# BAB 2

# LANDASAN TEORI

## 2.1 Teori Basis Data

### 2.1.1 Pengertian Basis Data

Basis data menurut Connoly (2002, p14) adalah kumpulan relasi logikal dari data/deskripsi data yang dapat digunakan bersama dan dibuat untuk memperoleh informasi yang dibutuhkan oleh perusahaan. Menurut Fathansyah (1999, p2) basis data ialah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

Basis data menurut C.J. Date (2000, p10) merupakan kumpulan data yang hampir tidak mengalami perubahan dan digunakan oleh sistem aplikasi pada beberapa perusahaan. Sedangkan menurut Kroenke, David.M. (1995, p10) basis data merupakan fakta, gambar atau suara yang bermanfaat bagi tugas tertentu.

Secara garis besar, basis data terdiri atas 2 kata, yaitu “basis” dan “data”. “Basis” kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan “data” adalah representasi fakta dunia nyata yang mewakili suatu obyek yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

#### 2.1.1.1 Tujuan Basis Data

Tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data adalah agar dapat memperoleh / menemukan kembali data (yang dicari) dengan mudah dan cepat (Fathansyah, 1999, p4). Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (obyektif) seperti berikut ini:

- Kecepatan dan kemudahan (*speed*)
- Efisiensi ruang penyimpanan (*space*)
- Keakuratan (*accuracy*)
- Ketersediaan (*availability*)
- Kelengkapan (*completeness*)
- Keamanan (*security*)
- Kebersamaan pemakaian (*sharability*)

Secara garis besar, basis data sangat bermanfaat untuk menampung data yang sangat besar dalam satu tempat dimana keamanannya terjamin. Sehingga diharapkan dengan adanya basis data dapat memudahkan segala aktivitas yang berhubungan dengan pengelolaan data.

### 2.1.2 *Database Management System* (*DBMS*)

*Database Management System (DBMS)* adalah sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara basis data dan menyediakan kontrol akses untuk basis data (Connoly, 2002,

p16). Menurut Connoly (2002, p18) lima komponen utama dalam lingkungan *DBMS* yaitu:

1. **Perangkat Keras (*Hardware*)**

   *DBMS* dan aplikasinya memerlukan perangkat keras untuk berjalan. Perangkat keras yang digunakan bisa berupa *PC*, *mainframe*, jaringan komputer. Terdiri dari:

- Penyimpanan sekunder (*magnetic disc*), perlengkapan I/O contoh: *disk drives, device Controller, I/O Channels*, dan lainnya.
- Prosessor dan memori utama, digunakan untuk mendukung saat eksekusi sistem perangkat lunak basis data.

2. **Perangkat Lunak (*Software*)**

   Komponen perangkat lunak terdiri dari perangkat lunak *DBMS* itu sendiri dan program aplikasi, bersama dengan sistem operasi (*OS*), termasuk perangkat lunak jaringan jika *DBMS* digunakan melalui jaringan.

3. **Data**

   Pada sebuah sistem basis data baik itu sistem *single-user* maupun sistem *multi-user* harus terintegrasi dan dapat digunakan bersama (*Integrated and Shared*). Data dalam basis data meliputi data operasional dan *meta data* (data mengenai data).

4. **Prosedur**

   Berupa instruksi dan aturan yang mengatur desain dan kegunaan basis data dan bagaimana cara menjalankan sistem basis data, termasuk cara untuk

masuk ke *DBMS*, menggunakan fasilitas *DBMS* atau program aplikasi, menjalankan dan menghentikan/mematikan *DBMS*, membuat *back-up* ke basis data, menangani kerusakan hardware atau software, mengubah struktur dari tabel.

**5. Orang**

Orang yang terlibat dengan sistem basis data, meliputi:

- *Application Programmers*, bertanggungjawab untuk membuat aplikasi basis data dengan menggunakan bahasa pemrograman yang ada, seperti: C++, Java, dan lainnya.
- *End Users*, yaitu siapapun yang berinteraksi dengan sistem secara online melalui *workstation*/terminal.
- *DA (Data Administrator)*, yaitu seseorang yang berwenang untuk membuat keputusan stategis dan kebijakan mengenai data yang ada.
- *DBA (DataBase Administrator)*, menyediakan dukungan teknis untuk implementasi keputusan tersebut, dan bertanggungjawab atas keseluruhan kontrol sistem pada level teknis.

Secara garis besar, *DBMS* merupakan sistem untuk membuat suatu basis data yang dapat dikontrol oleh pengguna. Komponen vital yang harus dimiliki dalam suatu *DBMS* meliputi perangkat keras, perangkat lunak, data, prosedur(aturan), dan orang untuk membuat, bekerja dan memelihara *DBMS*.

