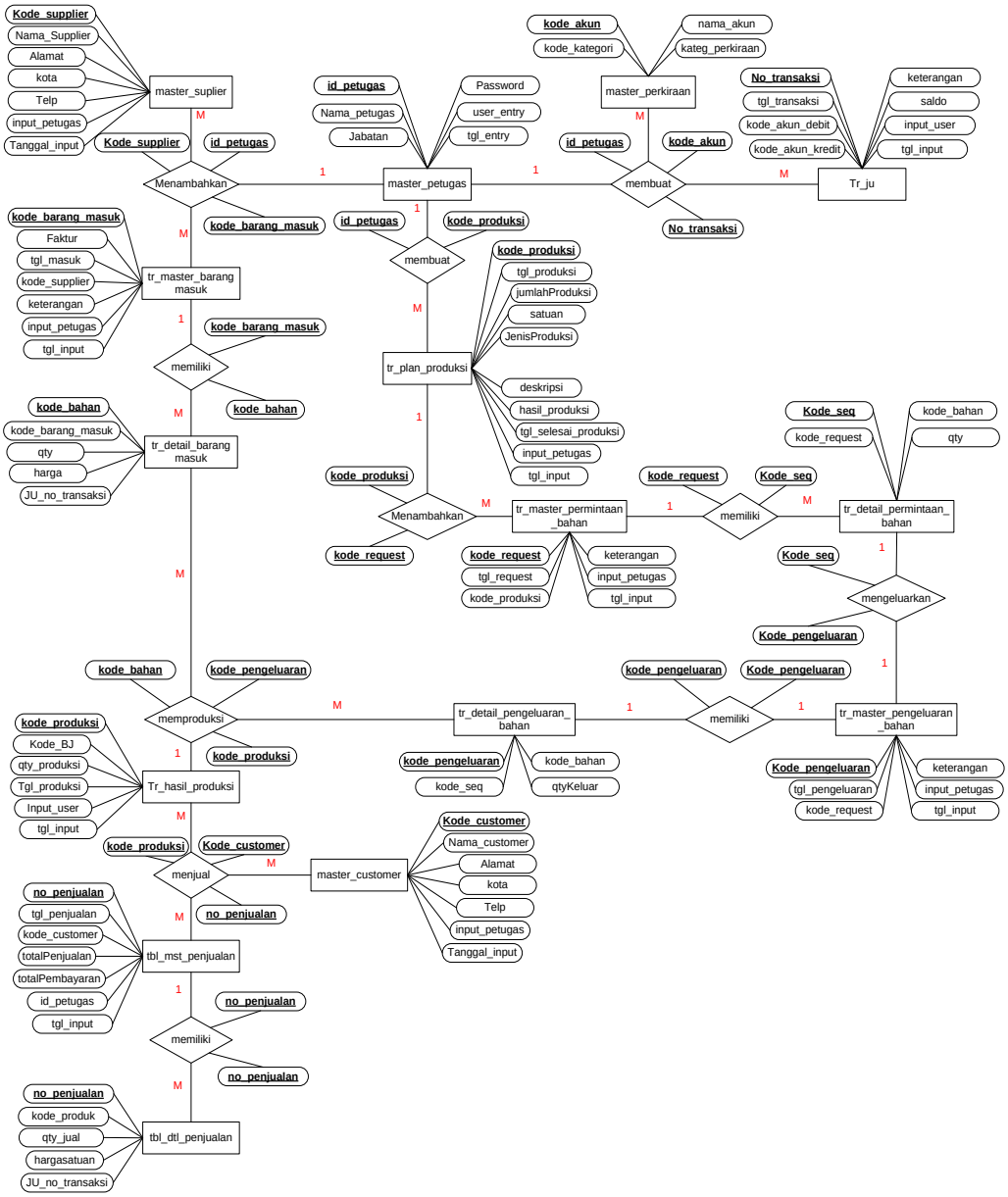
Gambar 5.4 Bentuk Normalisasi 3NF

Keterangan:

PK : Primary Key

** : Foreign Key

5.2 ERD (Entity Relationship Diagram)



Gambar 5.5 ERD (Entity Relationship Diagram)

Keterangan:

M: Many

Perancangan *Database*

a. Tabel Petugas

Nama tabel : master_petugas

Primary Key : id_petugas

Tabel 3. 1 Tabel Petugas

Field	***Type Text***	***Size***	**Keterangan**
id_petugas	*Varchar*	20	*Primary Key*
Nama_petugas	*Varchar*	100	
Jabatan	*Varchar*	20	
Password	*Varchar*	20	
user_entry	*Varchar*	20	
tgl_entry	*DateTime*		

b. Tabel *Supplier*

Nama tabel : master_supplier

Primary Key : Kode_supplier

Tabel 3. 2 Tabel Petugas

Field	***Type Text***	***Size***	**Keterangan**
Kode_supplier	*Varchar*	20	*Primary Key*
Nama_supplier	*Varchar*	50	
Alamat	*Varchar*	100	
kota	*Varchar*	50	
Telp	*Varchar*	12	
input_petugas	*Varchar*	20	
tanggal_input	*DateTime*		

c. Tabel Customer

Nama tabel : master_customer

Primary Key : Kode_customer

Tabel 5. 1 Tabel Customer

Field	***Type Text***	***Size***	**Keterangan**
Kode_customer	*Varchar*	20	*Primary Key*
Nama_customer	*Varchar*	50	
Alamat	*Varchar*	100	
kota	*Varchar*	50	
Telp	*Varchar*	12	
input_petugas	*Varchar*	20	
tanggal_input	*DateTime*		

d. Tabel Perkiraan

Nama tabel : master_perkiraan

Primary Key : kode_akun

Tabel 5.2 Tabel Perkiraan

Field	***Type Text***	***Size***	**Keterangan**
kode_kategori	*Varchar*	5	
kode_akun	*Varchar*	5	*Primary Key*
nama_akun	*Varchar*	50	
kateg_perkiraan	*Varchar*	50	

e. Tabel Produksi Order

Nama tabel : tr_plan_produksi

Primary Key : kode_produksi

Tabel 5.3 Tabel Produksi Order

Field	*Type Text*	*Size*	**Keterangan**
kode_produksi	*Varchar*	10	*Primary Key*
tgl_produksi	*Date*		
jumlahProduksi	*Double*	20	
satuan	*Varchar*	10	
JenisProduksi	*Varchar*	30	
deskripsi	*Varchar*	150	
hasil_produksi	*Double*	20	
tgl_selesai_produksi	*Date*		
input_petugas	*Varchar*	10	
tgl_input	*DateTime*		

f. Tabel Master Permintaan Bahan

Nama tabel : tr_master_permintaan_bahan

Primary Key : kode_request

Tabel 5.4 Tabel Master Permintaan Bahan

Field	*Type Text*	*Size*	**Keterangan**
kode_request	*Varchar*	10	*Primary Key*
tgl_request	*Date*		
kode_produksi	*Varchar*	10	
keterangan	*Varchar*	150	
input_petugas	*Varchar*	20	
tgl_input	*DateTime*		

g. Tabel Detail Permintaan Bahan

Nama tabel : tr_detail_permintaan_bahan

Primary Key : kode_seq

Seq

Tabel 5.5 Tabel Detail Permintaan Bahan

Field	***Type Text***	***Size***	**Keterangan**
kode_request	*Varchar*	10	
kode_seq	*Integer*	11	*Primary Key*
kode_bahan	*Varchar*	10	
qty	*Double*	20	

h. Tabel Master Pengeluaran Bahan

Nama tabel : tr_master_pengeluaran_bahan

Primary Key : Kode_pengeluaran

Tabel 5.6 Tabel Master Pengeluaran Bahan

Field	***Type Text***	***Size***	**Keterangan**
Kode_pengeluaran	*Varchar*	10	*Primary Key*
tgl_pengeluaran	*Date*		
kode_request	*Varchar*	10	
keterangan	*Varchar*	100	
input_petugas	*Varchar*	20	
tgl_input	*DateTime*		

i. Tabel Detail Pengeluaran Bahan

Nama tabel : tr_detail_pengeluaran_bahan

Primary Key : kode_pengeluaran

Tabel 5.7 Tabel Detail Pengeluaran Bahan

Field	***Type Text***	***Size***	**Keterangan**
kode_pengeluaran	*Varchar*	10	*Primary Key*
kode_seq	*Integer*	11	
kode_bahan	*Varchar*	10	
qtyKeluar	*Double*	20	

j. Tabel Master Barang Masuk

Nama tabel : tr_master_barangmasuk

Primary Key : kode_barang_masuk

Tabel 5.8 Tabel Master Barang Masuk

Field	***Type Text***	***Size***	**Keterangan**
kode_barang_masuk	*Varchar*	10	*Primary Key*
Faktur	*Varchar*	11	
tgl_masuk	*Date*		
kode_supplier	*Varchar*	20	
keterangan	*Varchar*	100	
input_petugas	*Varchar*	20	
tgl_input	*DateTime*		

k. Tabel Detail Barang Masuk

Nama tabel : tr_detail_barangmasuk

Primary Key : kode_bahan

Tabel 5.9 Tabel Detail Barang Masuk

Field	***Type Text***	***Size***	**Keterangan**
kode_barang_masuk	*Varchar*	10	
kode_bahan	*Varchar*	10	*Primary Key*
qty	*Double*	20	
harga	*Double*	20	
JU_no_transaksi	*Varchar*	50	

l. Tabel Hasil Produksi

Nama tabel : tr_barang_jadi

Primary Key : kode_produksi

Tabel 5.10 Tabel Detail Hasil Produksi

Field	***Type Text***	***Size***	**Keterangan**
kode_produksi	*Varchar*	50	*Primary Key*
kode_BJ	*Varchar*	50	
qty_produksi	*Double*	20	
tgl_produksi	*Date*		
input_user	*Varchar*	50	
tgl_input	*DateTime*		

m. Tabel Master Penjualan

Nama tabel : tbl_mst_penjualan

Primary Key : no_penjualan

Tabel 5.11 Tabel Master Penjualan

Field	***Type Text***	***Size***	**Keterangan**
no_penjualan	*Varchar*	10	*Primary Key*
tgl_penjualan	*Date*		
kode_customer	*Varchar*	50	
totalPenjualan	*Double*	20	
totalPembayaran	*Double*	20	
id_petugas	*Varchar*	20	
tgl_input	*DateTime*		

n. Tabel Detail Penjualan

Nama tabel : tbl_dtl_penjualan

Primary Key : no_penjualan

Tabel 5.12 Tabel Detail Penjualan

Field	***Type Text***	***Size***	**Keterangan**
no_penjualan	*Varchar*	10	*Primary Key*
kode_produk	*Varchar*	10	
qty_jual	*Double*	20	
hargasatuan	*Double*	20	
JU_no_transaksi	*Varchar*	50	

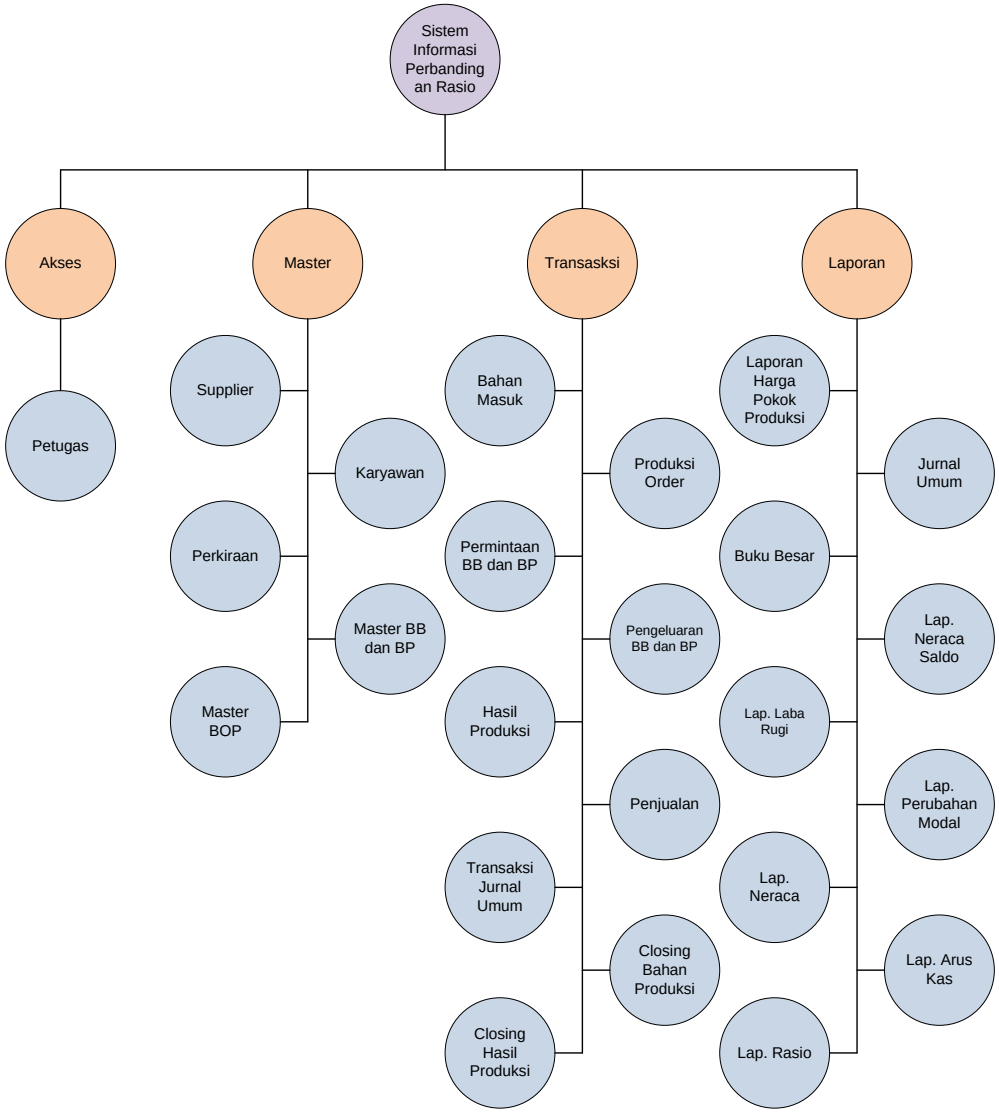
o. Tabel Jurnal Umum

Nama tabel : tr_ju

Primary Key : No_transaksi

Tabel 5.13 Tabel Jurnal Umum

Field	***Type Text***	***Size***	**Keterangan**
No_transaksi	*Varchar*	10	*Primary Key*
tgl_transaksi	*Date*		
kode_akun_debit	*Varchar*	10	
kode_akun_kredit	*Varchar*	10	
keterangan	*Varchar*	100	
saldo	*Double*	20	
input_user	*Varchar*	20	
tgl_input	*DateTime*		

