BAB 2

LANDASAN TEORI

2.1 Teori-teori Database

2.1.1 Pengertian Basisdata

Menurut Connolly & Begg (2002, p14) basis data adalah penggunaan bersama dari data yang berhubungan secara logis dan penjelasan dari data tersebut.

Menurut Date (2000, p4) basis data adalah suatu record terkomputerisasi yang mempunyai tujuan untuk menyediakan informasi pada saat dibutuhkan.

Menurut Mcleod (2001, p259) dua tujuan konsep basis data adalah meminimumkan pengulangan data dan mencapai independensi data. Pengulangan data adalah duplikasi data artinya data yang sama disimpan dalam beberapa file. Independensi data adalah kemampuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan pada program yang memproses data. Independensi data dicapai dengan menempatkan spesifikasi data dalam tabel dan kamus yang terpisah secara fisik dari program. Perubahan pada struktur data hanya dilakukan dalam sekali yaitu tabel.

2.1.2 Database Management System

Menurut Connolly & Begg (2002,p16) *Database Management System (DBMS)* adalah sistem perangkat lunak yang memampukan user untuk mendefinisikan, menciptakan, memelihara dan mengontrol akses ke basis data.

DBMS berinteraksi dengan program aplikasi pemakai dan basis data. *DBMS* menghasilkan fasilitas sebagai berikut :

- Mengijinkan pemakai untuk mendefinisikan basis data biasanya melalui DDL. DDL mengizinan user untuk menspesifikasi tipe data dan struktur serta batasan data yang dapat disimpan di basis data.
- Mengijinkan user (pemakai) untuk melakukan *insert*, *update*, *delete* dan *retrieve* data dari basis data biasanya melalui DML.
- Menghasilkan akses pengendalian basis data seperti :
    1. *Security system* untuk membatasi penggunaan basis data.
    2. *Integrity System* untuk menangani konsistensi penyimpanan data
    3. *Control concurrency system* untuk menangani penggunaan bersama basis data.
    4. *Control recovery system* untuk menyimpan kembali basis data saat terjadi kesalahan pada perangkat keras maupun perangkat lunak.
    5. *User accessible catalog* berisi deskripsi data di dalam basis data

Menurut Connolly & Begg (2002,pp18-20) ada 5 komponen DBMS yaitu:

1. Perangkat keras

   Dalam menjalankan aplikasi dari *DBMS* diperlukan perangkat keras.

   Contoh: single personal computer, single mainframe, jaringan computer berupa server.

2. Perangkat lunak

Perangkat lunak meliputi *DBMS* dan program aplikasi beserta sistem operasi. *DBMS* termasuk perangkat lunak jika DBMS tersebut digunakan dalam jaringan seperti LAN.

DBMS yang menggunakan jaringan contoh: C, C++, Visual Basic.

3. Data

   Data mungkin merupakan komponen terpenting dari *DBMS* khususnya dari sudut pandang pemakai akhir.

4. Prosedur

   Prosedur berupa panduan dan instruksi dalam mengarahkan desain dan penggunakan basis data. Pengguna dari sistem dan staf dalam mengelola basis data membutuhkan prosedur dalam menjalankan sistem dan mengelola basis data itu sendiri.

   Demikian prosedur di dalam basis data dapat berupa: *login* di dalam basis data, penggunaan sebagian fasilitas *DBMS*, cara menjalankan dan memberhentikan *DBMS*, membuat salinan *back up* basis data, memeriksa perangkat keras dan perangkat lunak yang sedang berjalan, mengubah struktur basis data, meningkatkan kinerja atau membuat arsip data pada *secondary storage*.

5. manusia

   Yang terlibat dalam sistem yaitu database administrator, perancang basis data, pengembang aplikasi, pemakai akhir.

Menurut Connolly&Begg (2002,pp48-52) fungsi *DBMS* sebagai berikut:

1. Data storage, retrieval, and update

DBMS harus melengkapi pengguna dengan kemampuan menyimpan, mengambil, dan meng*update* data dalam basis data.

2. User accessible catalog

   DBMS harus menyediakan sebuah katalog dimana deskripsi dari *data item* disimpan dan dapat diakses pengguna.

3. Transaction support

   DBMS harus menyediakan suatu mekanisme yang akan menjamin bahwa semua kegiatan *update* sesuai dengan transaksi yang dilakukan atau tidak sama sekali dilakukan.

4. Concurrency control services

   DBMS harus menyediakan mekanisme yang menjamin bahwa basis data di*update* dengan benar ketika lebih dari satu pemakai meng*update* basis data secara bersama.

5. Recovery service

   DBMS harus menyediakan mekanisme untuk memperbaiki basis data yang rusak karena suatu hal.

6. Authorization services

   DBMS harus menyediakan mekanisme untuk menjamin bahwa hanya pengguna yang mempunyai hak untuk mengakses basis data.

7. Support for data communication

   DBMS harus mampu berintegrasi dengan perangkat lunak komunikasi.

8. Integrity services

DBMS harus menyediakan cara untuk menjamin bahwa data dalam basis data dan perubahan pada data mengikuti aturan yang telah ditetapkan sebelumnya.

9. Services to promote database independence

   DBMS harus mencangkup fasilitas yang mendukung independensi program dari struktur actual basis data.

10. Utility service

    DBMS harus menyediakan satu set fasilitas layanan.

Menurut Petroutsos (2000, p5) DBMS menyediakan fungsi-fungsi sebagai berikut:

1. DBMS mengizinkan aplikasi mendefinisikan struktur dari basis data dengan pernyataan SQL. Pernyataan SQL yang mendefinisikan atau meng*edit* struktur ini disebut dengan *Data Definition Language (DDL).*
2. DBMS mengizinkan aplikasi memanipulasi informasi yang disimpan dalam basis data dengan pernyataan SQL. Pernyataan SQL yang memanipulasi informasi ini disebut dengan *Data Manipulation Language (DML).*

Menurut McLeod (2001, pp269-270) DBMS memungkinkan perusahaan maupun pemakai individu untuk:

- Mengurangi pengurangan data. Jumlah total file dikurangi dengan menghapus file-file duplikat. Juga hanya terdapat sedikit data yang sama di beberapa file.