#### 2.1.2.1 Keuntungan *Database Management System* (*DBMS*)

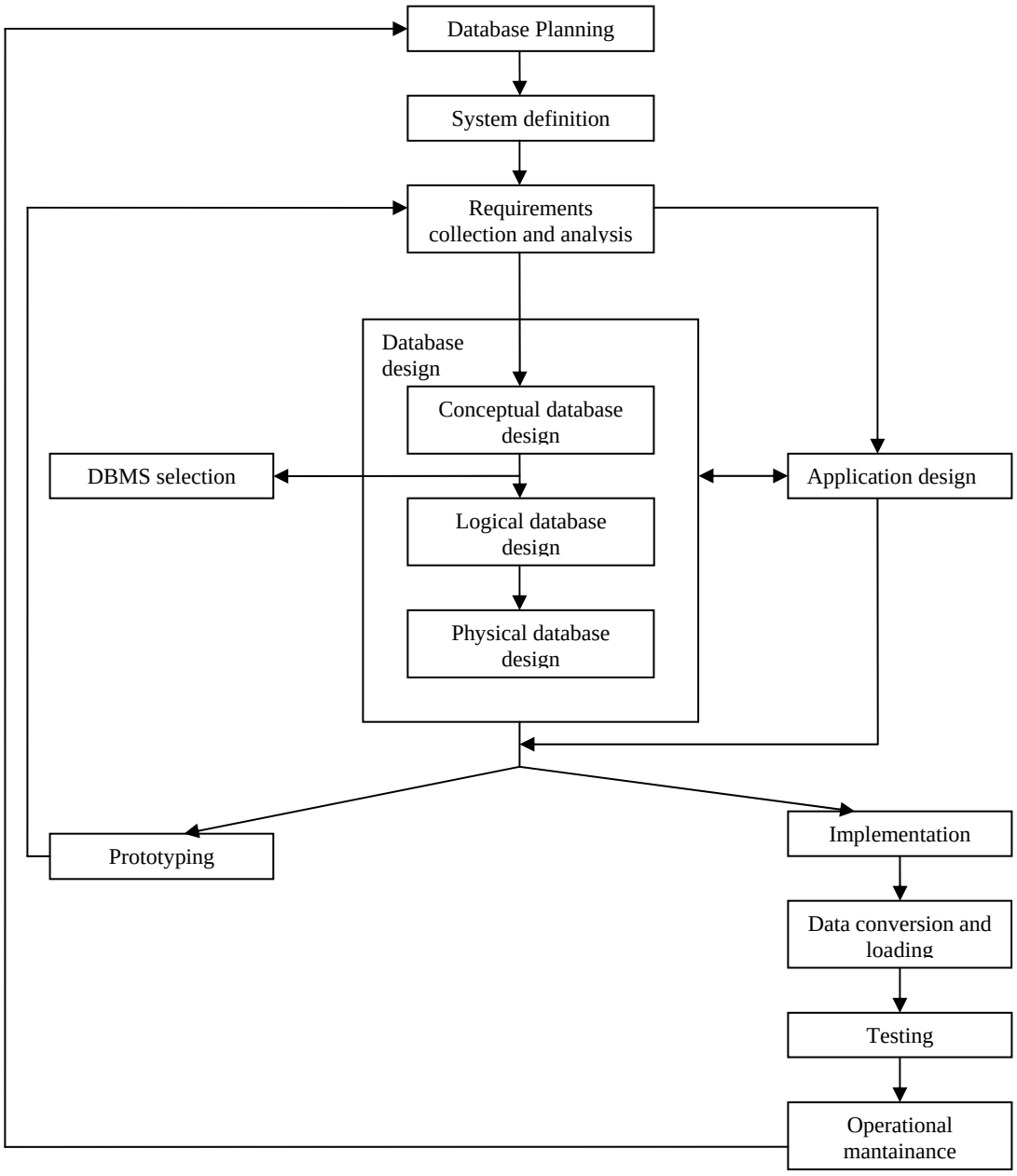
Berbagai macam keuntungan dapat diperoleh dari penggunaan *DBMS*. Dibawah ini adalah keuntungan-keuntungan dari *database management system (DBMS)* (Connoly, 2002, p26) :

- Mengontrol redudansi data
- Menghindari ketidakkonsistenan data
- Penggunaan data bersama
- Meningkatkan integritas data dan keamanan
- Kebutuhan *user* yang kompleks dapat diatasi
- Meningkatkan produktivitas
- Peningkatan layanan *back up* dan perbaikan

Secara garis besar, dengan adanya *DBMS* dapat meningkatkan keseluruhan sistem basis data dengan mengatur batasan di dalam sistem. Sehingga *DBMS* bisa digunakan oleh user yang berwenang untuk mendapatkan data secara cepat dan tepat.

## 2.2 Teori Perancangan *Database*

Langkah-langkah dalam merancang *database* terdapat dalam siklus aplikasi *database* (*Database Application Lifecycle*) seperti terlihat dalam gambar 2.1. *Database Application Lifecycle* menurut Connoly (2002, p271) merupakan komponen mendasar suatu sistem informasi, dimana pengembangan/pemakaiannya harus dilihat dari perspektif yang lebih luas berdasarkan kebutuhan organisasi.