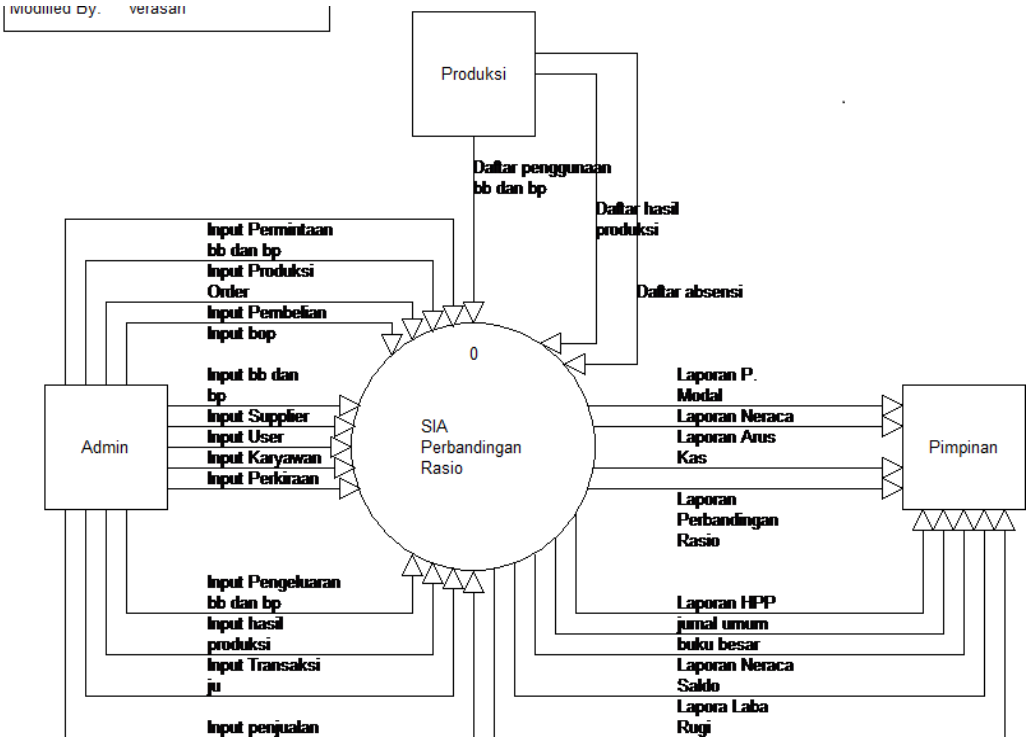


Dekomposisi

Gambar 5.6 Dekomposisi Sistem Informasi Perbandingan Rasio Laporan Keuangan

5.3 Data Flow Diagram (DFD)

a. Context Diagram



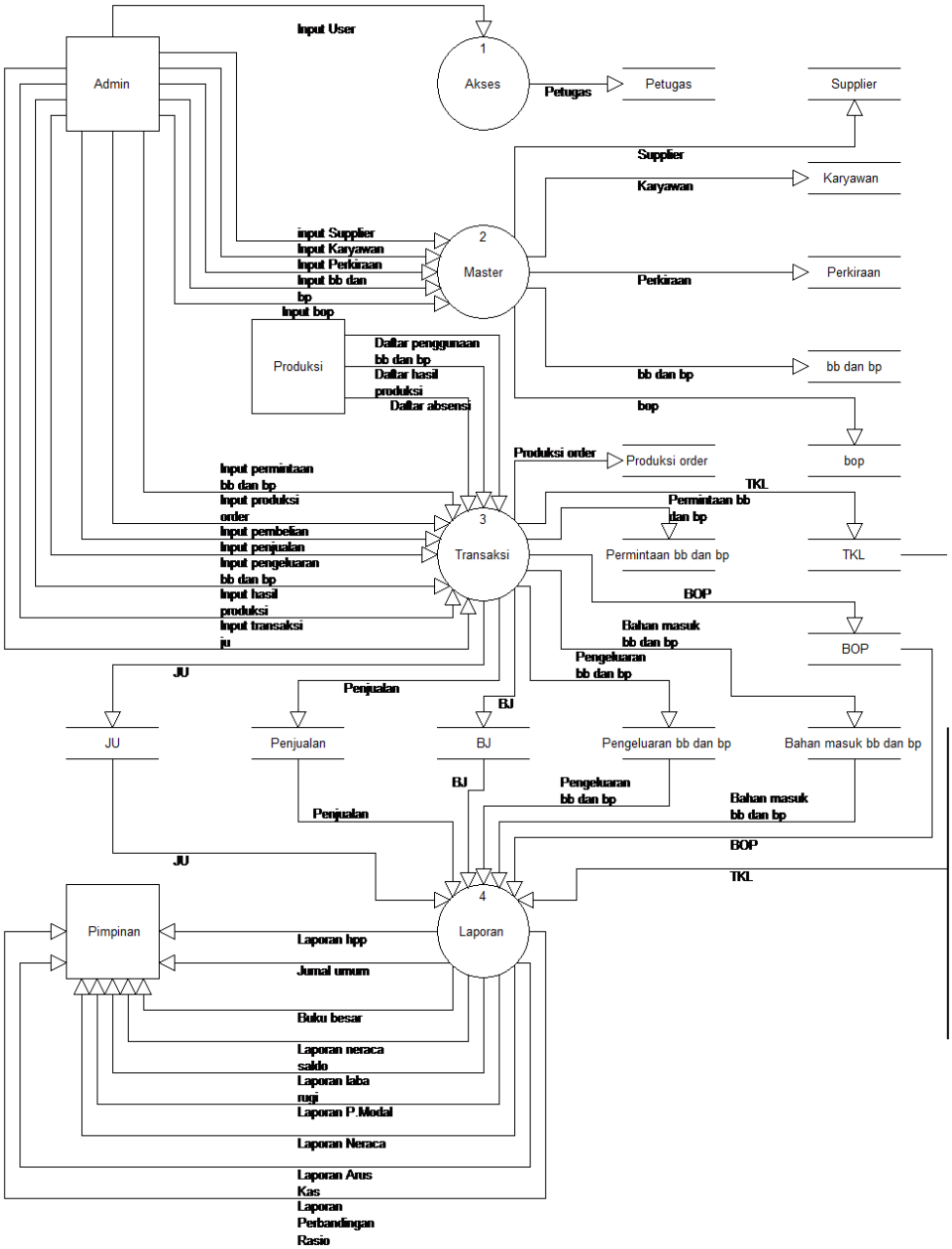
Rule Checking chart "Context Diagram".

I203: The External Entity "Admin" does not explode.
I203: The External Entity "Pimpinan" does not explode.
I203: The External Entity "Produksi" does not explode.
I203: The Data Flow "Input User" does not explode.
I203: The Data Flow "Laporan Neraca" does not explode.
I203: The Data Flow "Daftar penggunaan bb dan bp" does not explode.
I203: The Data Flow "Input Supplier" does not explode.
I203: The Data Flow "Input Karyawan" does not explode.
I203: The Data Flow "Input Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "Input hasil produksi" does not explode.
I203: The Data Flow "Input Transaksi ju" does not explode.
I203: The Data Flow "Input bop" does not explode.
I203: The Data Flow "Input Pembelian" does not explode.
I203: The Data Flow "Input Produksi Order" does not explode.
I203: The Data Flow "Input Permintaan bb dan bp" does not explode.
I203: The Data Flow "Input Perkiraan" does not explode.
I203: The Data Flow "Input penjualan" does not explode.
I203: The Data Flow "Input bb dan bp" does not explode.
I203: The Data Flow "Daftar hasil produksi" does not explode.
I203: The Data Flow "Daftar absensi" does not explode.
I203: The Data Flow "Laporan P. Modal" does not explode.
I203: The Data Flow "Laporan Arus Kas" does not explode.
I203: The Data Flow "Lapora Laba Rugi" does not explode.
I203: The Data Flow "Laporan Neraca Saldo" does not explode.
I203: The Data Flow "buku besar" does not explode.
I203: The Data Flow "jurnal umum" does not explode.
I203: The Data Flow "Laporan HPP" does not explode.
I203: The Data Flow "Laporan Perbandingan Rasio" does not explode.

No errors found.

Gambar 5.7 Context Diagram

b. DFD Level 0

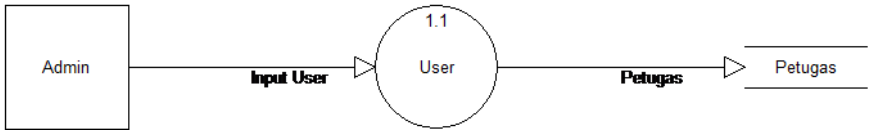

Input User
1
Akses
Admin
Petugas
Petugas
Supplier
Supplier
Karyawan
Karyawan
2
Master
input Supplier
Input Karyawan
Input Perkiraan
Input bb dan bp
Input bop
Perkiraan
Perkiraan
Produksi
Daftar penggunaan bb dan bp
Daftar hasil produksi
Daftar absensi
bb dan bp
bb dan bp
bop
bop
Produksi order
Produksi order
Input permintaan bb dan bp
Input produksi order
Input pembelian
Input penjualan
Input pengeluaran bb dan bp
Input hasil produksi
Input transaksi ju
3
Transaksi
TKL
Permintaan bb dan bp
Permintaan bb dan bp
TKL
BOP
BOP
Bahan masuk bb dan bp
Pengeluaran bb dan bp
JU
Penjualan
BJ
JU
Penjualan
BJ
Pengeluaran bb dan bp
Bahan masuk bb dan bp
BJ
Pengeluaran bb dan bp
Penjualan
Bahan masuk bb dan bp
BOP
JU
TKL
4
Laporan
Pimpinan
Laporan hpp
Jurnal umum
Buku besar
Laporan neraca saldo
Laporan laba rugi
Laporan P.Modal
Laporan Neraca
Laporan Arus Kas
Laporan Perbandingan Rasio

Rule Checking chart "SIA Perbandingan Rasio".

I203: The External Entity "Admin" does not explode.
I203: The Data Store "Petugas" does not explode.
I203: The Data Store "Supplier" does not explode.
I203: The Data Store "Karyawan" does not explode.
I203: The Data Store "bb dan bp" does not explode.
I203: The Data Store "Perkiraan" does not explode.
I203: The Data Store "bop" does not explode.
I203: The External Entity "Produksi" does not explode.
I203: The Data Store "Permintaan bb dan bp" does not explode.
I203: The Data Store "Pengeluaran bb dan bp" does not explode.
I203: The Data Store "Produksi order" does not explode.
I203: The Data Store "Bahan masuk bb dan bp" does not explode.
I203: The Data Store "BJ" does not explode.
I203: The Data Store "Penjualan" does not explode.
I203: The Data Store "JU" does not explode.
I203: The Data Store "BOP" does not explode.
I203: The Data Store "TKL" does not explode.
I203: The External Entity "Pimpinan" does not explode.
I203: The Data Flow "Input hasil produksi" does not explode.
I203: The Data Flow "Input pengeluaran bb dan bp" does not explode.
I203: The Data Flow "bop" does not explode.
I203: The Data Flow "Laporan hpp" does not explode.
I203: The Data Flow "Daftar penggunaan bb dan bp" does not explode.
I203: The Data Flow "Buku besar" does not explode.
I203: The Data Flow "Laporan Arus Kas" does not explode.
I203: The Data Flow "Laporan Perbandingan Rasio" does not explode.
I203: The Data Flow "bb dan bp" does not explode.
I203: The Data Flow "BJ" does not explode.
I203: The Data Flow "Laporan laba rugi" does not explode.
I203: The Data Flow "Laporan neraca saldo" does not explode.
I203: The Data Flow "Bahan masuk bb dan bp" does not explode.
I203: The Data Flow "BOP" does not explode.
I203: The Data Flow "TKL" does not explode.
I203: The Data Flow "JU" does not explode.
I203: The Data Flow "Penjualan" does not explode.
I203: The Data Flow "Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "Daftar absensi" does not explode.
I203: The Data Flow "Daftar hasil produksi" does not explode.
I203: The Data Flow "Input transaksi ju" does not explode.
I203: The Data Flow "Input bop" does not explode.
I203: The Data Flow "Jurnal umum" does not explode.
I203: The Data Flow "Laporan Neraca" does not explode.
I203: The Data Flow "Laporan P.Modal" does not explode.
I203: The Data Flow "Input User" does not explode.
I203: The Data Flow "Petugas" does not explode.
I203: The Data Flow "input Supplier" does not explode.
I203: The Data Flow "Input Karyawan" does not explode.
I203: The Data Flow "Input Perkiraan" does not explode.
I203: The Data Flow "Input bb dan bp" does not explode.
I203: The Data Flow "Supplier" does not explode.
I203: The Data Flow "Karyawan" does not explode.
I203: The Data Flow "Perkiraan" does not explode.
I203: The Data Flow "Input permintaan bb dan bp" does not explode.
I203: The Data Flow "Input produksi order" does not explode.
I203: The Data Flow "Input pembelian" does not explode.
I203: The Data Flow "Input penjualan" does not explode.
I203: The Data Flow "Produksi order" does not explode.
I203: The Data Flow "Permintaan bb dan bp" does not explode.
I203: The Data Flow "Bahan masuk bb dan bp" does not explode.
I203: The Data Flow "Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "JU" does not explode.
I203: The Data Flow "Penjualan" does not explode.
I203: The Data Flow "BJ" does not explode.
I203: The Data Flow "TKL" does not explode.
I203: The Data Flow "BOP" does not explode.

No errors found.

c. DFD Level 1 Akses



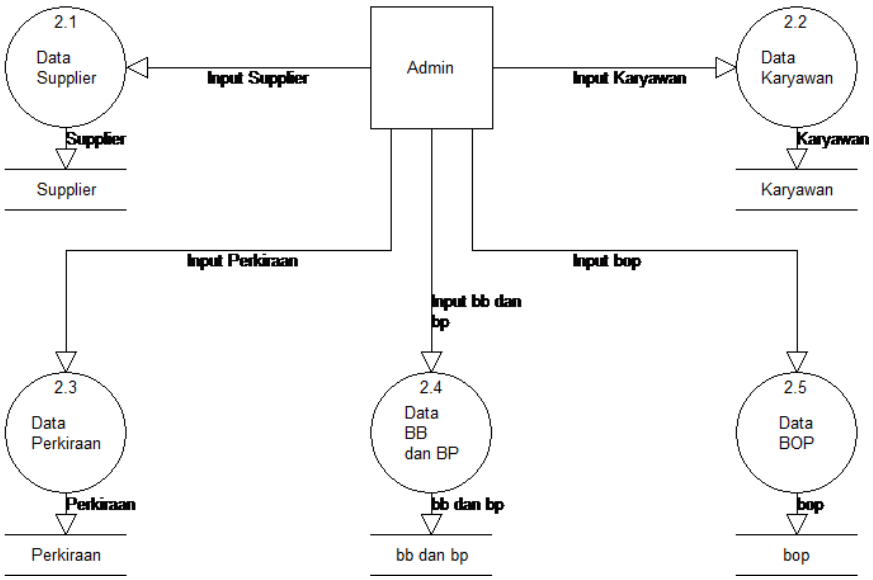
Rule Checking chart "Akses".

I203: The External Entity "Admin" does not explode.
I203: The Data Process "User" does not explode.
I203: The Data Store "Petugas" does not explode.
I203: The Data Flow "Input User" does not explode.
I203: The Data Flow "Petugas" does not explode.

No errors found.

Gambar 5.8 DFD Level 1 Akses

d. DFD Level 1 Master



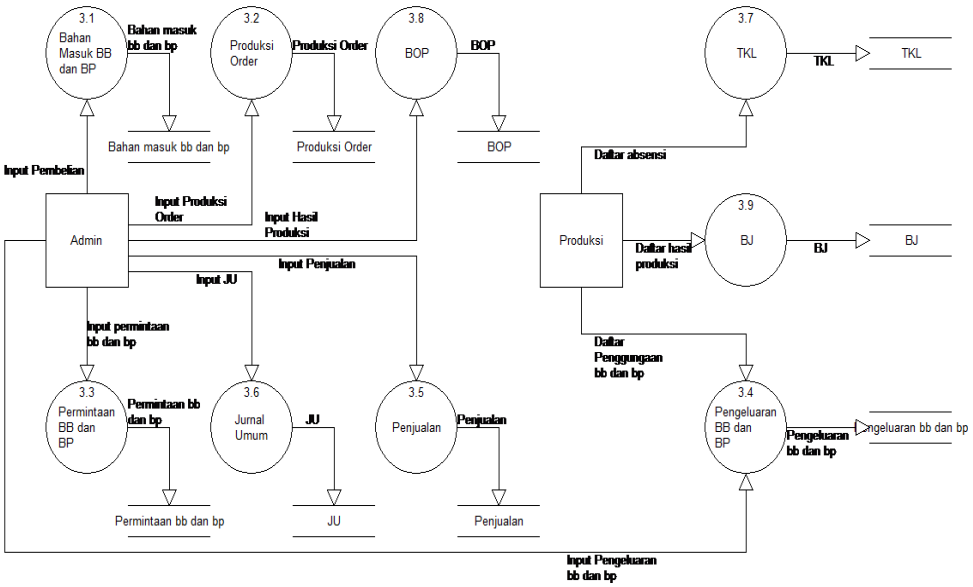
Rule Checking chart "Master".

I203: The External Entity "Admin" does not explode.
I203: The Data Process "Data BOP" does not explode.
I203: The Data Process "Data BB dan BP" does not explode.
I203: The Data Process "Data Supplier" does not explode.
I203: The Data Process "Data Perkiraan" does not explode.
I203: The Data Process "Data Karyawan" does not explode.
I203: The Data Store "Karyawan" does not explode.
I203: The Data Store "Supplier" does not explode.
I203: The Data Store "Perkiraan" does not explode.
I203: The Data Store "bb dan bp" does not explode.
I203: The Data Store "bop" does not explode.
I203: The Data Flow "bop" does not explode.
I203: The Data Flow "Input Karyawan" does not explode.
I203: The Data Flow "Input bop" does not explode.
I203: The Data Flow "Input bb dan bp" does not explode.
I203: The Data Flow "Input Perkiraan" does not explode.
I203: The Data Flow "bb dan bp" does not explode.
I203: The Data Flow "Perkiraan" does not explode.
I203: The Data Flow "Karyawan" does not explode.
I203: The Data Flow "Supplier" does not explode.
I203: The Data Flow "Input Supplier" does not explode.