- Mencapai independensi data. Spesifikasi data disimpan dalam skema daripada dalam tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.
- Mengintegrasikan data dari beberapa file ketika file dibentuk sehingga menyediakan kaitan logis, organisasi fisik tidak lagi menjadi kendala.
- Mengambil data dan informasi secara cepat. Hubungan-hubungan logis dan DML serta *query language* memungkinkan pemakai mengambil data dalam hitungan detik atau menit, yang sebelumnya mungkin memerlukan beberapa jam atau hari.
- Meningkatkan keamanan. Baik DBMS mainframe maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi atau *password, directory* pemakai, dan bahasa sandi atau *encryption.* Data yang dikelola oleh DBMS juga lebih aman daripada data lain dalam perusahaan.

Keputusan untuk menggunakan DBMS mengikat perusahaan atau pemakai untuk:

- Memperoleh perangkat lunak yang mahal. DBMS mainframe masih sangat mahal. DBMS berbasis komputer mikro, walau biayanya hanya beberapa ratus dolae, dapat menggambarkan pengeluaran yang besar bagi organisasi kecil.
- Memperoleh konfigurasi perangkat keras yang besar. DBMS sering memerlukan kapasitas penyimpanan primer dan sekunder yang lebih besar daripada yang diperlukan oleh program aplikasi lain. Juga, kemudahan

yang dibuat oleh DBMS dalam mengambil informasi mendorong lebih banyak terminal pemakai yang disertakan dalam konfigurasi daripada jika sebaliknya.

- Mempekerjakan dan mempertahankan sataf *database administrator*. DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemampuannya secara penuh pengetahuan khusus ini disediakan paling baik oleh para pengelola database atau *database administrator*.

2.1.3 Data Definition Language

Menurut Connolly&Begg (2002, p40) *Data Definition Language* (*DDL)* adalah bahasa yang mengizinkan database administrator atau pemakai untuk mendeskripsikan dan menamakan *entity*, *attribute*, dan *relationship* yang diperlukan oleh aplikasi, bersama dengan *associated integrity* dan *security constraint*.

Menurut Martina (2003, p58) *DDL* merupakan bagian dari sistem manajemen basis data yang dipakai untuk mendefinisikan dan mengatur semua attribute dan properti dari sebuah basis data. *DDL* digunakan untuk mendefinisikan basis data, tabel, dan *view*.

1. CREATE TABLE

   Pernyataan *create table* digunakan untuk mebuat tabel dengan mengidentifikasi tipe data untuk tiap kolom.

2. ALTER TABLE

   Pernyataan *alter table* dapat dipakai untuk menambah atau membuang kolom dan batasan.

3. DROP TABLE

   Pernyataan *drop table* dipakai untuk membuang atau menghapus tabel serta semua data yang terkait di dalamnya.

4. CREATE INDEX

   Pernyataan *create index* digunakan untuk membuat *index* pada tabel.

5. DROP INDEX

   Pernyataan drop index digunakan untuk membuang atau menghapus index yang telah dibuat sebelumnya.

2.1.4 Data Manipulation Language

Menurut Connolly&Begg (2002, p41) *Data Manipulation Language* (*DML)* adalah bahasa yang menghasilkan sekumpulan operasi untuk mendukung operasi manipulasi data pada data yang terdapat di basis data. Operasi manipulasi data meliputi:

- *Insertion*
- *Modification*
- *Retrieval*
- *Deletion*

Menurut Martina (2003, p60) *DML* digunakan untuk menampilkan, menambah, mengubah dan menghapus data di dalam obyek yang didefinisi *DDL*.

1. *Select*

Select dipakai untuk menampilkan sebagian atau seluruh isi dari suatu tabel dan menampilkan kombinasi isi dari beberapa tabel.

Rangkaian proses pada select:

FORM	menyatakan tabel yang digunakan.
WHERE	menyaring baris yang diinginkan untuk kondisi tertentu.
GROUP BY	mengelompokkan baris-baris dengan nilai kolom yang sama.
HAVING	menyaring grup dengan kondisi tertentu.
SELECT	menyatakan kolom mana yang akan menjadi tampilan *output.*
ORDER BY	menyatakan order untuk *input.*

2. *Update*

   Update digunakan untuk mengubah isi satu atau beberapa attribute dari suatu tabel.

3. *Insert*

   Insert digunakan untuk menambah satu atau beberapa baris nilai baru ke dalam suatu tabel.

4. *Delete*

   Delete digunakan untuk menghapus sebagian atau seluruh isi dari suatu tabel.

2.1.4.1 Entity Relationship Modeling

Menurut Connolly&Begg (2002, p331) *entity type* adalah sekelompok obyek-obyek dengan properti yang sama yang diidentifikasikan dengan keberadaan yang independent di perusahaan.

Menurut Connolly&Begg (2002, p332) konsep dasar dari *Entity Relationship Model* adalah *entity type* yang merepresentasikan grup dari obyek-obyek dalam dunia nyata dengan property sama. Sebuah *entity type* mempunyai keberadaan yang tidak terikat dan dapat menjadi obyek dengan keberadaan fisik ataupun konseptual. *Entity occurrence* adalah obyek dari sebuah *entity type* yang unik.

Menurut Connolly & Begg (2002, p334) *Relationship type* adalah sekumpulan hubungan antara satu atau lebih *entity type*. *Degree of relationship type* adalah jumlah dari partisipasi *entity type* dalam sebuah *relationship type* tertentu. *Entity* yang berkaitan dalam sebuah *relationship type* dikenal sebagai *participant* dalam *relationship* dan jumlah *participant* dalam *relationship* disebut sebagai *degree of relationship*. Oleh karena itu, *degree of relationship* menunjukkan jumlah dari *entity* yang terkait dalam *relationship*. Sebuah *relationship* berderajat dua disebut *binary*, sebuah *relationship* berderajat tiga disebut *ternary*, Sebuah *relationship* berderajat empat disebut *quartenar*.

Menurut Budiharto (2002, p4) perancangan basis data merupakan hal yang sangat penting dalam membuat basis data. Kita menggunakan *Entity Relationship* untuk menrancangnya. *Entity Relationship* merupakan sebuah gambaran untuk merancang basis data

yang baik. Karena tanpa *Entity Relationship* ini, bisa dipastikan proses pembuatan basis data berjalan lama dan tidak teratur. Pada saat merancang basis data yang perlu diperhatikan adalah membuat relasi-relasi yang benar di antara tabel. Proses perancangan basis data cukup memakan waktu yang lama jika basis datanya besar. Pendokumentasian rancangan basis data mutlak harus dilakukan dengan baik agar mudah di dalam pengembangan dan perbaikan nantinya.