Gambar 2.1 Siklus Aplikasi *Database*

(Sumber: Connoly, 2002, p271)

Secara garis besar, di dalam perancangan suatu *database* diperlukan suatu tahapan di dalam pengaplikasian suatu *database* agar mendapatkan suatu *database* yang sesuai kebutuhan. Sehingga dapat diterima oleh semua pihak yang bersangkutan.

### 2.2.1 Perencanaan *Database* (*Database planning*)

Merupakan aktivitas manajemen yang memungkinkan tahapan dari *database application lifecycle* direalisasikan seefektif dan seefisien mungkin. Perencanaan *database* harus terintegrasi dengan keseluruhan strategi sistem informasi dari organisasi. Terdapat 3 hal pokok yang berkaitan dengan strategi sistem informasi, yaitu:

- Identifikasi rencana dan sasaran (*goals*) dari perusahaan termasuk mengenai sistem informasi yang dibutuhkan.
- Evaluasi sistem informasi yang ada untuk menetapkan kelebihan dan kekurangan yang dimiliki.
- Penaksiran kesempatan teknologi informasi yang mungkin memberikan keuntungan kompetitif.

Secara garis besar, tahapan perencanaan database digunakan untuk mengetahui sasaran perusahaan, mengevaluasi sistem yang sudah ada, dan memperkirakan kemungkinan membuat sistem baru yang dapat digunakan secara efisien dan efektif. Di dalam tahapan ini, segala kemungkinan yang mungkin terjadi dalam pengembangan sistem baru harus diperhitungkan.

### 2.2.2 Definisi Sistem (*System definition*)

Menjelaskan batasan-batasan dan cakupan dari aplikasi database dan *user view* yang utama. Aplikasi *database* dapat memiliki satu atau lebih *user view*. Identifikasi *user view*, membantu memastikan bahwa tidak ada pengguna utama dari suatu *database* yang terlupakan ketika pembuatan aplikasi baru

yang dibutuhkan. *User view* juga membantu dalam pengembangan aplikasi *database* rumit/kompleks yang memungkinkan permintaan-permintaan dipecah kedalam bagian-bagian yang lebih mudah.

Secara garis besar, di dalam definisi sistem ditentukan batasan dan kemungkinan pemakai/pengguna database. Sehingga informasi untuk satu pemakai dapat berbeda dengan pemakai lain tergantung kebutuhan.

### 2.2.3 Analisis dan Pengumpulan Kebutuhan (*Requirements collection and analysis*)

Suatu proses pengumpulan dan analisa informasi mengenai bagian organisasi yang didukung oleh aplikasi *database*, dan menggunakan informasi tersebut untuk identifikasi kebutuhan pengguna akan sistem yang baru. Informasi didapat dengan metode penentuan fakta (*fact finding*). *Fact Finding* menurut Connoly (2002, p302) merupakan proses formal menggunakan teknik seperti wawancara dan kuisioner untuk mengumpulkan fakta tentang sistem, kebutuhan, dan keinginan. Lima metode *fact finding* yang dapat digunakan yaitu:

- Mempelajari dokumentasi (*Documentation Examination*), digunakan untuk mendeskripsikan masalah dan kebutuhan *database*, bagian perusahaan yang terpengaruh oleh masalah, dan sistem yang sedang berjalan dalam perusahaan.
- Wawancara (*Interview*), untuk mengetahui secara jelas masalah dalam perusahaan dengan bertanya langsung kepada pengguna sistem.

- Observasi (*Observing*), melakukan pengamatan secara langsung ke perusahaan yang bertujuan untuk memahami sistem yang berjalan dalam perusahaan.
- Penelitian (*Research*), mengumpulkan hasil penelitian dari internet, jurnal, buku yang berguna untuk memecahkan permasalahan.
- Kuisioner (*Questionnaires*), mengumpulkan informasi dari orang-orang dalam perusahaan dalam jumlah besar dan mempunyai respon yang berbeda-beda terhadap permasalahan.

Secara garis besar proses untuk mengumpulkan kebutuhan mengenai organisasi dapat dilakukan dengan lima cara yaitu mempelajari dokumen, wawancara, observasi, melakukan penelitian dan kuesioner

### 2.2.4 Desain *Database* (*Database Design*)

Merupakan suatu proses pembuatan sebuah desain *database* yang akan mendukung tujuan dan operasi suatu perusahaan. Tujuan utamanya adalah:

- Merepresentasikan data dan hubungan antar data yang dibutuhkan oleh seluruh area aplikasi utama dan sekelompok pengguna.
- Menyediakan model data yang mendukung segala transaksi yang diperlukan pada data.
- Menspesifikasikan desain minimal yang yang secara tepat disusun untuk memenuhi kebutuhan performa yang ditetapkan pada sistem (misal: waktu respon).

Pendekatan dalam desain database :

- *Top-down*

  Diawali dengan pembentukan model data yang berisi beberapa entitas *high-level* dan hubungan, yang kemudian menggunakan pendekatan *top-down* secara berturut-turut untuk mengindentifikasikan entitas *lower level*, hubungan dan atribut lainnya.