No errors found.

Gambar 5.9 DFD Level 1 Master

e. DFD Level 1 Transaksi



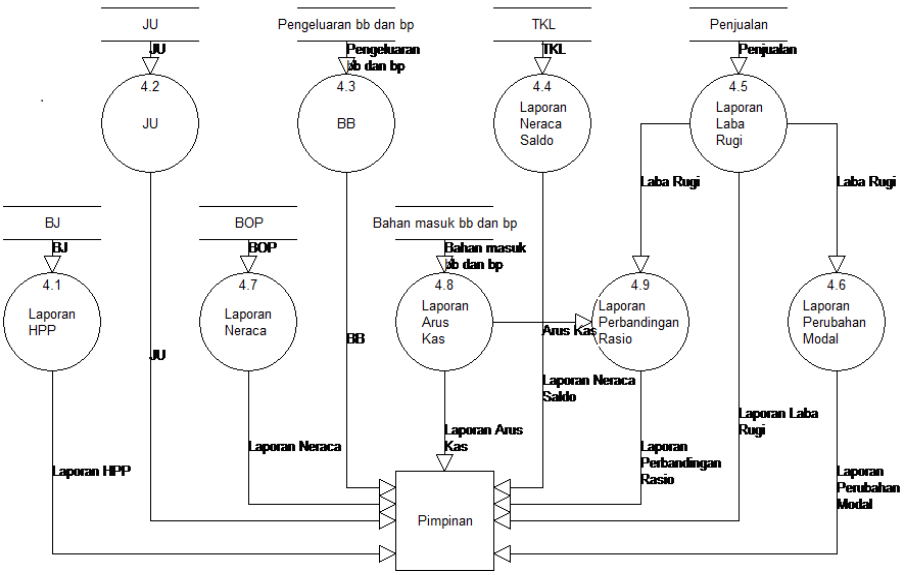
Rule Checking chart "Transaksi".

I203: The Data Process "TKL" does not explode.
I203: The Data Process "BOP" does not explode.
I203: The Data Process "Jurnal Umum" does not explode.
I203: The Data Process "Bahan Masuk BB dan BP" does not explode.
I203: The Data Process "Permintaan BB dan BP" does not explode.
I203: The Data Process "Pengeluaran BB dan BP" does not explode.
I203: The Data Process "BJ" does not explode.
I203: The Data Process "Penjualan" does not explode.
I203: The Data Process "Produksi Order" does not explode.
I203: The External Entity "Produksi" does not explode.
I203: The External Entity "Admin" does not explode.
I203: The Data Store "TKL" does not explode.
I203: The Data Store "BJ" does not explode.
I203: The Data Store "Pengeluaran bb dan bp" does not explode.
I203: The Data Store "Bahan masuk bb dan bp" does not explode.
I203: The Data Store "Produksi Order" does not explode.
I203: The Data Store "BOP" does not explode.
I203: The Data Store "Permintaan bb dan bp" does not explode.
I203: The Data Store "JU" does not explode.
I203: The Data Store "Penjualan" does not explode.
I203: The Data Flow "Input Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "Permintaan bb dan bp" does not explode.
I203: The Data Flow "JU" does not explode.
I203: The Data Flow "Penjualan" does not explode.
I203: The Data Flow "Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "BJ" does not explode.
I203: The Data Flow "TKL" does not explode.
I203: The Data Flow "Daftar Penggungaan bb dan bp" does not explode.
I203: The Data Flow "Daftar hasil produksi" does not explode.
I203: The Data Flow "Daftar absensi" does not explode.
I203: The Data Flow "BOP" does not explode.
I203: The Data Flow "Produksi Order" does not explode.
I203: The Data Flow "Bahan masuk bb dan bp" does not explode.
I203: The Data Flow "Input Hasil Produksi" does not explode.
I203: The Data Flow "Input JU" does not explode.
I203: The Data Flow "Input Penjualan" does not explode.
I203: The Data Flow "Input Produksi Order" does not explode.
I203: The Data Flow "Input permintaan bb dan bp" does not explode.
I203: The Data Flow "Input Pembelian" does not explode.

No errors found.

Gambar 5.10 DFD Level 1 Transaksi

f. DFD Level 4 Laporan



Rule Checking chart "Laporan".

I203: The Data Store "BJ" does not explode.
I203: The Data Store "JU" does not explode.
I203: The Data Store "Pengeluaran bb dan bp" does not explode.
I203: The Data Store "TKL" does not explode.
I203: The Data Store "Penjualan" does not explode.
I203: The Data Store "BOP" does not explode.
I203: The External Entity "Pimpinan" does not explode.
I203: The Data Store "Bahan masuk bb dan bp" does not explode.
I203: The Data Process "Laporan Perbandingan Rasio" does not explode.
I203: The Data Process "Laporan Perubahan Modal" does not explode.
I203: The Data Process "Laporan Neraca" does not explode.
I203: The Data Process "Laporan HPP" does not explode.
I203: The Data Process "JU" does not explode.
I203: The Data Process "BB" does not explode.
I203: The Data Process "Laporan Neraca Saldo" does not explode.
I203: The Data Process "Laporan Laba Rugi" does not explode.
I203: The Data Process "Laporan Arus Kas" does not explode.
I203: The Data Flow "Laporan Perubahan Modal" does not explode.
I203: The Data Flow "BB" does not explode.
I203: The Data Flow "Laporan HPP" does not explode.
I203: The Data Flow "BOP" does not explode.
I203: The Data Flow "Bahan masuk bb dan bp" does not explode.
I203: The Data Flow "Laporan Laba Rugi" does not explode.
I203: The Data Flow "Laporan Neraca Saldo" does not explode.
I203: The Data Flow "Laporan Neraca" does not explode.
I203: The Data Flow "JU" does not explode.
I203: The Data Flow "Laporan Arus Kas" does not explode.
I203: The Data Flow "Laporan Perbandingan Rasio" does not explode.
I203: The Data Flow "Penjualan" does not explode.
I203: The Data Flow "TKL" does not explode.
I203: The Data Flow "Pengeluaran bb dan bp" does not explode.
I203: The Data Flow "JU" does not explode.
I203: The Data Flow "BJ" does not explode.
I203: The Data Flow "Laba Rugi" does not explode.
I203: The Data Flow "Arus Kas" does not explode.
I203: The Data Flow "Laba Rugi" does not explode.

No errors found.

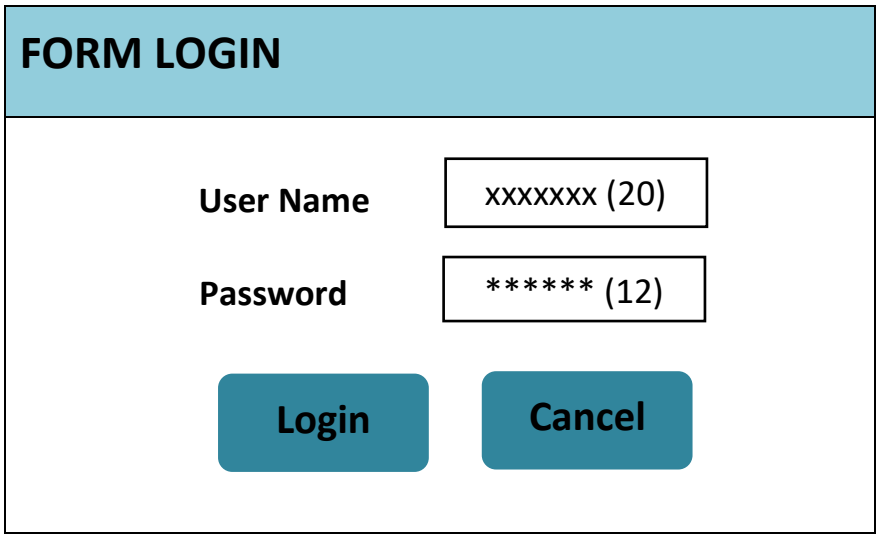
Gambar 5.11 DFD Level 1 Laporan

Desain Halaman Program

Berikut ini desain halaman program yang sedang dirancang oleh penulis:

a. Halaman *Intro*

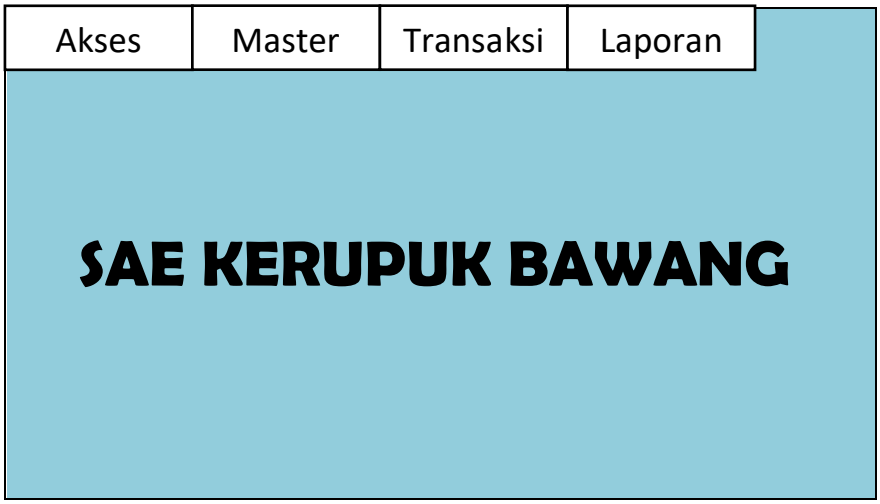
Halaman ini merupakan form *Login* yang akan digunakan *user* untuk mengakses form-form lainnya.



Gambar 5.12 Halaman Form Login

b. Halaman Menu

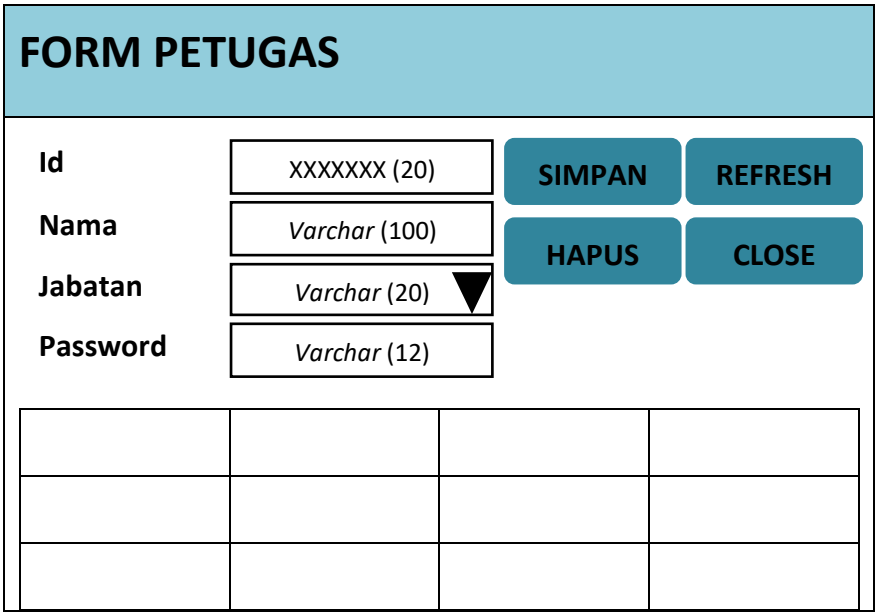
Halaman ini merupakan form menu dari rancangan program yang akan dibuat.



Gambar 5.13 Halaman Form Menu

c. Halaman Form Petugas

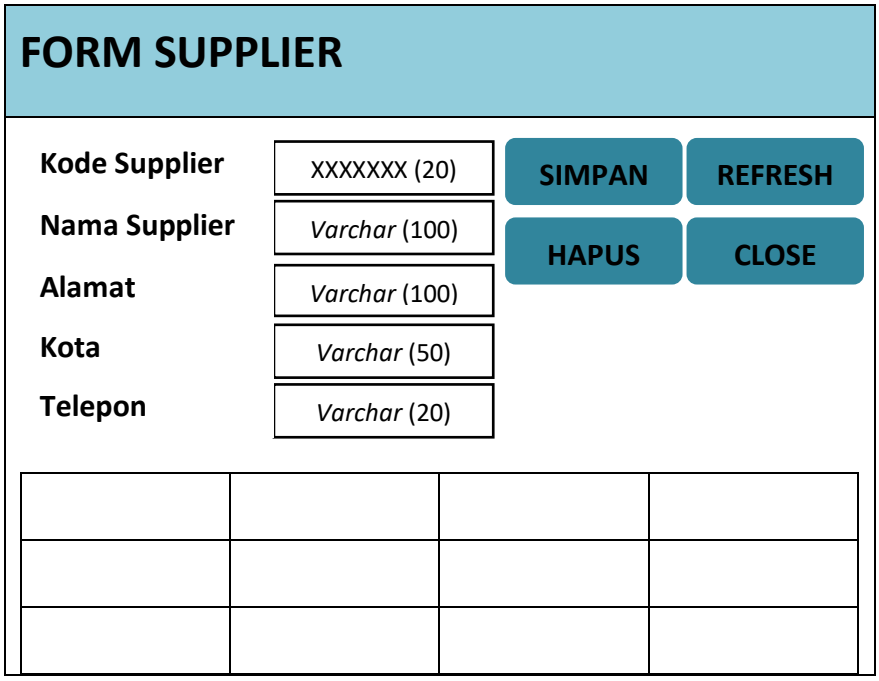
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menambah *user*/petugas



Gambar 5.14 Halaman Form Petugas

d. Halaman Form *Supplier*

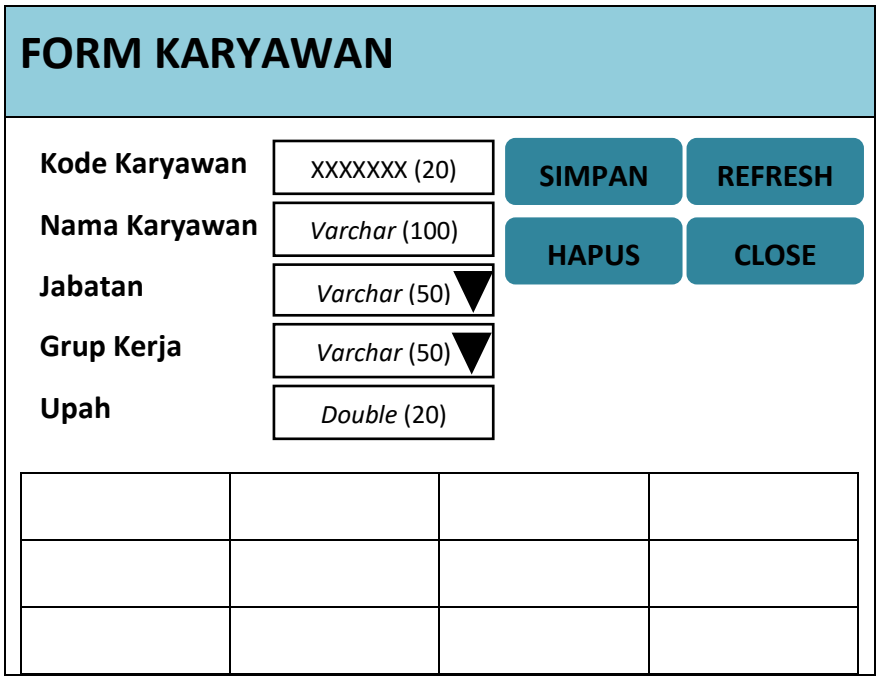
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput data *supplier*



Gambar 5.15 Halaman Form Supplier

e. Halaman Form Karyawan

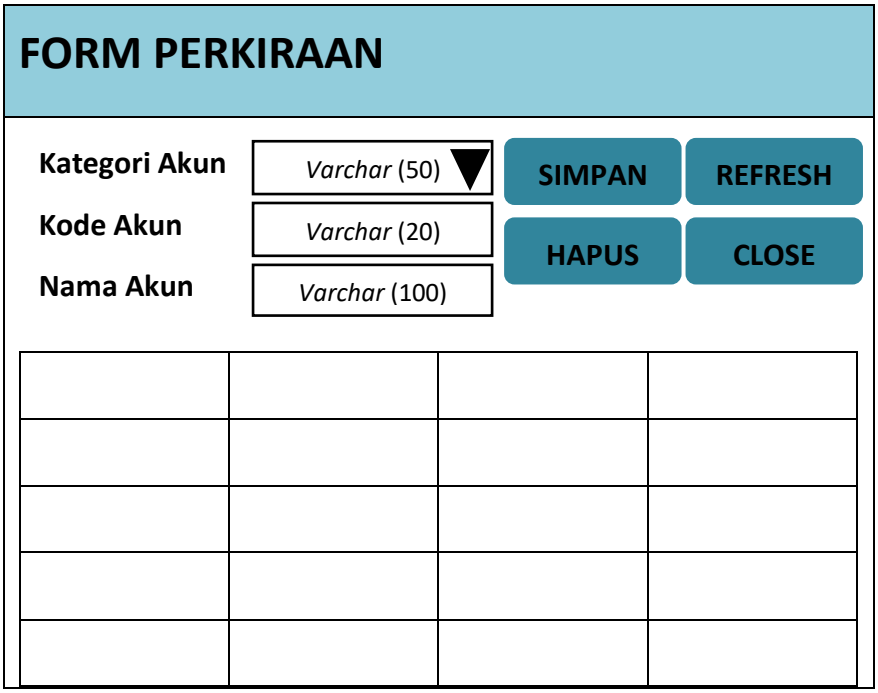
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput data karyawan