Menurut Budiharto (2002, p5) *entity type* dapat didefinisikan sebagai sesuatu yang mudah diidentifikasikan. Konsep dari *Entity Relationship* yaitu *type entity* yang menampilkan kumpulan dari obyek di dalam kenyataan yang memiliki sifat atau properti yang sama.

Menurut Connolly & Begg (2002, pp338-342) *attribute* adalah sifat dari *entity* atau *relationship type*. *Attribute* menyimpan nilai dari setiap *entity occurrence* dan mewakili bagian utama dari data yang disimpan dalam bentuk basis data. *Domain attribute* adalah satuan nilai-nilai untuk satu atau beberapa attribute. Setiap *attribute* yang dihubungkan dengan sejumlah nilai disebut *domain*. *Domain* mendefinisikan nilai-nilai yang dimiliki sebuah *attribute* dan sama dengan konsep domain dalam model relasional.

Simple attribute adalah *attribute* yang terdiri dari satu komponen dengan keberadaan yang bebas. *Simple attribute* tidak bisa dibagi lagi ke dalam komponen yang lebih kecil, missal: *attribute* posisi dan gaji dari *entity* pegawai. *Simple attribute* disebut juga *atomic attribute*.

Composite attribute adalah *attribute* yang terdiri dari banyak komponen dengan keberadaan yang bebas. Dalam hal ini beberapa *attribute* dapat dipisahkan menjadi komponen yang lebih kecil lagi dengan keberadaan yang bebas. Contoh: *attribute* alamat dari *entity* kantor cabang yang mengandung nilai (jalan, kota, kode pos) bisa dipecahkan menjadi *simple attribute* jalan, kota, dan kode pos.

Single value attribute adalah *attribute* yang memiliki nilai tunggal untuk masing-masing *entity occurance*. *Multi value attribute* adalah *attribute* yang memiliki banyak nilai untuk masing-masing *entity occurance*.

Derived attribute adalah *attribute* yang menggantikan sebuah nilai yang diturunkan dari nilai sebuah attribute yang berhubungan, tidak perlu pada jenis *entity* yang sama.

Menurut Budiharto (2002, p9) *key* adalah suatu properti yang menentukan apakah suatu kolom pada tabel sangat penting atau tidak. Key terdiri dari *candidate key*, *primary key*, *alternate key*, dan *composite key*.

Candidate key adalah kunci yang unik untuk mengenali setiap kejadian di dalam *entity type*. Sebuah *candidate key* tidak boleh NULL. Sebuah *entity* mungkin punya lebih dari satu *candidate key*.

Primary key adalah *candidate key* yang dipilih sebagai kunci primer untuk mengenali secara unik sebuah atau setiap *occurrence* dari sebuah *entity type*. Pemilihan *primary key* untuk sebuah *entity* adalah

berdasarkan pertimbangan panjang *attribute*, jumlah minimal dari kebutuhan *attribute*, dan memenuhi syarat unik.

Candidate key yang tidak dipilih menjadi *primary key* disebut sebagai *alternate key*.

Composite key adalah *candidate key* yang terdiri dari dua *attribute* atau lebih. *Foreign key* adalah attribute pada satu relasi yang cocok pada *candidate key* dari beberapa relasi.

2.1.4.2 Strong and Weak Entity Type

Strong entity type adalah *entity type* yang keberadaannya tidak bergantung pada *entity type* yang lain. Karakteristiknya adalah setiap *entity occurence* secara unik mampu diidentifikasikan menggunakan *attribute primary key* pada *entity*nya.

Weak entity type adalah *entity type* yang bergantung pada keberadaan *entity* lainnya. Karakteristiknya adalah setiap *entity occurrence* tidak bisa diidentifikasi secara unik hanya dengan menggunakan *attribute* yang bergantung pada *entity*nya.

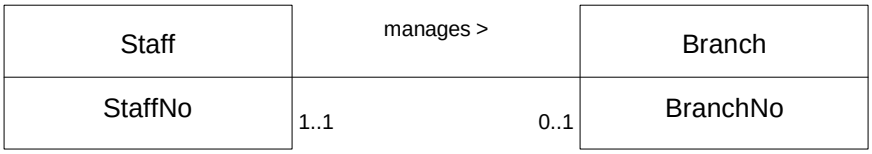
2.1.4.3 Structural constraints

Tipe utama dari batasan hubungan dalam relationship disebut *multiplicity*. *Multiplicity* adalah jumlah kemungkinan kejadian dari sebuah *entity* yang mungkin berhubungan ke sebuah kejadian tunggal dari sebuah *entity* yang tergabung melalui sebuah hubungan khusus.

Hubungan binary secara umum dibedakan menjadi:

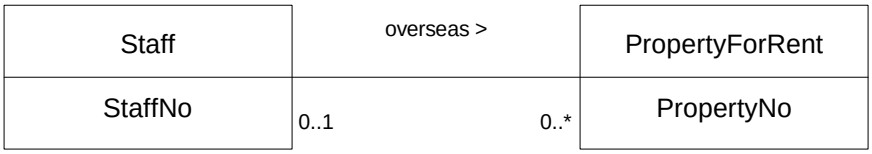
1. Derajat hubungan *one to one* (1:1)

   Terjadi bila setiap anggota *entity Staff* hanya boleh berpasangan dengan satu anggota dari *entity Branch*. Sebaliknya, tiap anggota dari *entity Branch* hanya boleh berpasangan dengan satu anggota *entity Staff*.



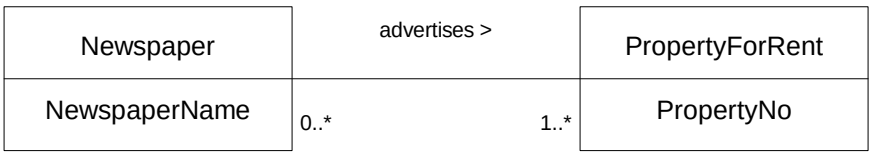
2. Derajat hubungan *one to many* (1:*)

   Terjadi bila setiap anggota *entity Staff* boleh berpasangan dengan lebih dari satu anggota *entity Property For Rent*. Sebaliknya, tiap anggota *entity Property For Rent* hanya boleh berpasangan dengan satu anggota *entity Staff*.