- *Bottom-up*

  Dimulai dari atribut dasar (yaitu sifat-sifat entitas dan hubungan), dengan analisis dari penggabungan antar atribut, yang dikelompokan kedalam suatu relasi yang merepresentasikan tipe dari entitas dan hubungan antar entitas.

- *Inside-out*

  Berhubungan dengan pendekatan *bottom-up* tetapi sedikit berbeda dengan identifikasi awal entitas utama dan kemudian menyebar ke entitas, hubungan, dan atribut terkait lainnya yang lebih dulu diidentifikasi.

- *Mixed*

  Menggunakan pendekatan *bottom-up* dan *top-down* untuk bagian yang berbeda sebelum pada akhirnya digabungkan.

Tiga fase desain database terdiri dari *: conceptual database design, logical database design* dan *physical database design*

Secara garis besar , pendekatan dalam desain database dilakukan dengan empat cara yaitu: *top-down, bottom-up, inside-out,* dan *mixed*. Untuk membuat

*database* diperlukan tiga tahapan yang meliputi *conceptual database design, logical database design* dan *phisical database design.*

### 2.2.5 Pemilihan *DBMS* (optional)

Pemilihan *DBMS* yang tepat untuk mendukung aplikasi *database*. Dapat dilakukan kapanpun sebelum menuju desain logikal asalkan terdapat cukup informasi mengenai kebutuhan sistem. Tahap-tahap utama untuk memilih *DBMS*:

- Mendefinisikan terminologi studi referensi
- Mendaftar dua atau tiga produk
- Evaluasi produk
- Rekomendasi pilihan dan laporan produk.

Secar garis besar, pemilihan DBMS untuk mendukung aplikasi *database* dilakukan sebelum design logical dengan cara: mendefinisikan terminologi, mendaftarkan beberapa produk, evaluasi produk, dan melakukan rekomendasi pilihan dan laporan.

### 2.2.6 Desain Aplikasi (*Application design*)

Desain antarmuka pemakai dan program aplikasi yang menggunakan dan memproses *database*. Desain *database* dan aplikasi merupakan aktifitas paralel yang meliputi dua aktifitas penting, yaitu :

- Desain Transaksi

  Transaksi adalah satu aksi atau serangkaian aksi yang dilakukan oleh pengguna tunggal atau program aplikasi, yang mengakses atau merubah

isi dari *database*. Kegunaan dari desain transaksi adalah untuk menetapkan dan keterangan karakteristik tingkat tinggi dari suatu transaksi yang dibutuhkan pada *database*, diantaranya:

- Data yang akan digunakan oleh transaksi
- Karakteristik fungsional dari suatu transaksi
- *Output* transaksi
- Keuntungannya bagi pengguna
- Tingkat kegunaan yang diharapkan

➢ Desain Antarmuka Pemakai

Beberapa aturan pokok dalam pembuatan antarmuka pemakai:

- *Meaningful title,* diusahakan pemberian nama suatu form cukup jelas menerangkan kegunaan dari suatu form/*report*.
- *Comprehensible instructions,* penggunaan terminologi yang familiar untuk menyampaikan instruksi ke pengguna dan jika informasi tambahan dibutuhkan, maka harus disediakan layar bantuan.
- *Logical grouping and sequencing of fields,* field yang saling berhubungan ditempatkan pada form/*report* yang sama. Urutan field harus logis dan konsisten.
- *Visually appealing layout of the form/report,* tampilan form/report harus menarik, dan sesuai dengan *hardcopy* agar konsisten.
- *Familiar field labels,* penggunaan label yang familiar.
- *Consistent terminology and abbreviation,* terminologi dan singkatan yang digunakan harus konsisten.

- *Consistent use of color*, penggunaan warna yang konsisten.
- *Visible space and boundaries for data-entry fields*, jumlah tempat yang disediakan untuk pemasukan data harus diketahui oleh pengguna.
- *Convinient cursor movement,* pengguna dapat dengan mudah menjalankan operasi yang diinginkan dengan menggerakkan kursor pada form/*report*.
- *Error correction for individual characters and entire fields,* pengguna dapat dengan mudah menjalankan operasi yang diinginkan dan melakukan perubahan terhadap nilai field.
- *Error messages for unacceptable values,* terdapat pesan kesalahan jika pengguna memasukkan data yang salah ke dalam suatu field.
- *Optional fields marked clearly,* field yang optional harus ditandai, biasanya diletakkan dibawah field yang harus diisi.
- *Explanatory messages for fields,* ketika pengguna meletakkan kursor pada suatu field, maka keterangan mengenai field tersebut harus dapat dilihat.
- *Completion signal,* indikator yang menjelaskan bahwa suatu proses telah selesai dilaksanakan.

Secara garis besar, *design* aplikasi meliputi dua aktivitas penting yaitu *design* transaksi dan *design antar muka pemakai*. *Design* transaksi berguna untuk menetapkan dan keterangan karakteristik tingkat tinggi

dari suatu transaksi yang dibutuhkan pada *database* dan *design antar muka pemakai*

### 2.2.7 *Prototyping* (optional)

Membuat model kerja suatu aplikasi *database*. Tujuan utama dari pembuatan *prototyping* adalah:

- Untuk mengidentifikasi fitur dari sistem yang berjalan dengan baik atau tidak.
- Untuk memberikan perbaikan-perbaikan atau penambahan fitur baru .
- Untuk klarifikasi kebutuhan user.
- Untuk evaluasi feasibilitas (kemungkinan yang akan terjadi) dari desain sistem khusus.