Gambar 5.16 Halaman Form Karyawan

f. Halaman Form Perkiraan Akun

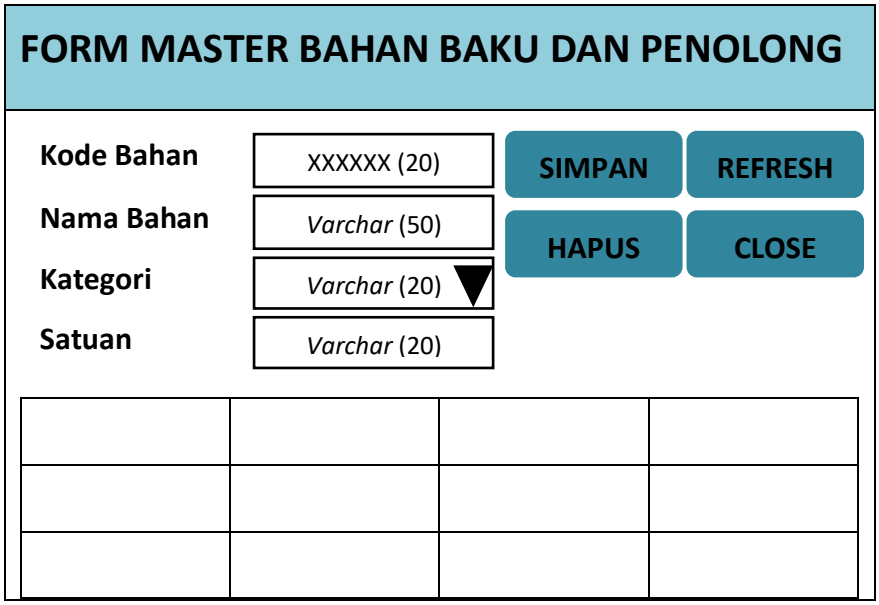
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput data perkiraan akun



Gambar 5.17 Halaman Form Perkiraan Akun

g. Halaman Form Master Bahan Baku dan Penolong

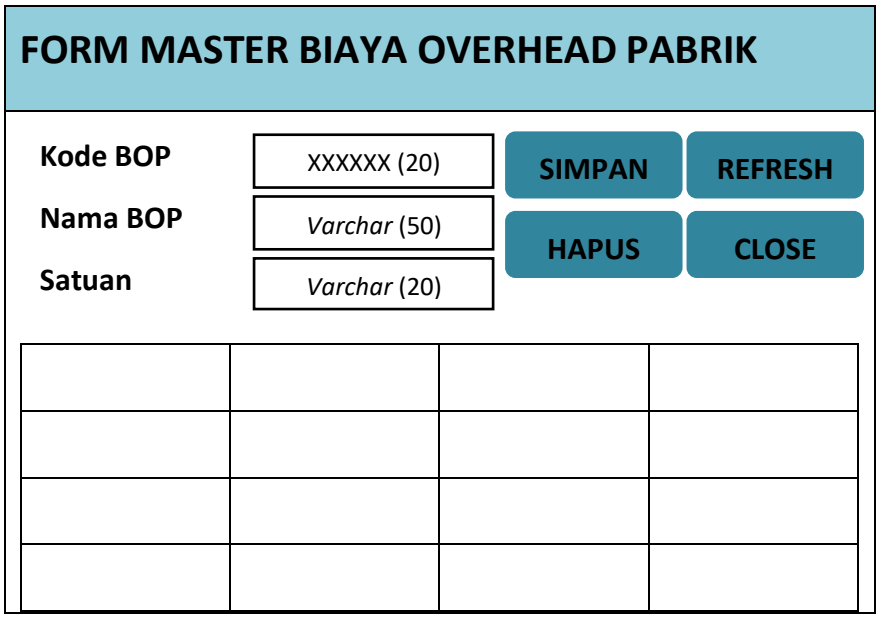
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput data bahan baku dan bahan penolong



Gambar 5.18 Halaman Form Master Bahan Baku dan Penolong

h. Halaman Form Master Biaya Overhead Pabrik

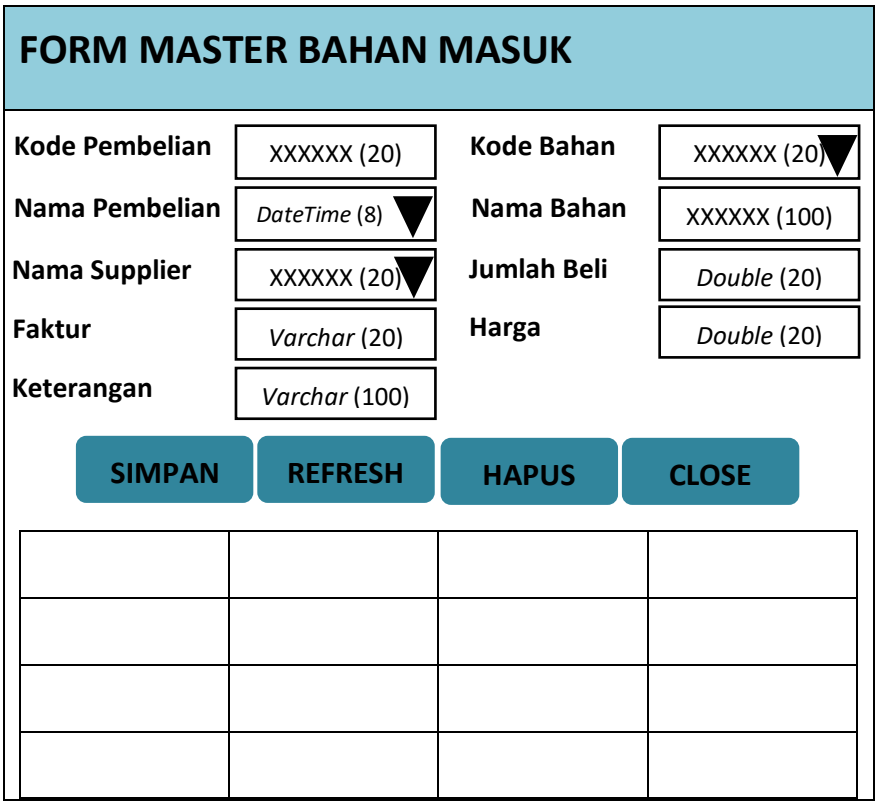
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput data overhead pabrik



Gambar 5.19 Halaman Form Master Biaya Overhead Pabrik

i. Halaman Form Barang Masuk

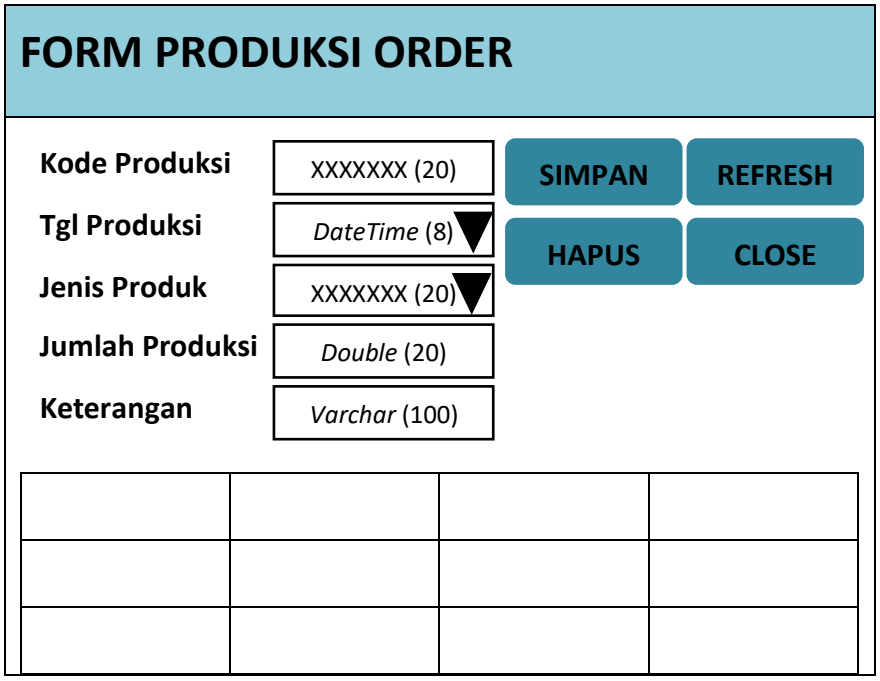
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput transaksi Pembelian bahan



Gambar 5.20 Halaman Form Bahan Masuk

j. Halaman Form Produksi Order

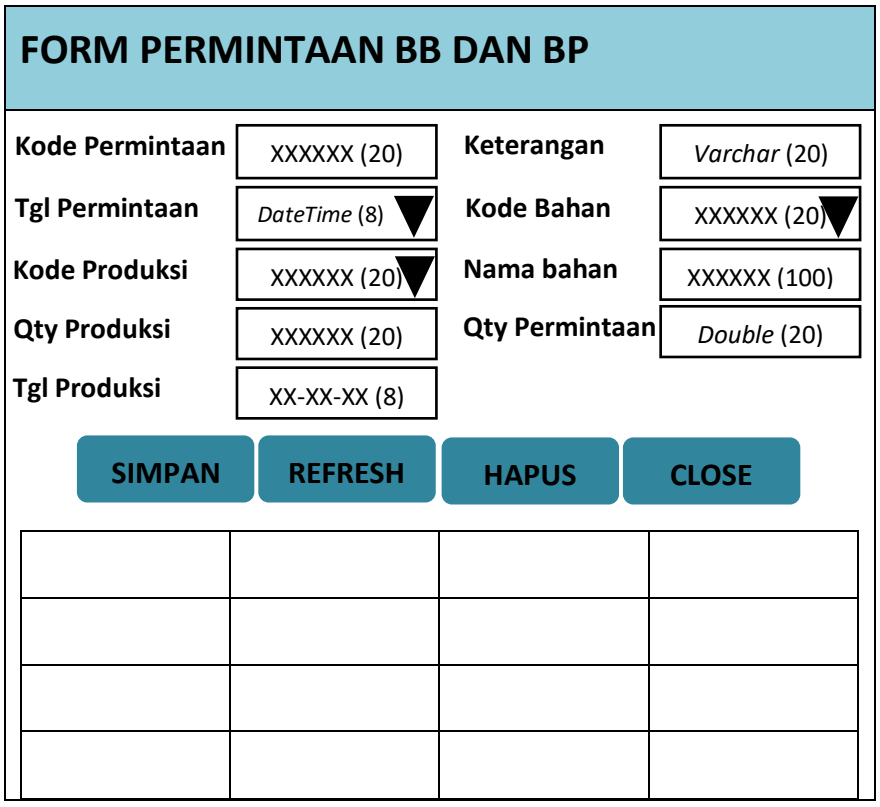
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput rencana produksi selama satu bulan



Gambar 5.21 Halaman Form Produksi Order

k. Halaman Form Permintaan Bahan Baku dan Penolong

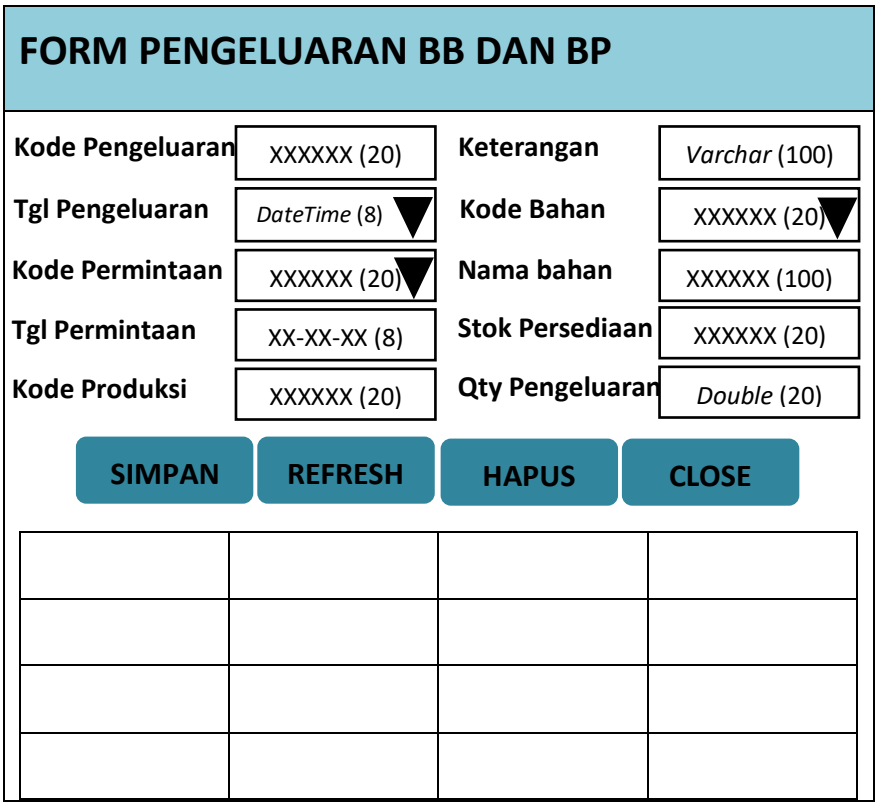
Halaman ini merupakan rancangan halaman yang akan digunakan untuk memasukkan bahan baku dan bahan penolong yang dibutuhkan



Gambar 5.22 Halaman Form Permintaan BB dan BP

l. Halaman Form Pengeluaran Bahan Baku dan Penolong

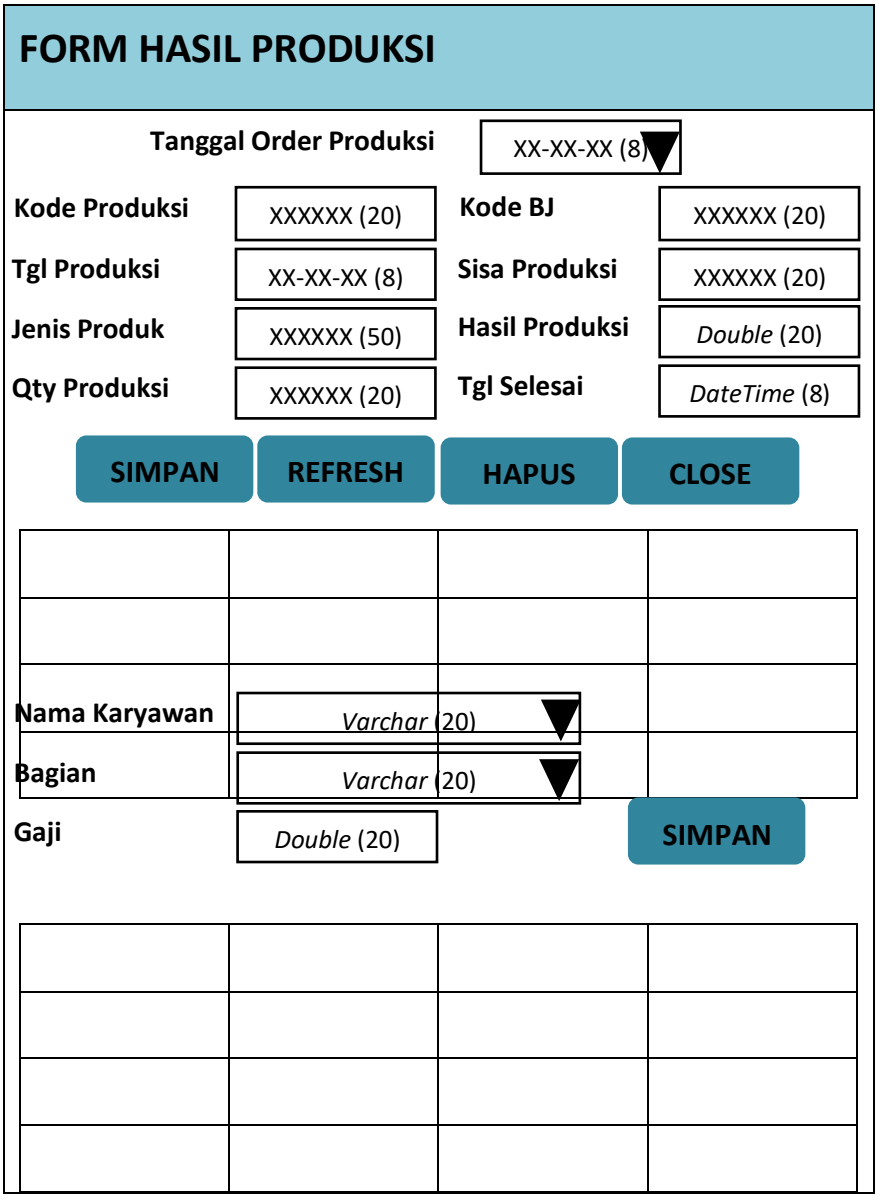
Halaman ini merupakan rancangan halaman yang akan digunakan untuk mencatat pengeluaran bahan baku dan bahan penolong yang diminta