3. Derajat hubungan many to many (*:*)

   Terjadi bila tiap anggota *entity newspaper* boleh berpasangan dengan lebih dari satu anggota *entity Property For Rent*. Sebaliknya, tiap anggota *entity Property For Rent* juga boleh berpasangan dengan lebih dari satu anggota *entity Newspaper*.



2.1.5 Normalisasi

Menurut Connolly (2002, p379) normalisasi adalah teknik untuk mengorganisasi data ke dalam table-tabel untuk memenuhi kebutuhan pemakai didalam sebuah organisasi. Tujuannya adalah untuk menghilangkan kerangkapan data, mengurangi kompleksitas, dan modifikasi data.

Menurut Kroenko (Kadir, 2000, pp64-65) normalisasi adalah proses untuk mengubah suatu relasi yang memiliki masalah tertentu ke dalam dua buah relasi atau lebih yang tidak memiliki masalah tersebut. Masalah yang dimaksud itu sering disebut dengan istilah *anomaly*. *Anomaly* adalah efek samping yang tidak diharapkan (misalnya menyebabkan ketidak konsistenan data atau membuat data menjadi hilang saat data lain dihapus) yang muncul dalam suatu proses perancangan basis data.

Menurut Kadir(2000, p68-71) *dependency* merupakan konsep yang menjadi dasar normalisasi. *Dependency* menjelaskan nilai suatu *attribute* yang menentukan nilai *attribute* lainnya. *Dependency* ini kelak menjadi acuan bagi proses dekomposisi data ke dalam bentuk yang paling efisien.

Bentuk-bentuk dependency dapat dibagi sebagai berikut:

1. *Functional dependency*

   Suatu *attribute* Y memiliki ketergantungan fungsional terhadap *attribute* X jika dan hanya jika setiap nilai X berhubungan dengan sebuah nilai Y.

2. *Full functional dependency*

   Suatu *attribute* Y memiliki ketergantungan fungsional penuh terhadap *attribute* X jika:

   - Y memiliki ketergantungan fungsional terhadap X
   - Y tidak memiliki ketergantungan terhadap bagian dari X

3. *Total dependency*

   Suatu *attribute* Y memiliki ketergantungan total terhadap *attribute* X jika:

   - Y memiliki ketergantungan fungsional terhadap X
   - Y memiliki ketergantungan terhadap bagian dari X

4. *Transitive dependency*

   Suatu *attribute* Z memiliki ketergantungan transitif terhadap *attribute* X jika:

   - Y memiliki ketergantungan fungsional terhadap X
   - Z memiliki ketergantungan terhadap bagian dari Y

Menurut Kadir (2000, pp73-89) bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi dalam basis data dan harus dipenuhi untuk relasi-relasi tersebut pada tingkatan normalisasi.

Secara umum, normalisasi dibagi menjadi tingkatan, yaitu bentuk normal pertama (*1NF*) untuk penghilangan *repetition group*, normal kedua (*2NF*) berdasarkan *functional dependency* , normal ketiga (*3NF*) berdasarkan *transitive dependency*, *BCNF* (*Baycode-Codd Normal Form*) merupakan penguatan bentuk normal ketiga (*3NF*), *4NF* yang berdasarkan ketergantungan

multivalue dependency dan normal kelima (*5NF*) yang disebut juga *Project-Join Normal Form* (*PJNF*).

Menurut Connolly & Begg (2002, p387) *UNF* adalah sebuah table yang mnegandung lebih dari satu bagian yang berulang (*repeating group*). Bentuk *1NF* adalah hubungan dimana irisan (*intersection*) dari setiap baris dan kolom hanya mengandung satu nilai. Untuk mengubah table *UNF* ke *1NF* harus mengidentifikasi dan menghilangkan bagian yang berulang pada *UNF*, antara lain:

1. Pendekatan pertama, hilangkan *repetition group* dengan memasukkan data yang berlebihan ke dalam kolom dan baris kosong sehingga hasil dari table nantinya hanya mengandung nilai atomik (tunggal).
2. Pendekatan kedua, dengan menempatkan data yang berlebihan, selanjutnya dengan meng-*copy key attribute*nya yang asli dalam sebuah relasi yang dipisahkan.

Kedua pendekatan ini benar. Tetapi , pendekatan kedua awalnya menghasilkan relasi yang paling sedikit pada *1NF* dengan mengurangi redundancy. Jika menggunakan pendekatan pertama, relasi *1NF* adalah buruk. Selanjutnya selama langkah normalisasi berikutnya akan menghasilkan relasi yang sama yang dihasilkan oleh pendekatan kedua. Akan tetapi hasil dari normalisasi *1NF* masih bisa menyebabkan update anomalies, sehingga diperlukan normalisasi bentuk kedua (*2NF*).

Menurut Connolly, *2NF* adalah berdasarkan konsep ketergantungan fungsional penuh (*full function dependency*). *Full function dependency* dinyatakan dengan jika A dan B adalah attribute dari suatu relasi, B adalah

fungsional ketergantungan penuh (*fully functional dependency*) pada A jika B adalah secara fungsional bergantung pada A tetapi bukan merupakan himpunan bagian dari A. jelasnya bentuk *2NF* adalah sebuah relasi di dalam *1NF* dan setiap *attribute* yang bukan *primary key* adalah secara fungsional tergantung pada *primary key*. Proses normalisasi dari relasi *1NF* ke *2NF* melibatkan penghilangan dari bagian yang ketergantungan.

Menurut Connolly & Begg (2002, p394) *3NF* berdasar pada konsep peralihan ketergantungan (*transitive dependency*). *Transitive dependency* adalah sebuah kondisi dimana A, B, dan C adalah *attribute* dari sebuah relasi bahwa jika A -> B dan B->C, maka C secara transitif bergantung pada A melewati B (menyatakan bahwa A tidak secara fungsional bergantung pada B atau C). Pada *3NF*, sebuah relasi pada bentuk *1NF* dan *2NF* dan tidak ada *non primary key* secara transitif bergantung pada *primary key*. Proses normalisasi dari *2NF* ke *3NF* melibatkan penghilangan dari *transitive dependency*. Jika sebuah *transitive dependency* muncul, maka dihilangkan *transitive dependency* antara *attribute*nya dengan menempatkan *attribute* tersebut ke dalam relasi baru, selanjutnya dengan sebuah salinan dari determinannya.