Secara garis besar, prototyping menghasilkan model kerja dari suatu aplikasi *database.* Bertujuan untuk mengidentifikasi fitur, memberikan perbaikan-perbaikan, klarifikasi kebutuhan user dan evaluasi feasibilitas

### 2.2.8 Implementasi (*Implementation*)

Merupakan realisasi fisik dari *database* dan desain aplikasi. Implementasi *database* dicapai dengan menggunakan :

- *Data Definition Language (DDL)* untuk membuat skema *database* dan file *database* kosong.
- *Data Definition Language (DDL)* untuk membuat *user view* yang diinginkan.

- 3GL dan 4GL untuk membuat program aplikasi. Termasuk transaksi *database* disertakan dengan menggunakan *Data Manipulation Language (DML)*, atau ditambahkan pada bahasa pemrograman.

Secara garis besar implementasi adalah realisasi fisik dari *database* dan desain aplikasi. Dilakukan dengan menggunakan DDL, 3GL, dan 4GL

### 2.2.9 Konversi Data dan Pemuatan (*Data Conversion and Loading*)

Pemindahan data yang ada ke dalam *database* baru dan mengkonversikan aplikasi yang ada agar dapat digunakan pada *database* yang baru. Tahapan ini dibutuhkan ketika sistem *database* baru menggantikan sistem yang lama. *DBMS* biasanya memiliki utilitas yang memanggil ulang file yang sudah ada kedalam *database* baru. Dapat juga mengkoversi dan menggunakan program aplikasi dari sistem yang lama untuk digunakan oleh sistem yang baru.

Secara garis besar, konversi data dan pemuatan merupakan proses pemindahan data ke dalam *database* baru dan mengkonversikan aplikasi yang sudah ada. Dibutuhkan ketika sistem *database* baru menggantikan sistem yang lama.

### 2.2.10 Pengetesan (*Testing*)

Suatu proses eksekusi program aplikasi dengan tujuan untuk menemukan kesalahan dengan mendemonstrasikan *database* dan program aplikasi untuk melihat apakah berjalan seperti yang diharapkan.

Dengan menggunakan strategi tes yang direncanakan dan data yang sesungguhnya. Pengujian hanya akan terlihat jika terjadi kesalahan pada perangkat lunak.

Secara garis besar, pengetesan bertujuan untuk menemukan kesalahan. Proses ini akan terlihat jika kesalahan terjadi pada perangkat lunak.

### 2.2.11 Pemeliharaan Operasional (*Operational maintenance*)

Suatu proses pengawasan dan pemeliharaan sistem setelah instalasi, meliputi:

- Pengawasan performa sistem, jika performa menurun maka memerlukan perbaikan atau pengaturan ulang database.
- Pemeliharaan dan pembaharuan aplikasi *database* (jika dibutuhkan).
- Penggabungan kebutuhan baru kedalam aplikasi *database*.

Secara garis besar**,** pemeliharaan sistem dilakukan setelah instalasi. Antara lain meliputi : pengawasan performa sistem, pemeliharaan dan pembaharuan aplikasi, dan penggabungan kebutuhan baru.

## 2.3 Keamanan dan Integritas *Database*

Keamanan *Database* menurut Connoly (2002, p522) adalah suatu mekanisme yang melindungi database terhadap serangan/ancaman yang disengaja atau tidak disengaja. Data merupakan sumber daya bernilai yang harus di atur dan diawasi secara ketat bersama dengan sumber daya perusahaan. Sebagian atau keseluruhan data perusahaan mempunyai kepentingan strategis dan karena itu harus dijaga agar tetap aman dan rahasia. Pertimbangan keamanan tidak hanya diaplikasikan pada

data yang ada dalam *database*. Pelanggaran terhadap keamanan dapat mempengaruhi bagian lain dari sistem, yang akan memberi timbal balik terhadap *database*.

Keamanan *database* terkait dengan keadaan berikut :

- Pencurian dan penipuan (*theft and fraud*)
- Kehilangan kerahasiaan (*loss of confidentiality*)
- Kehilangan keleluasaan pribadi (*loss of privacy*)
- Kehilangan integritas (*loss of integrity*)
- Kehilangan ketersediaan (*loss of availability*)

Ancaman (*threat*) menurut Connoly (2002, p523) adalah segala situasi atau kejadian, baik disengaja maupun tidak disengaja yang dapat menimbulkan efek merugikan terhadap sistem dan berikutnya organisasi. Tindakan pencegahan terhadap ancaman yang terjadi dapat dilakukan dengan:

- Otorisasi (*Authorization*)

  Pemberian hak atau wewenang, yang menyebabkan subjek memiliki legitimasi untuk mengakses sistem atau objek-objek dalam sistem.