Gambar 5.23 Halaman Form Pengeluaran BB dan BP

m. Halaman Form Hasil Produksi

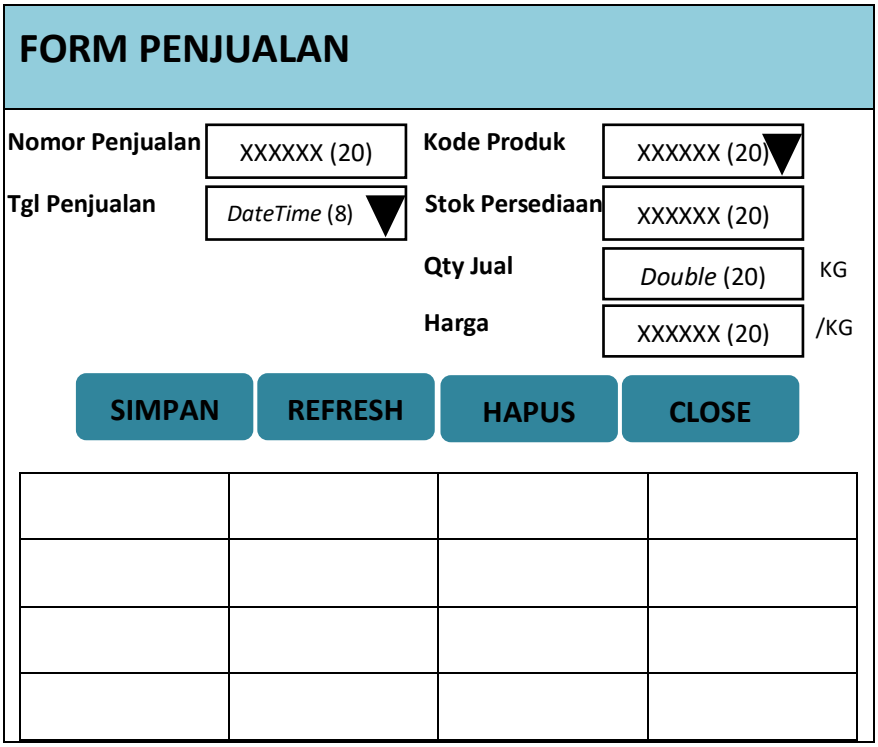
Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput dan menghitung hasil produksi



Gambar 5.24 Halaman Form Hasil Produksi

n. Halaman Form Penjualan

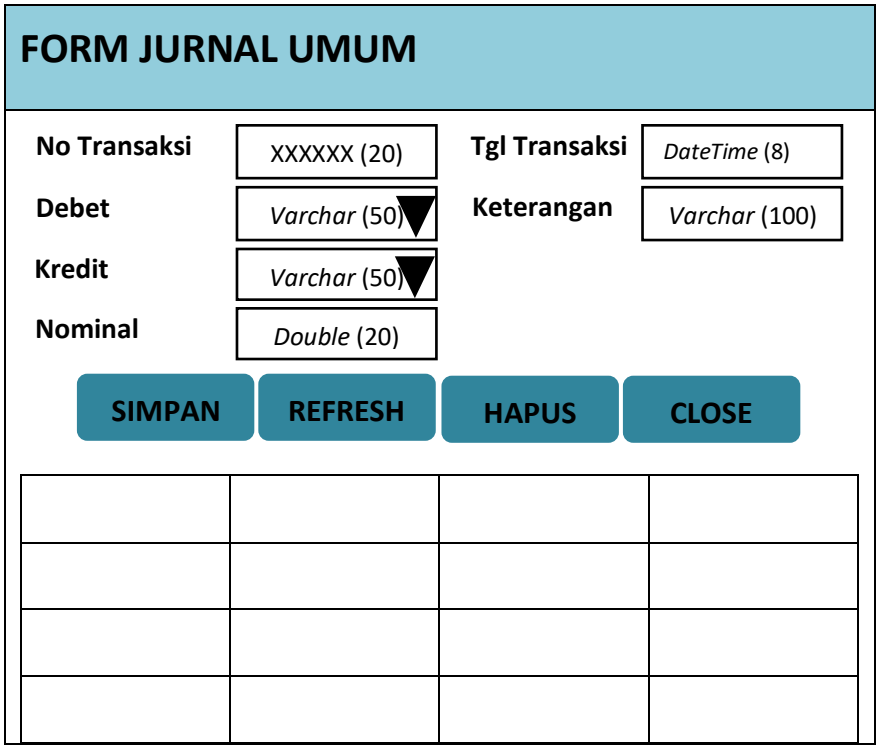
Halaman ini merupakan rancangan halaman yang akan digunakan untuk mencatat transaksi penjualan



Gambar 5.25 Halaman Form Penjualan

o. Halaman Form Jurnal Umum

Halaman ini merupakan rancangan halaman yang akan digunakan untuk menginput bukti transaksi



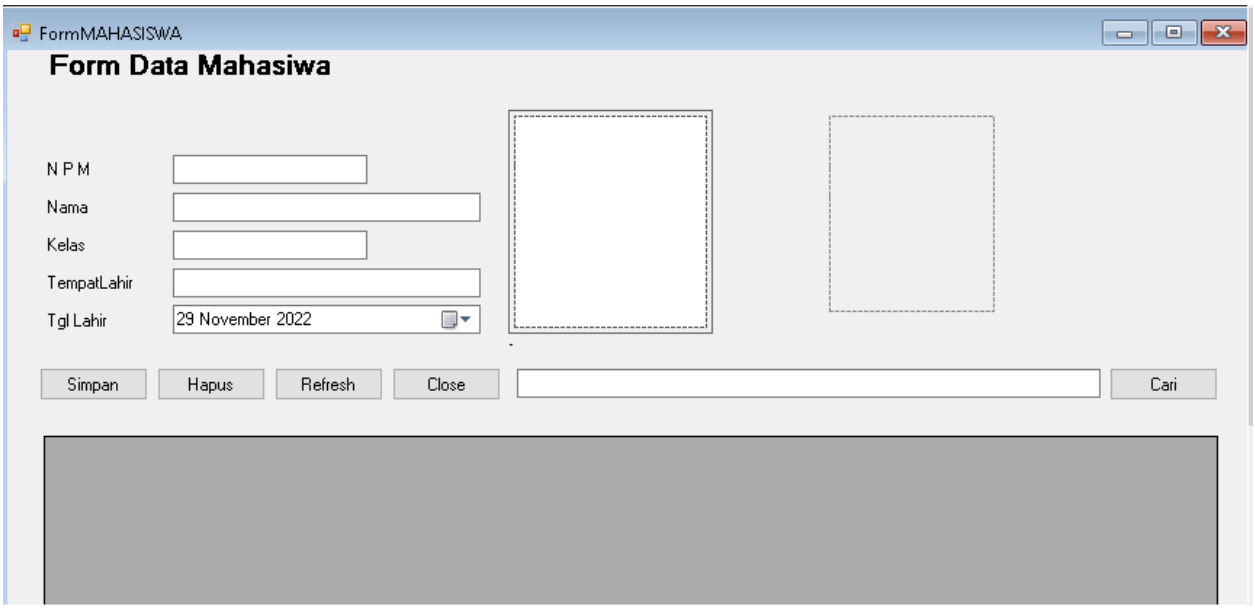
Gambar 5.26 Halaman Form Jurnal Umum

Bab 6

Implementasi Pemrograman-2

6.1 FORM DATA MAHASISWA

1. Aktifkan Program Visual Basic .Net
2. Klik menu File ⇒ New Project
3. Ketik PRNILAI pada kotak Name
4. Klik tombol OK
5. Klik propertie File Name
6. Ketik MAHASISWA.VB
7. Buat desain form di bawah ini



8. Klik menu File ⇒ Save All
9. Klik tombol Browse
10. Tentukan folder lokasi penyimpanan file proyek
11. Klik tombol Select Folder
12. Klik tombol Save
13. Buat database MS Access DBNILAI pada folder PRNILAI\PRNILAI\BIN\DEBUG
14. Buat struktur tabel berikut

Field Name	Data Type	
NPM	Text	Field Size : 15
NAMA	Text	Field Size : 35
KELAS	Text	Field Size : 10
TMPLAHIR	Text	Field Size : 25
TGLLAHIR	Date/Time	
FOTO	Text	Field Size : 100

15. Simpan table dengan nama MAHASISWA
16. Tutup Ms Access
17. Ketik kode program di bawah ini

```
Imports System.Data
Imports System.Data.OleDb
Public Class FORMMAHASISWA
    Public SQLSTR As String
    Public KONEKSI As New OleDb.OleDbConnection
    Public CMD As New OleDb.OleDbCommand
    Public DAMHS As New OleDb.OleDbDataAdapter
    Public DTMHS As New DataTable
    Public KONEKSISTRING As String
    Public LOKASI As String
    Public XFOTO As String
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        LOKASI = System.Environment.CurrentDirectory
        KONEKSISTRING = "PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE =" & LOKASI & "\DBNILAI.MDB"
        KONEKSI = New OleDb.OleDbConnection(KONEKSISTRING)
        KONEKSI.Open()
    End Sub
    Sub KOSONG()
        TXTNPM.Text = ""
        TXTNAMA.Text = ""
        TXTKELAS.Text = ""
        TXTTMPLAHIR.Text = ""
        TXTTGLLAHIR.Text = Now.Date
        PBFOTO.Image = Nothing
        TXTNPM.Focus()
    End Sub
    Sub DGV()
        SQLSTR = "SELECT * FROM MAHASISWA"
        DAMHS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
        DTMHS.Clear()
        DAMHS.Fill(DTMHS)
        DGVMHS.DataSource = DTMHS
    End Sub
    Private Sub Form1_Activated(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Activated
        DGV()
    End Sub
```

```
Private Sub TXTNPM_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TXTNPM.Leave
    On Error Resume Next
    SQLSTR = "SELECT * FROM MAHASISWA WHERE NPM='" & TXTNPM.Text & "'"
    DAMHS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
    DTMHS.Clear()
    DAMHS.Fill(DTMHS)
    DGVMHS.DataSource = DTMHS
    If DTMHS.Rows.Count > 0 Then
        TXTNAMA.Text = DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(1).Value
        TXTKELAS.Text = DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(2).Value
        TXTTMPLAHIR.Text = DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(3).Value
        TXTTGLLAHIR.Text = DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(4).Value
        PBFOTO.Image = Image.FromFile(DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(5).Value)
        XFOTO = DGVMHS.Rows.Item(DGVMHS.CurrentRow.Index).Cells(5).Value
    End If
    DGV()
End Sub
Private Sub PBFOTO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PBFOTO.Click
    OFD.ShowDialog()
    PBFOTO.Image = Image.FromFile(OFD.FileName)
    XFOTO = OFD.FileName
End Sub
Sub SIMPAN()
    SQLSTR = "SELECT * FROM MAHASISWA WHERE NPM='" & TXTNPM.Text & "'"
    DAMHS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
    DTMHS.Clear()
    DAMHS.Fill(DTMHS)
    DGVMHS.DataSource = DTMHS
    If DTMHS.Rows.Count > 0 Then
        SQLSTR = "UPDATE MAHASISWA SET NPM='" & TXTNPM.Text & "',NAMA='" & TXTNAMA.Text & "',KELAS='" & _
        TXTKELAS.Text & "',TMPLAHIR='" & TXTTMPLAHIR.Text & "',TGLLAHIR='" & _
        TXTTGLLAHIR.Value.ToString("yyyy.MM.dd") & "',FOTO='" & XFOTO & "' WHERE NPM='" & TXTNPM.Text & "'"
        CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
        CMD.ExecuteNonQuery()
    Else
        SQLSTR = "INSERT INTO MAHASISWA VALUES ('" & TXTNPM.Text & "','" & TXTNAMA.Text & "','" & TXTKELAS.Text & _
        "','" & TXTTMPLAHIR.Text & "','" & TXTTGLLAHIR.Value.ToString("yyyy.MM.dd") & "','" & XFOTO & "')"
        CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
        CMD.ExecuteNonQuery()
    End If
    KOSONG()
    DGV()
End Sub
```

```
    Sub HAPUS()
        Dim X As String
        X = MsgBox("DATA MAHASISWA JADI DIHAPUS !", MsgBoxStyle.YesNo)
        If X = vbYes Then
            SQLSTR = "DELETE FROM MAHASISWA WHERE NPM='" & TXTNPM.Text & "'"
            CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
            CMD.ExecuteNonQuery()
        End If
        KOSONG()
        DGV()
    End Sub
    Private Sub BTSIMPAN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTSIMPAN.Click
        SIMPAN()
        KOSONG()
        DGV()
    End Sub
    Private Sub BTHAPUS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTHAPUS.Click
        HAPUS()
    End Sub
    Private Sub BTBLANKFORM_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTBLANKFORM.Click
        KOSONG()
    End Sub
    Private Sub BTCLOSE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCLOSE.Click
        Close()
    End Sub
    Private Sub Form1_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Leave
        KONEKSI.Close()
    End Sub
End Class
```

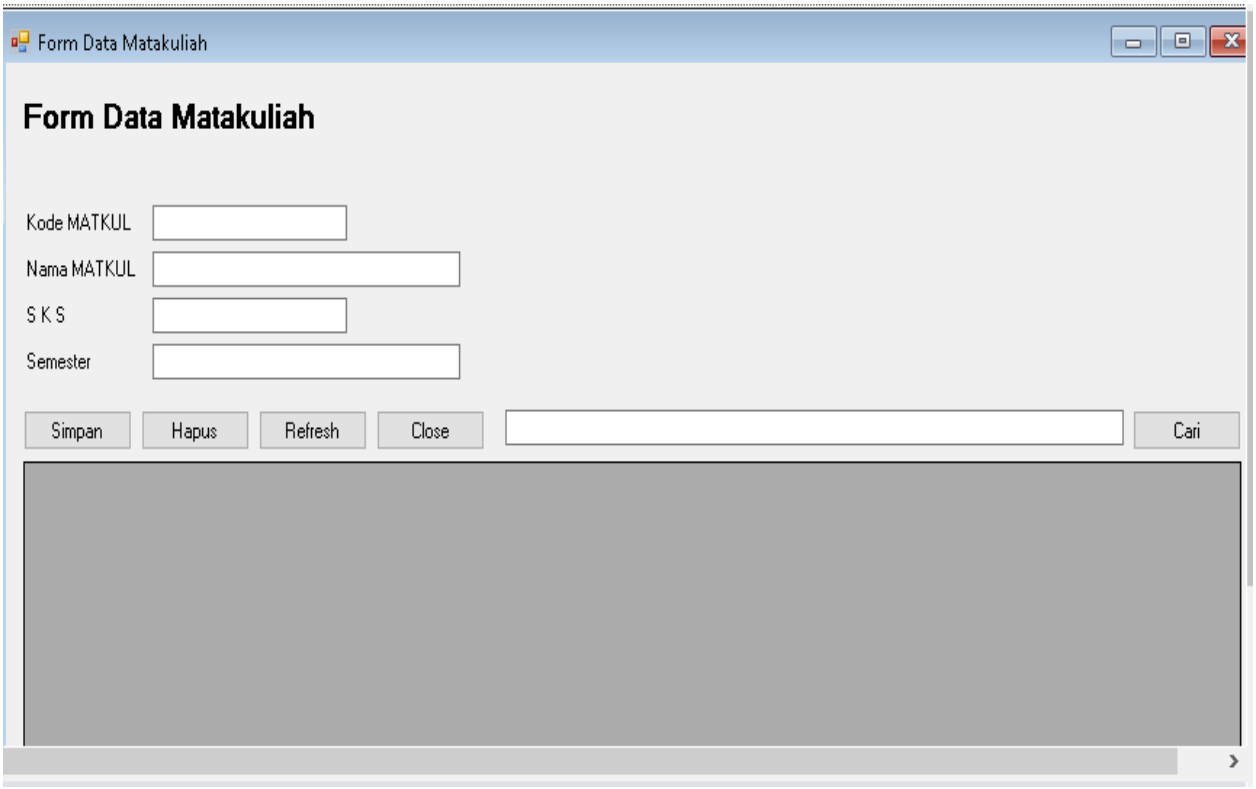
18. Isikan Data Mahasiswa

6.2 FORM DATA MATA KULIAH

1. Buka database DBNILAI.MDB
2. Buat struktur tabel berikut :

Field Name	Data Type	
KODEMK	Text	FieldSize : 6
NAMAMK	Text	FieldSize : 50
SKS	Number	
SEMESTER	Text	FieldSize : 10