2.1.6 4^{th} GL (Generation Language)

4^{th} GL meliputi :

1. *Presentation languages* seperti: *query languages* dan *report generators*.
2. *Speciality languages* seperti: *spreadsheets* dan *database languages*.
3. *Application generators* yang melakukan *define*, *insert*, *update* dan *retrieve* data dari basis data untuk membangun aplikasi.

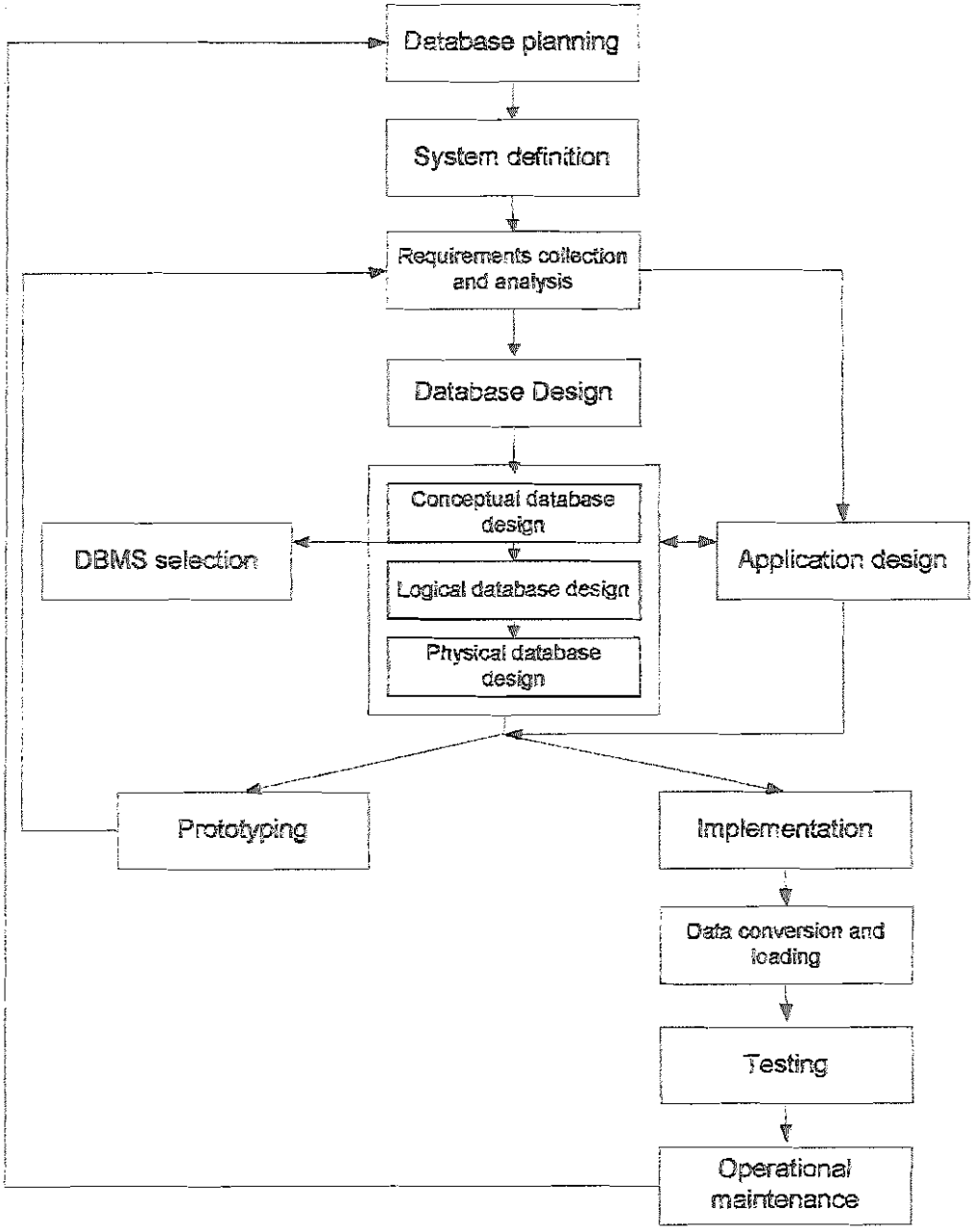
4. *Very high-level languages* yang digunakan untuk menurunkan kode aplikasi.

Contoh *4th GL* adalah SQL dan QBE.

2.1.7 Siklus Hidup Aplikasi Database

Database merupakan komponen mendasar suatu sistem informasi, dimana pengembangan/pemakaiannyaharus dilihat dari perspektif yang lebih luas berdasarkan kebutuhan organisasi.

Menurut Connolly & Begg (2002, p279) Tahapan Perancangan Database adalah proses membuat desain database yang akanmendukung operasi perusahaan. Menurut Connolly & Begg (2002, p272) Tahapan Database Application Lifecycle digambarkan sebagai berikut:



Gambar 2.1 Tahapan database Application Lifestyle

Berikut penjelasan dari gambar:

1. Perencanaan Database (*Database Planning*)

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 dari perusahaan termasuk mengenai sistem informasi yang dibutuhkan.
- Evaluasi sistem informasi yang ada untuk menetapkan kelebihan dan kekurangan yang dimiliki.
- Penaksiran kesempatan IT yang mungkin memberikan keuntungan kompetitif.

Metodologi untuk mengatasi hal tersebut diatas yaitu :

1. Database planning – mission statement:

   Mission statement untuk database project mendefinisikan tujuan utama dari aplikasi database. Mengarahkan database project, biasanya mendefinisikan perintah tugas (mission statement). Mission statement membantu menjelaskan kegunaan dari database project dan menyediakan alur yang lebih jelas untuk mencapai efektifitas dan efisiensi penciptaan dari suatu aplikasi database yang diinginkan.

2. Database planning – mission objectives:

   Ketika mission statement telah didefinisikan, maka mission objectives didefinisikan. Setiap objective (tujuan) harus mengidentifikasikan tugas khusus yang harus didukung oleh database. Dapat juga disertai dengan beberapa informasi tambahan yang

menspesifikasikan pekerjaan yang harus diselesaikan, sumberdaya yang digunakan dan biaya untuk membayar kesemuanya itu.