- Pembuktian keaslian (*Authentication*)

  Suatu mekanisme yang menentukan apakah user yang mengakses benar-benar pengguna yang dimaksud, yaitu dengan menggunakan password.

- Sudut pandang (*View*)

  Merupakan hasil dinamis dari satu atau lebih operasi relasional yang dioperasikan pada relasi/tabel dasar untuk menghasilkan relasi/tabel lainnya. *View* merupakan relasi/tabel virtual yang tidak benar-benar ada dalam

*database*, tetapi dihasilkan berdasarkan permintaan oleh user tertentu pada saat tertentu.

- Salinan dan perbaikan (*Back Up and Recovery*)

  Suatu proses yang secara periodik mengambil salinan *database* dan log file (dapat juga berupa program) untuk disimpan pada media penyimpanan offline.

- Penjurnalan (*Journaling*)

  Suatu proses pemeliharaan dan penyimpanan log file (jurnal) dari semua perubahan yang dilakukan terhadap *database* untuk kemudahan perbaikan bila terjadi kerusakan.

Pembatasan integritas juga berkontribusi dalam menjaga keamanan sistem database dengan mencegah data dari ketidaksesuaian (*invalid*) dan mengakibatkan pemberian hasil yang salah. Batasan integritas dapat dilakukan dengan:

- Enkripsi (*Encryption*)

  Penyandian (*encoding*) data dengan menggunakan algoritma khusus yang membuat data tidak dapat dibaca oleh program manapun tanpa kunci dekripsi (*decryption*).

- Teknologi *RAID* (*Redundant Array of Independent Disks*)

  Perangkat keras dimana *DBMS* berjalan harus *fault-tolerant*, yang berarti bahwa *DBMS* harus tetap melanjutkan operasi walaupun terdapat satu komponen perangkat keras yang rusak. Komponen perangkat keras utama yang harus memiliki *fault-tolerant* meliputi *disk drives, disk controllers, CPU, power supplies, cooling fans*. *Disk drives* merupakan komponen yang paling

mudah diserang dengan waktu terpendek dibandingkan dengan kerusakan komponen *hardware* lainnya. Salah satu solusinya dengan menggunakan teknologi *RAID*, yaitu menyediakan serangkaian besar disk, yang terdiri dari susunan beberapa disk independen diatur untuk memperbaiki ketahanan (*reliability*) dan meningkatkan performa (*performance*), dengan cara:

- Performa (*performance*) meningkat melalui *data striping*, yaitu data dibagi menjadi beberapa bagian dengan ukuran yang sama (*striping units)*, yang secara jelas didistribusikan melewati beberapa disk.
- Ketahanan (*reliability*) diperbaiki melalui penyimpanan informasi berlebih melewati disk dengan menggunakan skema parity atau skema *error-correcting*, seperti *Reed-Solomon codes*.

Secara garis besar, keamanan *database* terkait dengan keadaan pencurian dan penipuan (*theft and fraud*), kehilangan kerahasiaan (*loss of confidentiality*), kehilangan keleluasaan pribadi (*loss of privacy*), kehilangan integritas (*loss of integrity*), kehilangan ketersediaan (*loss of availability*). Dapat dicegah dengan otorisasi, kebuktian keaslian, sudut pandang, dalinan dan perbaikna, dan penjurnalan.

### 2.3.1 Keamanan pada *SQL Server DBMS*

Untuk menjamin keamanan SQL Server menyediakan pernyataan GRANT dan REVOKE yang berdasarkan konsep:

- Identifier otoritas (*authorization identifier*), merupakan identifier yang digunakan untuk membuat identifikasi dari user. Biasanya memiliki password.

- Kepemilikan (*ownership*), *owner* adalah orang yang mengetahui tentang keberadaan objek dan operasi apa saja yang dapat dilakukan. Setiap objek yang dibuat dalam SQL memiliki *owner* yang dideklarasikan di awal pembuatan database di dalam pembuatan skema. Dengan bentuk:

  CREATE SCHEMA [Nama|AUTHORIZATION NamaPembuat]

  Contoh: CREATE SCHEMA SqlTest AUTHORIZATION Smith;

- Hak akses (*privileges*)

  Hak akses berdasarkan standar ISO adalah:

  - SELECT, hak untuk mengambil data dari tabel.
  - INSERT, hak untuk memasukkan baris baru ke dalam tabel.
  - UPDATE, hak untuk memodifikasi baris dari data dalam tabel.
  - DELETE, hak untuk menghapus baris dari data dalam tabel.
  - REFERENCES, hak untuk mereferensi kolom dari tabel yang disebutkan dalam batasan integritas.
  - USAGE, hak untuk menggunakan domain, perbandingan, set karakter dan terjemahan.

Pemberian hak akses ke pengguna lain dengan menggunakan pernyataan GRANT digunakan untuk memberikan kewenangan pada objek *database* ke pengguna tertentu. Bentuk format pernyataan GRANT:

GRANT {DaftarHak|ALL PRIVILEGES}

ON NamaObjek

TO {NamaUser|PUBLIC}

[WITH GRANT OPTION]

Contoh: GRANT SELECT,UPDATE (salary)

ON Staff

TO Personnel, Director;

Artinya: memberikan hak SELECT dan UPDATE pada kolom salary di tabel Staff pada Personnel dan Director.