3. Simpan tabel dengan nama MATAKULIAH
4. Tutup program MS Access
5. Double klik file PROJECTNILAI.SLN untuk membuka project Nilai
6. Klik menu Project ⇒ Add Windows Form untuk menambahkan form baru
7. Ketik nama form FORMMATAKULIAH
8. Klik tombol Add
9. Buat Desain form berikut



10. Isikan data Mata Kuliah

KODE	MATA KULIAH	SKS	SEMESTER
2MI101	Sistem Operasi Komputer	3	Semester 1
2MI102	Dasar Komputasi	2	Semester 1
2MI103	Dasar Akuntansi	2	Semester 1
2MI104	Lab. Dasar Akuntansi	1	Semester 1
2MI105	Logika dan Algoritma	3	Semester 1
2MI106	Aplikasi Administrasi Perkantoran	3	Semester 1
2MIA07	Desain Grafis	3	Semester 1
2MIA08	Pengantar Manajemen	2	Semester 1
2MIA09	Teknologi Informasi dan E-Commerce	2	Semester 1
2MIA10	Sistem Basis Data	3	Semester 2
2MIA11	Teknologi Internet	3	Semester 2
3MIA12	Komputer Akuntansi	2	Semester 2
3MI213	Pemrograman Basis Data	2	Semester 2
4MI214	Praktikum Pemrograman Basis Data	1	Semester 2
2MI215	Komunikasi Data	2	Semester 2
2MI216	Aplikasi Pengolah Data	3	Semester 2
3MI217	Sistem Informasi Manajemen	2	Semester 2

2MI218	Animasi	3	Semester 2
4MI319	Pemrograman SQL	2	Semester 3
2MI320	Dasar Pemrograman Visual	3	Semester 3
4MI321	Pemrograman Paket Niaga	3	Semester 3
3MI322	Manajemen Operasional	2	Semester 3
1MI323	*Pendidikan Agama*	*2*	Semester 3
4MI324	Jaringan Komputer-1	3	Semester 3
2MIB25	Bahasa Inggris 1(General English)	2	Semester 3
4MIB26	Desain Web	3	Semester 3
4MIB27	Jaringan Komputer-2	3	Semester 3
3MIB28	Pemrograman Visual 1	2	Semester 4
4MIB29	Praktikum Pemrograman Visual 1	1	Semester 4
1MIB30	*Pend Kewarganegaraan dan Pancasila*	3	Semester 4
3MI431	Pemrograman Berorientasi Objek	3	Semester 4
2MI432	*Bahasa Inggris 2(Corespondence)*	2	Semester 4
4MI433	Database Server	3	Semester 4
4MI434	Pemrograman Web-1	3	Semester 4
3MI435	Manajemen Sumber Daya Manusia	2	Semester 4
2MI536	Pemrograman Delphi-1	2	Semester 5
4MI537	Praktikum Pemrograman Delphi-1	1	Semester 5
3MI538	Bahasa Inggris 3 (Conversation)	2	Semester 5
4MI539	Pemrograman Web-2	3	Semester 5
3MI540	Perilaku Organisasi	2	Semester 5
4MIC41	Pemrograman Client Server - 1	3	Semester 5
5MIC42	Kewirausahaan dan Etika Profesi	2	Semester 5
1MIC43	Bahasa Indonesia	*2*	Semester 5
3MIC44	Keamanan Sistem Komputer	2	Semester 6
4MIC45	Aplikasi Pemrograman .Net-1	3	Semester 6
3MIC46	Manajemen Perubahan	2	Semester 6
3MI647	Pemrograman Delphi-2	*2*	Semester 6
4MI648	Praktikum Pemrograman Delphi-2	1	Semester 6
3MI649	Statistik	2	Semester 6
3MI650	Aplikasi Pemrograman Web 1	3	Semester 6
3MI651	Bahasa Inggris 4 (TOEFL)	2	Semester 6
4MI752	Aplikasi Pemrograman .Net-2	3	Semester 7
3MI753	Analisa dan Perancangan Sistem Informasi	2	Semester 7
4MI754	Pemrograman Client Server - 2	3	Semester 7
4MI755	Komputer Statistik	3	Semester 7
3MI756	Metode Penelitian	2	Semester 7
3MID57	Manajemen Strategik	2	Semester 7
3MID58	Sistem Pendukung Manajemen	2	Semester 7

4MID59	Rekayasa Perangkat Lunak	3	Semester 7
4MID60	Presentasi Multimedia	3	Semester 8
4MID61	Aplikasi Pemrograman Web 2	3	Semester 8
5MI862	Skripsi	6	Semester 8

11. Ketikkan Kode programnya sebagai berikut :

```
Imports System.Data
Imports System.Data.OleDb

Public Class FormMATAKULIAH
    Public SQLSTR As String
    Public KONEKSI As OleDb.OleDbConnection
    Public CMD As OleDb.OleDbCommand
    Public RS As OleDb.OleDbDataAdapter
    Public TBMATKUL As New DataTable
    Public KONEKSISTRING As String
    Public LOKASI As String


    Private Sub KONEKDB()
        On Error GoTo NO_KONEKSI
        LOKASI = System.Environment.CurrentDirectory
        KONEKSISTRING = "Provider=Microsoft.JET.OLEDB.4.0;Persist Security Info=True;Data Source=" & LOKASI & "\DBNILAI.MDB"
        KONEKSI = New OleDb.OleDbConnection(KONEKSISTRING)
        KONEKSI.Open()
        Exit Sub

NO_KONEKSI:
        MsgBox("Error !" & Chr(13) & Chr(13) & _
            "Gagal menyambung Server !", vbCritical, "Peringatan !")
        End
    End Sub
    Private Sub KOSONG()
        EDKODEMK.Text = ""
        EDNAMAMK.Text = ""
        EDSKS.Text = ""
        EDSEMESTER.Text = ""
        EDKODEMK.Focus()
    End Sub


  Private Sub SETKOLOM()
      DGMatkul.Columns(0).Width = 100
      DGMatkul.Columns(1).Width = 350
      DGMatkul.Columns(2).Width = 100
      DGMatkul.Columns(3).Width = 200
  End Sub
  Private Sub TAMPILMATKUL()
      SQLSTR = "SELECT * FROM MATAKULIAH ORDER BY KODEMK ASC"
      RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
      TBMATKUL.Clear()
      RS.Fill(TBMATKUL)
      DGMatkul.DataSource = TBMATKUL
      SETKOLOM()
  End Sub
  Private Sub CARIDATA()
      SQLSTR = "SELECT * FROM MATAKULIAH where NAMAMK LIKE '%" & TSQL.Text & "%' ORDER BY KODEMK ASC"
      RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
      TBMATKUL.Clear()
      RS.Fill(TBMATKUL)
      DGMatkul.DataSource = TBMATKUL
      SETKOLOM()
  End Sub
```

```
Private Sub SIMPAN()
    SQLSTR = "Select * from matakuliah where KODEMK='" & EDKODEMK.Text & "'"
    'TSQL.Text = SQLSTR
    RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
    TBMATKUL.Clear()
    RS.Fill(TBMATKUL)
    DGMatkul.DataSource = TBMATKUL
    If TBMATKUL.Rows.Count > 0 Then
        SQLSTR = "Update matakuliah set KODEMK='" & EDKODEMK.Text & "',NAMAMK='" & EDNAMAMK.Text & "', SKS='" & _
                 EDSKS.Text & "', SEMESTER = '" & EDSEMESTER.Text & "' WHERE KODEMK='" & EDKODEMK.Text & "'"
        TSQL.Text = SQLSTR

        CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
        CMD.ExecuteNonQuery()
        MsgBox("Berhasil update data", vbOKOnly)
    Else
        SQLSTR = "insert into matakuliah (KODEMK,NAMAMK,SKS,SEMESTER) values ('" & EDKODEMK.Text & "','" & _
            EDNAMAMK.Text & "','" & EDSKS.Text & "','" & EDSEMESTER.Text & "')"
        'TSQL.Text =
        CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
        CMD.ExecuteNonQuery()
        MsgBox("Berhasil simpan data", vbOKOnly)
    End If
    TAMPILMATKUL()
End Sub
```

```
Private Sub HAPUS()
    Dim X As String
    X = MsgBox("Data Mahasiswa '" & EDKODEMK.Text & "' Jadi Dihapus ?", MsgBoxStyle.YesNo)
    If X = vbYes Then
        SQLSTR = "Delete from matakuliah where KODEMK='" & EDKODEMK.Text & "' "
        CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
        CMD.ExecuteNonQuery()
    End If
    KOSONG()
    TAMPILMATKUL()
End Sub
Private Sub FormMATAKULIAH_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    KONEKDB()
    TAMPILMATKUL()
    DGMatkul.MultiSelect = True


End Sub
```

```
Private Sub BTRefresh_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTRefresh.Click
    TAMPILMATKUL()
    KOSONG()
End Sub
```

```
Private Sub BTSimpan_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTSimpan.Click
    SIMPAN()
    TAMPILMATKUL()
End Sub
```

```
Private Sub BTHapus_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTHapus.Click
    HAPUS()
End Sub
```

```
Private Sub BTClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTClose.Click
    Close()
End Sub
```

```
Private Sub EDKODEMK_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles EDKODEMK.Leave
    Dim BARIS As Integer

    If EDKODEMK.Text <> "" Then
        SQLSTR = "Select * from matakuliah where KODEMK='" & EDKODEMK.Text & "'"
        RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
        TBMATKUL.Clear()
        RS.Fill(TBMATKUL)
        DGMatkul.DataSource = TBMATKUL

        If TBMATKUL.Rows.Count <> 0 Then
            BARIS = DGMatkul.CurrentRow.Index
            EDNAMAMK.Text = DGMatkul.Item(1, BARIS).Value
            EDSKS.Text = DGMatkul.Item(2, BARIS).Value
            EDSEMESTER.Text = DGMatkul.Item(3, BARIS).Value
        End If
    End If
    TAMPILMATKUL()
End Sub
```

```
Private Sub BTCARI_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCARI.Click
    CARIDATA()
End Sub
```

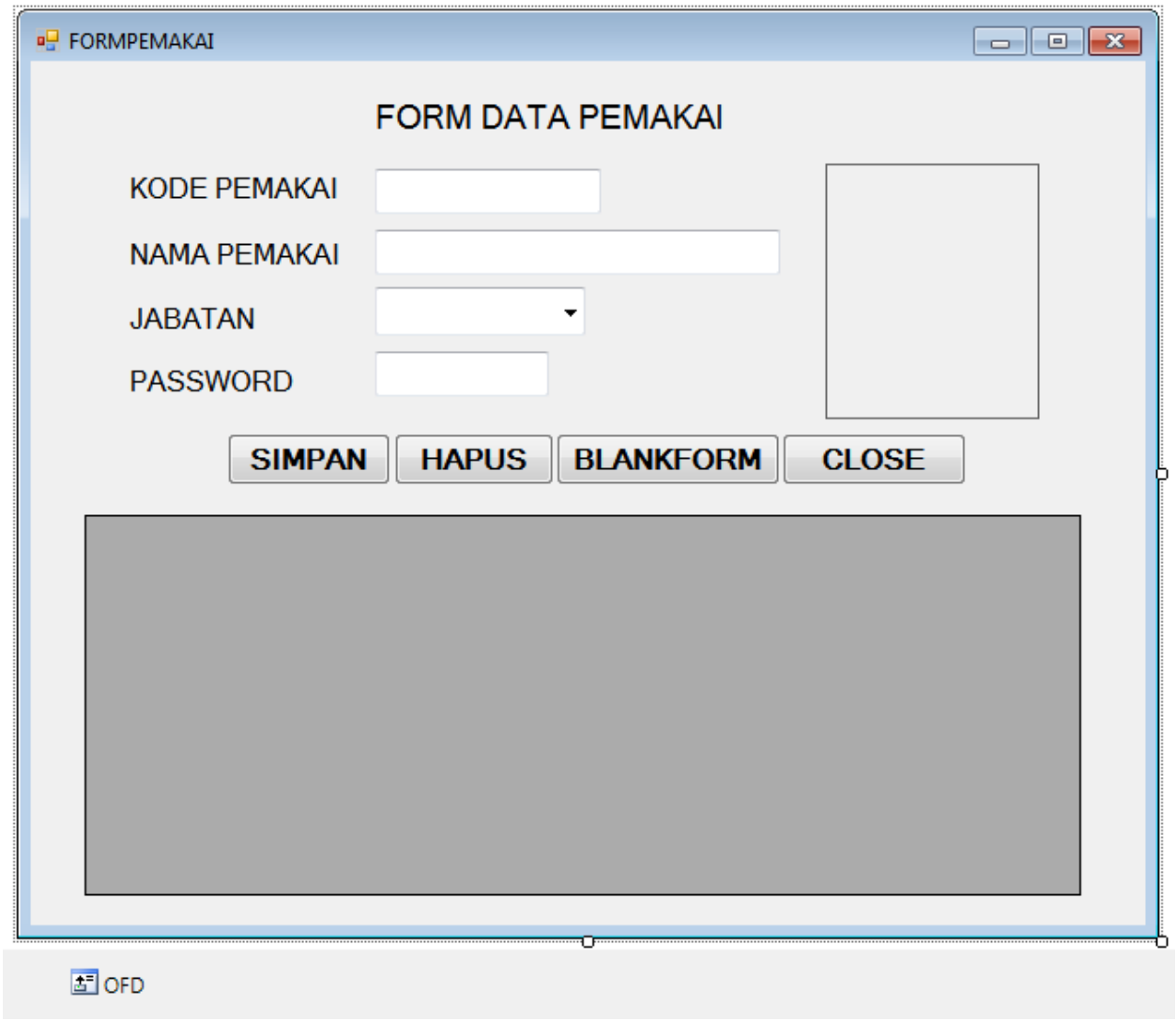
```
Private Sub TSQL_KeyPress(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles TSQL.Ke
    If e.KeyChar = Chr(13) Then
        BTCARI.PerformClick()
    End If
End Sub
```

6.3 FORM DATA PEMAKAI

1. Buka database DBNILAI.MDB
2. Buat struktur tabel berikut :

Field Name	Data Type	
KODEPK	Text	FieldSize : 15
NAMAPK	Text	FieldSize : 25
JABATAN	Text	FieldSize : 12
PASSWORD	Text	FieldSize : 10
FOTO	Text	FieldSize : 100

3. Simpan tabel dengan nama PEMAKAI
4. Tutup program MS Access
5. Double klik file PROJECTNILAI.SLN untuk membuka project Nilai
6. Klik menu Project ⇒ Add Windows Form untuk menambahkan form baru
7. Ketik nama form FORMPEMAKAI
8. Klik tombol Add
9. Buat Desain form berikut



Keterangan :

ComboBox TXTJABATAN diisi dengan DOSEN dan ADMINISTRASI (Propertie Items)