Database planning juga harus menyertakan pengembangan standar-standar yang menentukan :

- Bagaimana data akan dikumpulkan.
- Bagaimana menspesifikasi format/bentuk data
- Dokumentasi penting apakah yang akan diperlukan
- Bagaimana desain dan implementasi harus dilakukan

2. Definisi Sistem (*System Definition*)

Menjelaskan batasan-batasan dan cakupan dari aplikasi basis data dari sudut pandang user (user view) yang utama. User view mendefinisikan apa yang diwajibkandari suatu aplikasi basis data dari perspektif aturan kerja khusus (seperti Manajer atau Supervisor) atau area aplikasi enterprise (seperti Marketing, Personnel, atau Stock Control). Aplikasi basis data dapat memiliki satu atau lebih user view. Identifikasi user view, membantu memastikan bahwa tidak ada user utama dari suatu basis data yang terlupakan ketika pembuatan aplikasi baru yang dibutuhkan. User view juga membantu dalam pengembangan aplikasi basis data yang rumit memungkinkan permintaan-permintaan dipecah ke dalam bagian-bagian yang lebih sederhana.

3. *Requirement Collections and Analysis*

Suatu proses pengumpulan dan analisa informasi mengenai bagian organisasi yang didukung oleh aplikasi basis data, dan menggunakan

informasi tersebut untuk identifikasi kebutuhan pemakai akan sistem yang baru. Informasi dikumpulkan untuk setiap user view utama meliputi:

- Deskripsi data yang digunakan atau dihasilkan
- Detail mengenai bagaimana data dihasilkan
- Beberapa kebutuhan tambahan untuk aplikasi basis data yang baru

4. Desain Database (*Database Design*)

Database design merupakan suatu proses pembuatan sebuah desain basis data yang akan mendukung tujuan dan operasi suatu perusahaan. Tujuan utamanya adalah:

- Merepresentasikan data dan relationship antar data yang dibutuhkan oleh seluruharea aplikasi utama dan user group.
- Menyediakan model data yang mendukung segala transaksi yang diperlukan pada data.
- Menspesifikasikan desain minimal yang secara tepat disusun untuk memenuhi kebutuhan performa yang diterapkan pada sistem (misal: waktu respon).

5. Pemilihan *DBMS* (*DBMS Selection*)

Pemilihan *DBMS* dilakukan untuk memilih *DBMS* yang sesuai dengan aplikasi basis data. Langkah-langkah utama dalam memilih *DBMS* (Connolly, 2002, p284):

- menggambarkan cangkupan tugas berdasarkan kebutuhan perusahaan
- membuat perbandingan mengenal dua atau tiga produk *DBMS*
- mengevaluasi produk-produk *DBMS* tersebut

- merekomendasi pemilihan *DBMS* dan membuat laporan hasil dari evaluasi produk *DBMS* tersebut.

6. Desain Aplikasi (Application Design)

   Rancangan dari user interface dan program aplikasi yang digunakan dan memproses basis data (Connolly, 2002, p287).

7. *Prototyping*

   Pada kondisi tertentu kita dapat memilih apakah akan membuat *prototype* atau langsung mengimplementasikan aplikasi basis data. Suatu *prototype* adalah suatu model aplikasi basis data yang mempunyai semua corak yang diperlukan dan menyediakan semua kemampuan sistem. Tujuan utama *prototype* adalah mengizinkan pada pemakai untuk menggunakan *prototype* itu untuk menguji apakah fitur-fitur pada sistem telah bekerja sesuai dengan spesifikasi pengguna. Dengan cara ini, kita dapat memperjelas kebutuhan pemakai dan pengembang sistem dan mengevaluasi kelayakan desain sistem tersebut.

   Ada 2 cara strategi membuat *prototype* yaitu *requirements prototyping* dan *evolutionary prototyping* (Connolly, 2002, p291) untuk *requirements prototyping* digunakan *prototype* untuk menentukan kebutuhan suatu aplikasi basis data yang diusulkan dan ketika kebutuhan dirasakan sudah lengkap maka *prototype* tersebut tidak digunakan lagi. *Prototype* evolusioner digunakan untuk tujuan yang sama, perbedaannya adalah bahwa *prototype* tidaklah dibuang tetapi dikembangkan lebih lanjut sehingga aplikasi basis data tersebut.

8. Implementasi (*Implementation*)

Implementation merupakan perwujudan fisik dari basis data dan desain aplikasi (Connolly, 2002, p292) setelah menyelesaikan tahap desain (dengan atau tanpa *prototype*), kini kita berada pada tahap implementasi basis data dan program aplikasi. Implementasi basis data dicapai dengan menggunakan *DDL* dari *DBMS* yang telah dipilih atau dengan menggunakan *Graphical User Interface (GUI)*, masing-masing menyediakan fungsi keetika menyembunyikan pernyataan *DDL* yang *low-level*. Pernyataan *DDL* digunakan untuk menciptakan struktur basis data dan mengosongkan file yang terdapat dalam basis data tersebut. *User view* juga diterapkan pada langkah implementasi.

Program aplikasi diterapkan dengan menggunakan bahasa generasi ke-4 (*4GL*) atau generasi ke-3 (*3GL*) yang lebih disukai. Bagian dari program aplikasi ini adalah transaksi basis data yang diterapkan dengan menggunakan *DML*. Transaksi basis data juga dapat dibuat dalam bahasa pemograman seperti Visual Basic, Delphi, C, C++, Java, COBOL, Fortran, Ada, atau Pascal. Kita juga menerapkan komponen lain dari desain aplikasi seperti layer menu, format masukan data dan laporan.

Pengendalian keamanan dan integritas untuk aplikasi juga diimplementasikan. Sebagian dari kendali ini telah diterapkan dengan menggunakan *DDL* tetapi yang lain mungkin perlu untuk digambarkan di luar dari *DDL* sebagai contoh kegunaan yang disediakan *DBMS* atau kendali sistem operasi.

9. Konversi data dan Loading (Data Conversion and Loading)

Pemindahan data yang ada ke dalam basis data yang baru dan mengubah aplikasi yang sedang berjalan agar dapat digunakan dalam basis data (Connolly, 2002, p293)

10. Pengujian (Testing)

Testing adalah suatu proses melaksanakan program aplikasi dengan tujuan menemukan kesalahan (Connolly, 2002, p293). Dalam melakukan *testing* para pemakai sistem yang baru harus dilibatkan untuk menguji proses aplikasi dan basis data tersebut. Situasi yang ideal untuk pengujian sistem adalah mempunyai suatu pengujian basis data pada suatu sistem perangkat keras yang terpisah tetapi ini sering tidak tersedia. Jika data riil diharapkan untuk digunakan maka adalah penting untuk mempunyai *back-up*. Setelah pengujian diselesaikan, maka sistem aplikasi dan basis data ini telah siap untuk digunakan.