Pencabutan hak akses dari user dengan menggunakan pernyataan REVOKE digunakan untuk mengambil kembali hak yang sebelumnya diberikan dengan pernyataan GRANT. Format pernyataan REVOKE:

REVOKE [GRANT OPTION FOR]{DaftarHak|ALL PRIVILEGES}

ON NamaObjek

FROM {NamaUser|PUBLIC}[RESTRICT|CASCADE]

Contoh: REVOKE SELECT

ON Branch

FROM PUBLIC;

Artinya: mencabut hak SELECT pada tabel Branch dari PUBLIC/semua pengguna.

Secara garis besar, untuk menjamin keamanan SQL server melalui indentifier otoritas, kepemilikan, dan hak akses. Keamanan ini menggunakan pernyataan GRANT dan REVOKE.

## 2.4 Metodologi Perancangan *Database*

Metodologi desain menurut Connoly (2002, p418) adalah pendekatan terstruktur yang menggunakan prosedur-prosedur, teknik-teknik, alat-alat,

dokumentasi tambahan untuk mendukung dan memberi fasilitas dari desain tersebut. Ada 3 fase utama dalam metodologi desain *database*, yaitu:

- *Conceptual Database Design*

  Merupakan proses pembuatan sebuah model dari informasi yang digunakan pada sebuah perusahaan, dimana tidak memiliki hubungan sama sekali dengan masalah fisik. Langkah-langkah dalam *conceptual database design* :

  **Langkah 1 : Membangun Model Data Konseptual Lokal Untuk Masing-Masing *View***

  Untuk membangun sebuah model data konseptual lokal dari sebuah perusahaan untuk setiap spesifik *view*. Setiap model data konseptual lokal terdiri dari:

  - Tipe entitas
  - Tipe hubungan
  - Atribut dan domain atribut
  - Primary key dan Alternate key
  - Batasan integritas

  Adapun langkah-langkahnya yaitu:

  1. **Identifikasi Tipe Entitas**

     Untuk mengidentifikasikan entitas utama yang dibutuhkan oleh *view*. Mendefinisikan objek utama dimana *user* mempunyai ketertarikan dengan objek tersebut. Objek-objek ini adalah tipe entitas untuk model. Salah satu metode untuk mengidentifikasi entitas adalah dengan menguji spesifikasi kebutuhan dari *user*. Dari spesifikasi ini kita mengidentifikasikan kata benda dan frase kata benda yang disebutkan. Kita juga dapat melihat objek

utama seperti orang, tempat atau konsep dari ketertarikan diluar kata benda lainnya yang merupakan kualitas dari objek lain.

2. **Identifikasi Tipe Hubungan**

   Tujuannya untuk mengidentifikasikan hubungan penting yang ada antara tipe entitas yang telah diidentifikasikan. Kita dapat menggunakan tata bahasa dari spesifikasi kebutuhan tersebut untuk mengidentifikasi hubungan, biasanya hubungan dinyatakan oleh kata kerja/*verb* atau ekspresi verbal. Secara langsung hubungan tersebut adalah *binary*, dengan kata lain hubungan tersebut berada antara dua tipe entitas. Kita pun harus berhati-hati untuk mencari hubungan yang kompleks yang dapat menghubungkan lebih dari dua tipe entitas. Langkah-langkah identifikasi tipe relasi:

   - Gunakan *Entity Relationship Diagram* (ERD).
   - Cari batasan dari tipe hubungan.
   - Periksa *fan* dan *chasm traps.*
   - Periksa bahwa masing-masing entitas ikut serta setidaknya dalam satu hubungan.
   - Dokumentasikan tipe hubungan.

3. **Identifikasi dan Hubungkan Atribut dengan Entitas atau Tipe Hubungan**

   Tujuannya untuk menghubungkan atribut dengan entitas atau tipe hubungan yang sesuai dan mendokumentasikan detail dari setiap atribut. Atribut-atribut bisa diidentifikasi dengan kata benda atau ungkapan kata

benda seperti properti, kualitas, identifier, atau karakteristik dari satu entitas atau hubungan. Atribut dapat dibagi menjadi 3 yaitu:

- Atribut *simple* atau *composite*
- Atribut *single* atau *multi value*
- Atribut *derived*

**4. Tetapkan domain atribut**

Tujuannya untuk menetapkan domain atribut dalam model data konseptual lokal dan mendokumentasikan setiap detail dari domain. Domain merupakan sekumpulan nilai-nilai dari satu atau lebih atribut yang menggambarkan nilainya. Model data yang dibuat menspesifikasikan domain untuk tiap-tiap atribut dan menyertakan :

- Nilai yang diizinkan untuk atribut
- Ukuran dan format atribut

**5. Tetapkan Atribut *Primary* dan *Candidate key***

Untuk mengidentifikasikan *candidate key* untuk setiap entitas dan jika terdapat lebih dari satu *candidate key*, maka pilih satu sebagai *primary key*.