Kode programnya sebagai berikut :

```
Imports System.Data
Imports System.Data.OleDb

Public Class FormPEMAKAI
    Public SQLSTR As String
    Public KONEKSI As OleDb.OleDbConnection
    Public CMD As OleDb.OleDbCommand
    Public RS As OleDb.OleDbDataAdapter
    Public TBPEMAKAI As New DataTable
    Public KONEKSISTRING As String
    Public LOKASI As String
    Public XFOTO As String
```

```
    Private Sub KONEKDB()
        On Error GoTo NO_KONEKSI
        LOKASI = System.Environment.CurrentDirectory
        KONEKSISTRING = "Provider=Microsoft.JET.OLEDB.4.0;Persist Security Info=True;Data Source=" & LOKASI & "\DBNILAI.MDB"
        KONEKSI = New OleDb.OleDbConnection(KONEKSISTRING)
        KONEKSI.Open()
        Exit Sub

NO_KONEKSI:
        MsgBox("Error !" & Chr(13) & Chr(13) & _
            "Gagal menyambung Server !", vbCritical, "Peringatan !")
        End
    End Sub
    Private Sub KOSONG()
        EDKODEPK.Text = ""
        EDNAMAPK.Text = ""
        EDJABATAN.Text = ""
        EDPASSWORD.Text = ""
        EDFOTO.Text = ""
        PBFOTO.Image = Nothing
        EDKODEPK.Focus()
    End Sub
```

```
Private Sub SETKOLOM()
    DataGrid1.Columns(0).Width = 100
    DataGrid1.Columns(1).Width = 350
    DataGrid1.Columns(2).Width = 100
    DataGrid1.Columns(3).Visible = False
    DataGrid1.Columns(4).Width = 200
End Sub
Private Sub TAMPILDATA()
    SQLSTR = "SELECT * FROM PEMAKAI ORDER BY KODEPK ASC"
    RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
    TBPEMAKAI.Clear()
    RS.Fill(TBPEMAKAI)
    DataGrid1.DataSource = TBPEMAKAI
    SETKOLOM()
End Sub
Private Sub CARIDATA()
    SQLSTR = "SELECT * FROM PEMAKAI WHERE NAMAPK LIKE '%" & TSQL.Text & "%'ORDER BY KODEPK ASC"
    RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
    TBPEMAKAI.Clear()
    RS.Fill(TBPEMAKAI)
    DataGrid1.DataSource = TBPEMAKAI
    SETKOLOM()
End Sub
```

```
    Private Sub SIMPAN()
        SQLSTR = "Select * from PEMAKAI where KODEPK='" & EDKODEPK.Text & "'"
        'TSQL.Text = SQLSTR
        RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
        TBPEMAKAI.Clear()
        RS.Fill(TBPEMAKAI)
        DataGrid1.DataSource = TBPEMAKAI
        If TBPEMAKAI.Rows.Count > 0 Then
            SQLSTR = "Update PEMAKAI set  NAMAPK='" & EDNAMAPK.Text & "', JABATAN='" & _
                   EDJABATAN.Text & "', PASSWORDS = '" & EDPASSWORD.Text & "', FOTO = '" & EDFOTO.Text & _
                   "' WHERE KODEPK='" & EDKODEPK.Text & "'"
            'TSQL.Text = SQLSTR

            CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
            CMD.ExecuteNonQuery()
            MsgBox("Berhasil update data", vbOKOnly)
        Else
            SQLSTR = "insert into PEMAKAI (KODEPK,NAMAPK,JABATAN,PASSWORDS,FOTO) values ('" & EDKODEPK.Text & "','" & _
                EDNAMAPK.Text & "','" & EDJABATAN.Text & "','" & EDPASSWORD.Text & "','" & EDFOTO.Text & "')"

            'TSQL.Text = SQLSTR
            CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
            CMD.ExecuteNonQuery()
            MsgBox("Berhasil simpan data", vbOKOnly)
        End If
        TAMPILDATA()
    End Sub
```

```
    Private Sub HAPUS()
        Dim X As String
        X = MsgBox("Data PEMAKAI '" & EDKODEPK.Text & "' Jadi Dihapus ?", MsgBoxStyle.YesNo)
        If X = vbYes Then
            SQLSTR = "Delete from PEMAKAI where KODEPK='" & EDKODEPK.Text & "' "
            CMD = New OleDb.OleDbCommand(SQLSTR, KONEKSI)
            CMD.ExecuteNonQuery()
        End If
        KOSONG()
        TAMPILDATA()
    End Sub
    Private Sub FormPEMAKAI_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        KONEKDB()
        TAMPILDATA()

    End Sub

    Private Sub PBFOTO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        On Error Resume Next
        OFD.ShowDialog()
        PBFOTO.Image = Image.FromFile(OFD.FileName)
        XFOTO = OFD.FileName
    End Sub
```

```
Private Sub EDKODEPK_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles EDKODEPK.Leave
    On Error Resume Next
    Dim BARIS As Integer

    If EDKODEPK.Text <> "" Then
        SQLSTR = "Select * from pemakai where KODEPK='" & EDKODEPK.Text & "'"
        RS = New OleDb.OleDbDataAdapter(SQLSTR, KONEKSI)
        TBPEMAKAI.Clear()
        RS.Fill(TBPEMAKAI)
        DataGrid1.DataSource = TBPEMAKAI

        If TBPEMAKAI.Rows.Count <> 0 Then
            BARIS = DataGrid1.CurrentRow.Index
            EDNAMAPK.Text = DataGrid1.Item(1, BARIS).Value
            EDJABATAN.Text = DataGrid1.Item(2, BARIS).Value
            EDPASSWORD.Text = DataGrid1.Item(3, BARIS).Value
            EDFOTO.Text = DataGrid1.Rows.Item(DataGrid1.CurrentRow.Index).Cells(4).Value
            If EDFOTO.Text <> "" Then
                PBFOTO.Image = Image.FromFile(DataGrid1.Rows.Item(DataGrid1.CurrentRow.Index).Cells(4).Value)
            End If
            'Else
            '    MessageBox.Show("Data Tidak Ditemukan")
            '    KOSONG()
        End If
    End If
    TAMPILDATA()
End Sub
```

```
Private Sub PBFOTO_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PBFOTO.Click
    On Error Resume Next
    OFD.ShowDialog()
    PBFOTO.Image = Image.FromFile(OFD.FileName)
    XFOTO = OFD.FileName
    EDFOTO.Text = XFOTO
End Sub

Private Sub BTRefresh_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTRefresh.Click
    TAMPILDATA()
    KOSONG()
End Sub

Private Sub BTSimpan_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTSimpan.Click
    SIMPAN()
    TAMPILDATA()
End Sub

Private Sub BTHapus_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTHapus.Click
    HAPUS()
End Sub

Private Sub BTClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTClose.Click
    Close()
End Sub
```

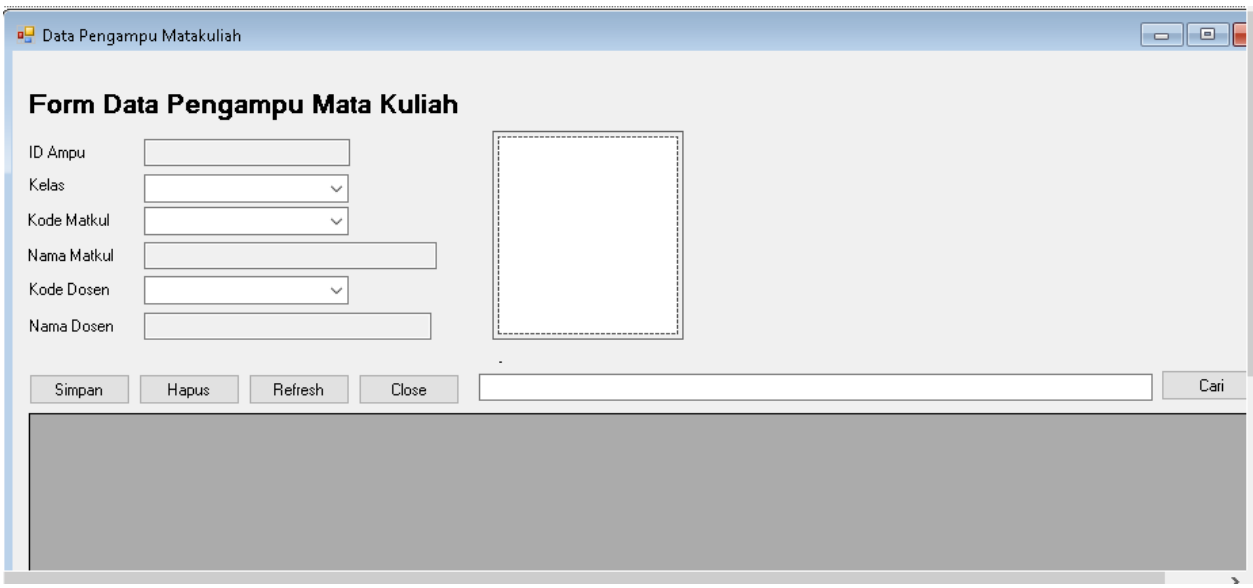
```
Private Sub BTCARI_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCARI.Click
    CARIDATA()
End Sub

Private Sub TSQL_KeyPress(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles TSQL.Key
    If e.KeyChar = Chr(13) Then
        BTCARI.PerformClick()
    End If
End Sub
```

6.4 FORM PENGAMPU MATA KULIAH

1. Buka database DBNILAI.MDB
2. Buat struktur tabel berikut :

Field Name	Data Type	
KELAS	Text	FieldSize : 10
KODEMK	Text	FieldSize : 6
KODEPK	Text	FieldSize : 15

3. Simpan tabel dengan nama PENGAMPU
4. Tutup program MS Access
5. Double klik file PROJECTNILAI.SLN untuk membuka project Nilai
6. Klik menu Project ⇒ Add Windows Form untuk menambahkan form baru
7. Ketik nama form FORMPENGAMPU
8. Klik tombol Add
9. Buat Desain form berikut



10. Ketik kode program di bawah ini

```
Imports System.Data
Imports System.Data.OleDb
Public Class FORMPENGAMPU
    Public LOKASI As String
    Public koneksi As String
    Public sqlstr As String
    Public conn As New OleDb.OleDbConnection
    Public CMD As New OleDb.OleDbCommand
    Public DA As New OleDb.OleDbDataAdapter
    Public DT As New DataTable
    Public DS As New DataSet
    Public DR As OleDbDataReader
    Public XKODEMK As String
    Public XKODEDS As String
    Public XFOTO As String
    Private Sub FORMPENGAMPU_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        LOKASI = System.Environment.CurrentDirectory
        koneksi = "PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE =" & LOKASI & "\DBNILAI.MDB"
        conn = New OleDb.OleDbConnection(koneksi)
        conn.Open()

        sqlstr = "SELECT DISTINCT KELAS FROM MAHASISWA ORDER BY KELAS"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        TXTKELAS.Items.Clear()
        While DR.Read
            TXTKELAS.Items.Add(DR("KELAS"))
        End While
        DR.Close()

        sqlstr = "SELECT KODEMK,NAMAMK FROM MATAKULIAH ORDER BY NAMAMK"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTKODEMK.Items.Add(DR("KODEMK") & "  " & DR("NAMAMK"))
        End While
        DR.Close()

        sqlstr = "SELECT KODEPK,NAMAPK FROM PEMAKAI ORDER BY NAMAPK"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTKODEDS.Items.Add(DR("KODEPK") & "  " & DR("NAMAPK"))
        End While
        DR.Close()
    End Sub
    Sub KOSONG()
        TXTKELAS.Text = ""
        TXTKODEMK.Text = ""
        TXTNAMAMK.Text = ""
        TXTKODEDS.Text = ""
        TXTNAMADS.Text = ""
        PBFOTO.Image = Nothing
        TXTKELAS.Focus()
    End Sub
```

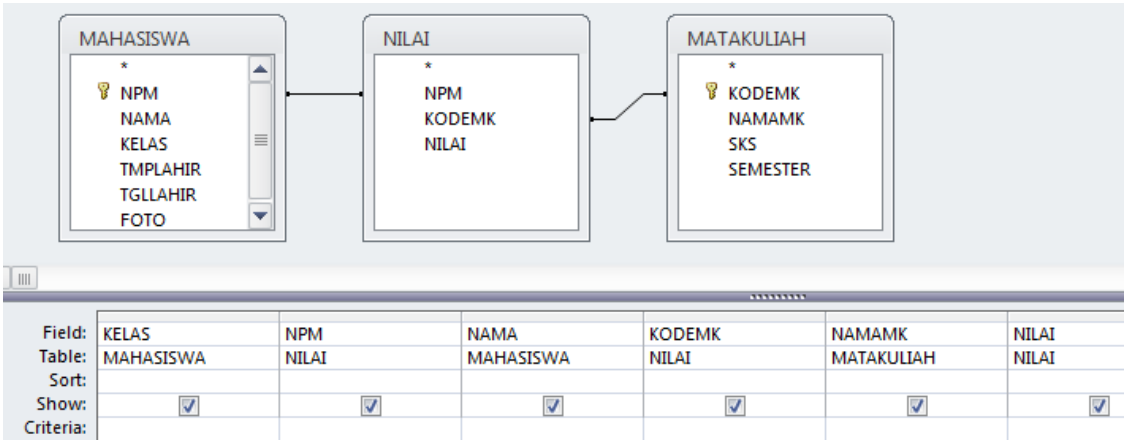
```
Sub DGV()
    On Error Resume Next
    sqlstr = "SELECT KELAS,PENGAMPU.KODEMK,NAMAMK,PENGAMPU.KODEPK,NAMAPK FROM PEMAKAI INNER JOIN " & _
             "(MATAKULIAH INNER JOIN PENGAMPU ON MATAKULIAH.KODEMK = PENGAMPU.KODEMK) " & _
             "ON PEMAKAI.KODEPK = PENGAMPU.KODEPK ORDER BY KELAS,NAMAMK"
    DA = New OleDbDataAdapter(sqlstr, conn)
    DT.Clear()
    DA.Fill(DT)
    DGVPENGAMPU.DataSource = DT
End Sub
Private Sub FORMPENGAMPU_Activated(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Activated
    DGV()
End Sub
Private Sub TXTKODEMK_SelectedValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    XKODEMK = Microsoft.VisualBasic.Left(TXTKODEMK.Text, 6)
    sqlstr = "SELECT KODEMK,NAMAMK FROM MATAKULIAH WHERE KODEMK='" & XKODEMK & "'"
    CMD = New OleDbCommand(sqlstr, conn)
    DR = CMD.ExecuteReader
    While DR.Read
        TXTNAMAMK.Text = DR("NAMAMK")
    End While
    DR.Close()
    TXTKODEDS.Focus()
End Sub
```

6.5 FORM DATA NILAI MAHASISWA

1. Buka database DBNILAI.MDB
2. Buat struktur tabel berikut :

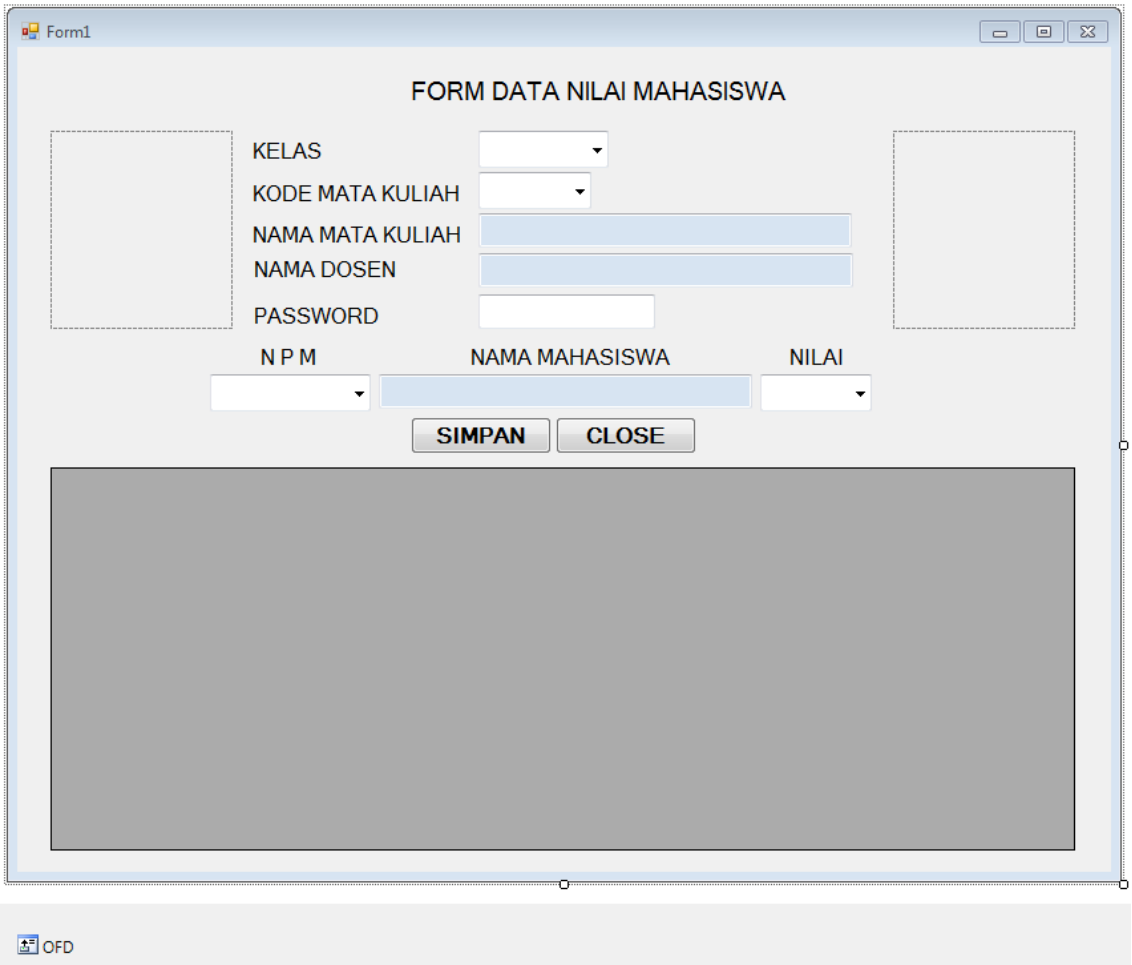
Field Name	Data Type	
NPM	Text	FieldSize : 15
KODEMK	Text	FieldSize : 6
NILAI	Text	FieldSize : 1