11. Pemeliharaan Operational (Operational Maintenance)

Setelah melalui tahap-tahap sebelumnya maka sistem sekarang telah pada tahap pemeliharaan yang melibatkan aktivitas berikut (Connolly, 2002, p293):

- mengawasi kinerja dari sistem. Jika performance jauh di bawah suatu tingkatan yang bisa diterima reorganisasi basis data mungkin diperlukan.
- Menjaga dan meningkatkan mutu aplikasi basis data (ketika diperlukan). Kebutuhan baru disatukan ke dalam aplikasi basis data

yang mengikuti langkah-langkah sebelumnya yang terdapat dalam siklus hidup basis data.

- Ketika aplikasi basis data sedang beroperasi, perlu dilakukan monitoring secara dekatuntuk memastikan bahwa *performance* dalam tingkatan yang bisa diterima.
- Monitoring proses akan terus berlanjut sepanjang seluruh hidup suatu aplikasi basis data tersebut dan pada waktu tertentu boleh melakukan reorganisasi basis data untuk mencukupi kebutuhan dari sistem. Perubahan ini menyediakan informasi pada evolusi sistem dan sumber daya yang pada masa yang akan dating mungkin diperlukan. Hal ini memungkinkan *database administrator* untuk terlibat dalam perencanaan kapasitas dan untuk memberitahu staf senior siaga untuk melakukan penyesuaian rencana jika *DBMS* kekurangan kegunaan tertentu, database administrator dapat mengembangkan kegunaan yang diperlukan atau membeli peralatan tambahan jika diperlukan

2.1.8 Design Konseptual, Logical, dan Fisikal Database

Menurut Connolly&Begg (2002, pp417-476) tahapan perancangan database adalah proses membuat suatu perancangan database yang dipakai untuk mendukung operasi dan tujuan perusahaan dibagi menjadi 3 tahapan yaitu:

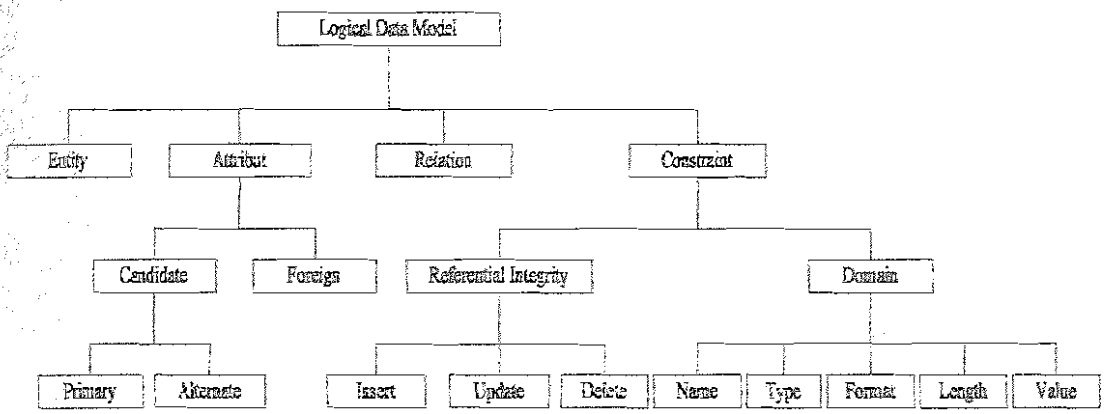
1. Konseptual Database Desain

Tahap ini merupakan proses pembuatan model informasi yang digunakan dalam sebuah perusahaan yang tidak tergantung pada semua masalah fisik. Awal tahap ini dimulai dengan pembuatan konseptual data model perusahaan yang secara keseluruhan bebas dari detail implementasi seperti DBMS (Database Management System) yang digunakan, program aplikasi, bahasa pemograman, platform untuk hardware, tingkat kinerja, maupun bentuk masalah fisik lainnya.

Perancangan ini terdiri dari 3 langkah:

- Menentukan entity pada database
- Mendefinisikan relationship antar entitas
- Menerjemahkan hubungan kedalam entity

Langkah-langkah diatas melibatkan komponen-komponen sebagaimana diperlihatkan pada gambar berikut:



Gambar 2.2 Model Data Logical

Penjelasan mengenai komponen-komponen diatas sebagai berikut:

1. *Entity*

Kumpulan objek-objek dengan properti yang sama yang keberadaannya independent / tidak bergantung dengan yang lain.

2. *Attribut*

   Properti dari sebuah entity atau tipe hubungan.

3. *Relation*

   Hubungan yang terjadi antara entity satu dengan entity lainnya.

4. *Constraint*

   Digunakan untuk melindungi integritas data yang terjadi pada saat pengisian data.

5. *Referential Integrity*

   Aturan-aturan yang mengatur hubungan antara primary key dan foreign key milik tabel-tabel yang berada dalam suatu database relatonal untuk menjaga konsistensi data.

   Berdasarkan operasi yang dilakukan, integritas referential dibagi menjadi 3 macam:

   - Insert
   - Update
   - Delete

   Tujuan *Referential Integrity* adalah menjamin agar elemen dalam suatu tabel yang menunjuk ke suatu pengenal unik pada suatu baris pada tabel lain benar-benar menunjuk ke suatu nilai yang memang ada.

Referential Integrity pada *Insert* contohnya: mahasiswa tidak dizinkan mendaftar mata kuliah yang tidak ditawarkan pada semester ini.

Referential Integrity pada *Update* memungkinkan pengubahan suatu key pada suatu tabel akan menyebabkan semua nilai pada tabel lain yang tergantung pada tabel yang bersangkutan juga akan berubah. Kemampuan ini disebut juga *cascade update*.

Referential Integrity pada *Delete* tidak memungkikan ada data yang dihapus. Contoh, data pelanggan pada tabel pelanggan tidak dapat dihapus jika ada salah satu data pelanggan yang dipakai pada tabel lain.

6. *Domain*

Himpunan nilai yang berlaku bagi suatu attribut. Batasan *domain* mendefinisikan nama/name, tipe, panjang/length, nilai data/value. Pada berbagai database akan sering ditemui tipe seperti Char dan Numeric. Dimana char menyatakan tipe alphanumeric/karakter yang dapat digabung dengan huruf, simbol, dan angka. Sedangkan numeric menyatakan tipe bilangan.