**6. Mempertimbangkan Kegunaan dari konsep *Enhanced Modeling* (optional)**

Dalam langkah ini kita mempunyai pilihan untuk mengembangkan ER model dengan menggunakan konsep *enhanced modeling*, seperti spesialisasi, generalisasi, penggabungan (*aggregation*), komposisi (*composition*).

**7. Periksa Model Untuk Pengurangan**

Dalam langkah ini kita menguji model data konseptual lokal dengan tujuan spesifik untuk mengidentifikasikan apakah ada redundansi dalam data dan memindahkan data yang telah ada. Dua aktifitas dalam langkah ini adalah:

- Menguji ulang hubungan 1-1 (one-to-one)
- Menghilangkan hubungan yang redundan

**8. Validasi Model Konseptual Lokal Terhadap Transaksi User**

Tujuannya untuk memastikan model konseptual lokal mendukung transaksi yang dibutuhkan oleh *view*. Diuji dua pendekatan untuk memastikan model data konseptual lokal mendukung transaksi yang dibutuhkan, dengan cara:

- Mendeskripsikan transaksi-transaksi

  Memeriksa seluruh informasi (entitas, hubungan, dan atribut) yang dibutuhkan oleh setiap transaksi telah disediakan oleh model, dengan mendokumentasikan setiap kebutuhan transaksi.

- Mengunakan jalur-jalur transaksi

  Untuk validasi model data terhadap transaksi yang dibutuhkan termasuk representasi diagram jalur yang digunakan oleh setiap transaksi langsung pada ER diagram.

**9. Review Model Data Konseptual Lokal Dengan User**

Tujuannya untuk mereview model data konseptual lokal dengan *user* untuk memastikan model tersebut adalah representasi sebenarnya dari *view*. Model data konseptual ini termasuk ER diagram dan dokumentasi

pendukung yang mendeskripsikan model data. Bila ada kejanggalan (*anomali*) dalam model data, maka harus dibuat perubahan yang sesuai yang mungkin membutuhkan pengulangan langkah-langkah sebelumnya.

- ***Logical database design***

Suatu proses pembentukan model dari informasi yang digunakan dalam perusahaan berdasarkan model data yang spesifik (misal: relasional), tetapi tidak tergantung pada DBMS tertentu dan masalah fisik lainnya. Model data konseptual yang telah dibuat sebelumnya, diperbaiki dan dipetakan kedalam model data logikal.

Keseluruhan proses dari pengembangan model data logikal diuji dan disahkan pada kebutuhan pemakai. Teknik normalisasi digunakan untuk menguji kebenaran dari model data logikal, dimana normalisasi memastikan hubungan yang diperoleh dari model data tidak memperlihatkan kelebihan atau perulangan, yang bisa menyebabkan penyimpangan *update* ketika diimplementasikan. Model data logikal merupakan sumber informasi untuk tahapan selanjutnya yaitu design fisik basis data.

Aktivitas pada design logikal basis data terdiri dari dua langkah, dimana langkah pertama adalah membangun sebuah model data logikal lokal dari model data konseptual lokal yang menggambarkan pandangan tertentu dari perusahaan dan kemudian mengesahkan model ini untuk memastikan strukturnya telah benar. Sedangkan langkah kedua adalah membuat kombinasi

model data logikal lokal individual ke dalam sebuah model data logikal global tunggal yang menggambarkan perusahaan.

- ***Physical Database Design***

  Merupakan suatu proses pembuatan deskripsi dari implementasi Basis Data di penyimpanan secondary yang menjelaskan tentang hubungan utama, file organisasi dan indeks yang digunakan untuk memperoleh akses yang efisien ke data dan hubungan integritas constraint yang lainnya dan hal yang berkaitan dengan keamanan. Design fisik basis data merupakan tahap ketiga dan terakhir dari proses perancangan basis data. Di mana perancang memutuskan bagaimana basis data tersebut diimplementasikan. Akan tetapi dalam pengembangan design fisik basis data, hal pertama yang harus ditegaskan adalah sasaran dari sistem basis data. Oleh karena itu, design fisik disesuaikan untuk sebuah sistem DMBS khusus. Ini adalah alur balik antara design fisik dan logikal, karena keputusan yang diambil untuk meningkatkan dayaguna yang menggunakan struktur model data logikal.

  Tujuan utama model relasional ini adalah :

  - Membuat kumpulan tabel relasional dan batasan dari informasi yang didapat dalam model data logikal.
  - Mengidentifikasi struktur penyimpanan tertentu dan metode akses terhadap data untuk mencapai performa optimal dari sistem basis data.
  - Merancang proteksi keamanan sistem.

Secara garis besar, metodologi perancangan database meliputi tiga tahapan yaitu konseptual, logical dan fisikal. Konseptual dan logikal merupakan tahap pembuatan model *database* yang tidak tergantung pada masalah fisik dari perusahaan, sedangkan fisikal merupakan tahapan implementasi fisikal dari *database*