3. Simpan tabel dengan nama NILAI
4. Buat desain Query



5. Simpan dengan nama QUERYNILAI
6. Tutup program MS Access

7. Double klik file PROJECTNILAI.SLN untuk membuka project Nilai
8. Klik menu Project ⇒ Add Windows Form untuk menambahkan form baru
9. Ketik nama form FORMNILAI
10. Klik tombol Add
11. Buat Desain form berikut



Combo Box NILAI diisi : A,B,C,D,E
Combo Box yang lain Kosong

19. Ketik kode program di bawah ini

```
Imports System.Data
Imports System.Data.OleDb
Public Class FORMNILAIMHS
    Public LOKASI As String
    Public koneksi As String
    Public sqlstr As String
    Public conn As New OleDb.OleDbConnection
    Public CMD As New OleDb.OleDbCommand
    Public CMDNILAI As New OleDb.OleDbCommand
    Public DA As New OleDb.OleDbDataAdapter
    Public DT As New DataTable
    Public DS As New DataSet
    Public DR As OleDbDataReader
    Public XFOTO As String
    Public VKODEMK As String
    Public VKODEDS As String
    Public XPASSWORD As String
    Public VNPM As String
    Private Sub FORMNILAI_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        LOKASI = System.Environment.CurrentDirectory
        koneksi = "PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE =" & LOKASI & "\DBNILAI.MDB"
        conn = New OleDb.OleDbConnection(koneksi)
        conn.Open()

        sqlstr = "SELECT DISTINCT KELAS FROM PENGAMPU ORDER BY KELAS"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        TXTKELAS.Items.Clear()
        While DR.Read
            TXTKELAS.Items.Add(DR("KELAS"))
        End While
        DR.Close()
    End Sub
    Sub DGV()
        sqlstr = "SELECT * FROM QUERYNILAI WHERE KELAS='" & TXTKELAS.Text & "' AND NILAI!KODEMK='" & VKODEMK & "'"
        DA = New OleDbDataAdapter(sqlstr, conn)
        DT.Clear()
        DA.Fill(DT)
        DGVNILAI.DataSource = DT
    End Sub
```

```
    Private Sub FORMPENGAMPU_Activated(ByVal sender As Object, ByVal e As System.EventArgs) Han
        TXTKELAS.Text = ""
        TXTKODEMK.Text = ""
        TXTNAMAMK.Text = ""
        TXTNAMADS.Text = ""
        TXTPASSWORD.Text = ""
        PBFOTO.Image = Nothing
        PBMHS.Image = Nothing
        TXTNPM.Enabled = False
        TXTNILAI.Enabled = False
        TXTKELAS.Focus()
        DGV()
    End Sub
    Private Sub TXTKELAS_SelectedValueChanged(ByVal sender As System.Object, ByVal e As System.
        sqlstr = "SELECT PENGAMPU.KODEMK,NAMAMK FROM MATAKULIAH INNER JOIN PENGAMPU ON " & _
                 "MATAKULIAH.KODEMK=PENGAMPU.KODEMK WHERE KELAS='" & TXTKELAS.Text & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        TXTKODEMK.Items.Clear()
        While DR.Read
            TXTKODEMK.Items.Add(DR("KODEMK") & "  " & DR("NAMAMK"))
        End While
        DR.Close()

        sqlstr = "SELECT NPM,NAMA FROM MAHASISWA WHERE KELAS='" & TXTKELAS.Text & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTNPM.Items.Add(DR("NPM") & "  " & DR("NAMA"))
        End While
        DR.Close()
        DGV()
    End Sub
```

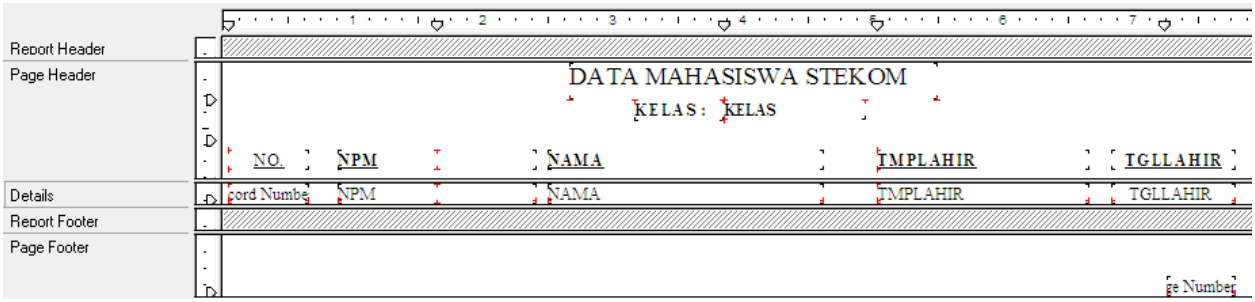
```
    Private Sub TXTKODEMK_SelectedValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handle
        On Error GoTo SALAH
        VKODEMK = Microsoft.VisualBasic.Left(TXTKODEMK.Text, 6)
        sqlstr = "SELECT KODEMK,NAMAMK FROM MATAKULIAH WHERE KODEMK='" & VKODEMK & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTNAMAMK.Text = DR("NAMAMK")
        End While
        DR.Close()

        sqlstr = "SELECT KODEPK FROM PENGAMPU WHERE KELAS='" & TXTKELAS.Text & "' AND KODEMK='" & VKODEMK & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            VKODEDS = DR("KODEPK")
        End While
        DR.Close()

        sqlstr = "SELECT KODEPK,NAMAPK,FOTO,PASSWORD FROM PEMAKAI WHERE KODEPK='" & VKODEDS & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTNAMADS.Text = DR("NAMAPK")
            PBFOTO.Image = Image.FromFile(DR("FOTO"))
            XPASSWORD = DR("PASSWORD")
        End While
        DR.Close()
        TXTNPM.Enabled = False
        TXTNILAI.Enabled = False
        TXTPASSWORD.Text = ""
        TXTPASSWORD.Focus()
        DGV()
        Exit Sub
SALAH:
        PBFOTO.Image = Nothing
    End Sub
```

```
    Private Sub TXTPASSWORD_KeyPress(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyPressEventA
        If e.KeyChar = Chr(13) Then
            If UCase(TXTPASSWORD.Text) = UCase(XPASSWORD) Then
                TXTNPM.Enabled = True
                TXTNILAI.Enabled = True
                TXTNPM.Focus()
            Else
                TXTNPM.Enabled = False
                TXTNILAI.Enabled = False
                TXTPASSWORD.Text = ""
                TXTPASSWORD.Focus()
            End If
        End If
    End Sub
    Private Sub TXTNPM_SelectedValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TX
        VNPM = Microsoft.VisualBasic.Left(TXTNPM.Text, 12)
        sqlstr = "SELECT NPM,NAMA FROM MAHASISWA WHERE NPM='" & VNPM & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTNAMAMHS.Text = DR("NAMA")
        End While
        DR.Close()
        TXTNILAI.Focus()
    End Sub
    Private Sub BTSIMPAN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTSIMPAN.Click
        sqlstr = "INSERT INTO NILAI VALUES ('" & VNPM & "','" & VKODEMK & "','" & TXTNILAI.Text & "')"
        CMD = New OleDb.OleDbCommand(sqlstr, conn)
        CMD.ExecuteNonQuery()
        TXTNPM.Text = ""
        TXTNAMAMHS.Text = ""
        TXTNILAI.Text = ""
        TXTNPM.Focus()
        DGV()
    End Sub
    Private Sub BTCLOSE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCLOSE.Click
        Close()
    End Sub
    Private Sub FORMNILAI_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Leave
        conn.Close()
    End Sub
End Class
```

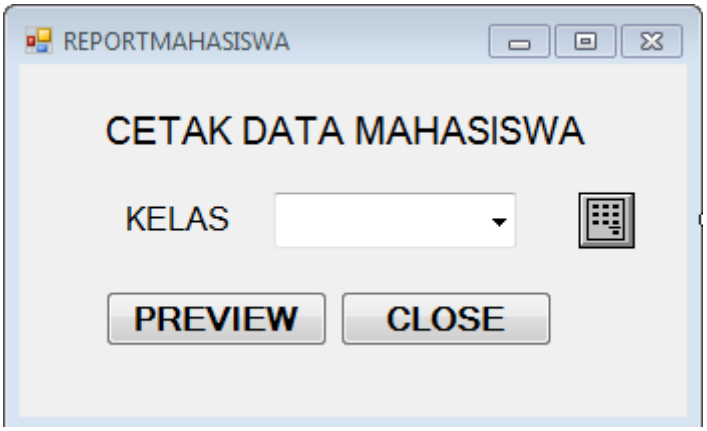
```
    Private Sub TXTKODEDS_SelectedValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        On Error GoTo SALAH
        XKODEDS = Microsoft.VisualBasic.Left(TXTKODEDS.Text, 12)
        sqlstr = "SELECT KODEPK,NAMAPK,FOTO FROM PEMAKAI WHERE KODEPK='" & XKODEDS & "'"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        While DR.Read
            TXTNAMADS.Text = DR("NAMAPK")
            PBFOTO.Image = Image.FromFile(DR("FOTO"))
        End While
        DR.Close()
        TXTKELAS.Focus()
        Exit Sub
SALAH:
        PBFOTO.Image = Nothing
    End Sub
    Private Sub BTSIMPAN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTSIMPAN.Click
        DR.Close()
        sqlstr = "SELECT * FROM PENGAMPU WHERE KELAS='" & TXTKELAS.Text & "' AND KODEMK='" & XKODEMK & "'"
        DA = New OleDbDataAdapter(sqlstr, conn)
        DT.Clear()
        DA.Fill(DT)
        DGVPENGAMPU.DataSource = DT
        If DT.Rows.Count > 0 Then
            sqlstr = "Update PENGAMPU SET KODEPK='" & XKODEDS & "' WHERE KELAS='" & TXTKELAS.Text & "' AND KODEMK='
            CMD = New OleDb.OleDbCommand(sqlstr, conn)
            CMD.ExecuteNonQuery()
        Else
            sqlstr = "INSERT INTO PENGAMPU VALUES ('" & TXTKELAS.Text & "','" & XKODEMK & "','" & XKODEDS & "')"
            CMD = New OleDb.OleDbCommand(sqlstr, conn)
            CMD.ExecuteNonQuery()
        End If
        KOSONG()
        DGV()
    End Sub
    Private Sub BTHAPUS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTHAPUS.Click
        Dim X As String
        X = MsgBox("DATA PENGAMPU JADI DIHAPUS !", MsgBoxStyle.YesNo)
        If X = vbYes Then
            sqlstr = "DELETE FROM PENGAMPU"
            CMD = New OleDb.OleDbCommand(sqlstr, conn)
            CMD.ExecuteNonQuery()
        End If
        KOSONG()
        DGV()
    End Sub
    Private Sub BTCLOSE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCLOSE.Click
        Close()
    End Sub
    Private Sub FORMPENGAMPU_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Leave
        conn.Close()
    End Sub
End Class
```

6.6 MENCETAK DATA DENGAN CRYSTAL REPORT

1. Aktifkan Program Crystal Report
2. Buat desain report seperti di bawah ini



3. Simpan desain report dengan nama MAHASISWA.RPT pada folder \PRNILAI\PRNILAI\bin\Debug
4. Buka PROJECTNILAI
5. Tambahkan form baru dengan nama REPORTMAHASISWA
6. Jika pada ToolBox belum ada komponen Crystal Report, lakukan langkah berikut :
   a. Klik Tools ⇒ Choose ToolBox Items
   b. Klik tab COM Components
   c. Klik kotak di sebelah kiri Crystal Report Control
   d. Klik OK
7. Buat desain form di bawah ini



8. Ketik kode program di bawah ini

```
Imports System.Data
Imports System.Data.OleDb
Public Class REPORTMAHASISWA
    Public LOKASI As String
    Public koneksi As String
    Public sqlstr As String
    Public conn As New OleDb.OleDbConnection
    Public CMD As New OleDb.OleDbCommand
    Public DR As OleDbDataReader
    Private Sub REPORTMAHASISWA_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        LOKASI = System.Environment.CurrentDirectory
        koneksi = "PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE =" & LOKASI & "\DBNILAI.MDB"
        conn = New OleDb.OleDbConnection(koneksi)
        conn.Open()

        sqlstr = "SELECT DISTINCT KELAS FROM MAHASISWA ORDER BY KELAS"
        CMD = New OleDbCommand(sqlstr, conn)
        DR = CMD.ExecuteReader
        TXTKELAS.Items.Clear()
        While DR.Read
            TXTKELAS.Items.Add(DR("KELAS"))
        End While
        DR.Close()
    End Sub
    Private Sub BTPREVIEW_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTPREVIEW.Click
        CR.SelectionFormula = "{MAHASISWA.KELAS}='" & TXTKELAS.Text & "'"
        CR.ReportFileName = "MAHASISWA.RPT"
        CR.WindowState = Crystal.WindowStateConstants.crptMaximized
        CR.RetrieveDataFiles()
        CR.Action = 0
    End Sub
    Private Sub BTCLOSE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTCLOSE.Click
        Close()
    End Sub
    Private Sub FORMPENGAMPU_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Leave
        conn.Close()
    End Sub
End Class
```

DAFTAR PUSTAKA

Raharjo Budi, 2021, “Sistem Manajemen Database “, Semarang : Yayasan Prima Agus Teknik

Rozikin Khoirur, 2021, “Aplikasi Inventory Barang Dengan Visual Basic .Net “, Semarang : Yayasan Prima Agus Teknik

Santoso Joseph T, 2021, “Pemrograman Kompetitif “, Semarang : Yayasan Prima Agus Teknik

Sudirman Bagus , 2021, “VB .NET Untuk Pemula “, Semarang : Yayasan Prima Agus Teknik

Wibowo Agus , 2022, “Dasar Komputer Digital “, Semarang : Yayasan Prima Agus Teknik

IMPLEMENTASI PEMROGRAMAN Visual VB.NET

Indra Ava Dianta, S.Kom., M.T

BIO DATA PENULIS

Indra Ava Dianta lahir di Salatiga, Provinsi Jawa Tengah. Menyelesaikan studi pendidikan Strata-1 di Sekolah Tinggi Elektronika dan Komputer Pat pada tahun 2012 dengan jurusan yang di ambil adalah Sistem Komputer. Kemudian dilanjutkan dengan menempuh pendidikan strata-2 di universitas islam sultan agung semarang jurusan magister teknik elektro pada fakultas tekni industri tahun 2013 dan meraih gelar magister teknik pada 2016, focus kajian pada bidang Simulasi Sistem Kendali dan Arus Lemah.

Mengawali Karir sebagai Asisten Dosen pada tahun 2010 di kampus untuk membentuk pribadi sebagai pengajar profesional. Pada tahun 2016 setelah menyelesaikan Studi S-2 mendapat amanah menjadi dosen di Sekolah Tinggi Elektronika dan Komputer. Sebagai Dosen di Bidang Komputer, mata kuliah yang diampu diantaranya Logika dan Algoritma Pemrograman, Pemrograman Visual, Pemrograman Web. Karir sekarang menjadi Staff LPPM Universitas Sains dan Teknologi Komputer pada tahun 2020 dan banyak menghasilkan penelitian yang di danai oleh Dikti

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-99-6 (PDF)