Menurut Yuswanto (2003, pp35-38) Tipe-tipe data yang ada dalam database adalah sebagai berikut:

- Integer
  - BigInt

    Integer merupakan bilangan bulat atau bilangan penuh. Adapun jangkauan yang dimiliki oleh tipe ini adalah dari -2^63 (-9223372036854775808) sampai 2^63-1 (9223372036854775807).

- Int

  Tipe bilangan ini termasuk bilangan bulat namun jangkauannya lebih kecil dibandingkan dengan BigInt. Adapun jangkauan yang dimiliki oleh tipe bilangan ini dari -2^31 (-2,147,483,648) sampai 2^31-1 (2,147,483,647).

- SmallInt

  Tipe bilangan ini termasuk ke dalam tipe bilangan bulat namun jangkauan yang dimilikinya lebih kecil dibandingkan dengan kedua tipe bilangan BigInt dan Int yaitu dari -2^31 (-2,147,483,648) sampai 2^31-1 (2,147,483,647).

- TinyInt

  Tipe bilangan yang termasuk ke dalam keluarga bilangan bulat ini mempunyai jangkauan dari 0 sampai 255.

- Bit

  - Bit

    Pada tipe bit ini data hanya berisi nilai 0 atau 1.

- Decimal dan Numeric

  - Decimal

    Ketepatan yang dimiliki oleh tipe bilangan decimal ini dari 10^38+1 sampai 10^38-1

  - Numeric

Tipe bilangan numeric secara fungsional memiliki ketepatan sama dengan ketepatan yang dimiliki oleh tipe decimal yaitu dari 10^38_1 sampai 10^38-1

- Money dan SmallMoney
  - Money

    Tipe data money memiliki jangkauan dari -2^63(-922,337,203,685,477.5808) sampai 2^63-1 (+922,337,203,685,477.5807), dengan ketelitian yang cukup baik.
  - SmallMoney

    Tipe data money memiliki jangkauan dari -214,748.3648 melalui/ sampai +214,748.3647 lebih kecil dibandingkandengan tipe data money, namun masih memiliki ketelitian yang cukup tinggi.
- Float dan Real
  - Float

    Tipe data ini memiliki ketepatan nilai dari -1.79E+308 sampai 1.79E+308.
  - Real

    Tipe data ini memiliki ketepatan nilai dari -3.40E+38 melalui/ sampai 3.40E+38.
- Datetime dan Smalldatetime
  - Datetime

    Tipe data ini berupa data waktu dan tanggal dari Januari 1, 1753 sampai Desember 31, 9999.

- Smalldatetime

  Tipe data ini berupa data waktu dan tanggal dari Januari 1, 1900 sampai Juni 6, 2079 dengan suatu ketelitian satu menit.

- String
  - Char

    Tipe data char merupakan fixed-length non unicode yang mempunyai jangkauan maksimum sebesar 8000 karakter.

  - Varchar

    Tipe data char merupakan length non unicode yang mempunyai jangkauan maksimum sbesar 8000 karakter

  - Text

    Tipe data char merupakan variabel length non unicode dengan panjang maksimum 2^31-1 (2,147,483,647) karakter.

- Unicode String
  - nChar

    Tipe data nChar merupakan fixed-length unicode yang mempunyai jangkauan maksimum sebesar 4000 karakter.

  - nVarchar

    Tipe data nVarchar merupakan length nonunicode yang mempunyai jangkauan maksimum sebesar 4000 karakter

  - nText

    Tipe data nText merupakan variabel length non unicode dengan panjang maksimum 2^30-1 (1,073,741,483,647) karakter.

- Binary String
  - Binary

    Binary merupakan tipe data dengan jenis fixed-length dengan panjang maksimum sebesar 8000 bytes.
  - Varbinary

    Binaty merupakan tipe data dengan jenis variabel-length dengan panjang maksimum sebesar 8000 bytes.
  - Image

    Image merupakan tipe data dengan jenis variabel-length dengan panjang maksimum 2^31-1 (2,147,483,647) bytes.

2. Logical Database Desain

   Tahap ini merupakan pembuatan model informasi yang digunakan perusahaan berdasarkan pada model data khusus, tetapi bebas dari DBMS (Database Management System) tertentu dan masalah fisik lainnya. Tahap ini memetakan model konseptual pada model logical yang dipengaruhi oleh data model untuk database tujuan. Model data logical merupakan sumber informasi untuk tahap perancangan fisikal dan menyediakan suatu kendaraan bagi perancang fisikal database untuk melakukan pertukaran yang sangat penting bagi perancangan database yang benar.

3. Fisikal database Desain

   Pada tahap ini merupakan pembuatan deskripsi dari implementasi database pada secondary storage (penyimpanan sekunder) yang menjelaskan dasar relasi, organisasi file, dan indeks yang digunakan untuk

memperoleh akses data yang benar dan masalah integritas lain yang berkaitan dan menentukan mekanisme security (keamanan data). Tahap ini memungkinkan perancang untuk menentukan bagaimana database diimplementasikan.

Oleh karena itu, rancangan fisikal dirancang untuk *DBMS* yang khusus. Antara rancangan logical dan fisikal terdapat keterkaitan, hal ini disebabkan karena keputusan yang diambil selama perancangan fisikal untuk meningkatkan kinerja bisa mempengaruhi logikal data model.

Kegiatan-kegiatan yang dilakukan antara lain:

- Mendefinisi seluruh kolom untuk semua tabel
- Mendefinisi kebutuhan *referential integrity*
- Mendefinisi *view*
- Mendefinisi indeks

2.2 Teori-Teori Lainnya

2.2.1 Teori Penjualan

Penjualan atau sebuah aktivitas dari jual adalah bagian dari bentuk komersil sebagai salah satu bentuk implementasi dari marketing, penjualan, yang mempunyai bagian-bagian seperti struktur organisasi, memperkerjakan seorang spesialis disebut, sales (penjual).

Penjualan merupakan fungsi yang dikhususkan pada upaya-upaya membangun hubungan pembelian dengan pelanggan. Penjualan adalah yang mendatangkan pelanggan untuk membeli pada perusahaan kita.

Dari sudut pandang marketing, penjualan adalah salah satu metode promosi.

Bentuk-bentuk penjualan sebagai berikut:

- Direct Sales Involving face-to-face contact
- Retail or Consumer
- Door-to-Door or Travelling
- Indirect - human-mediated but with direct contact
- Telephone or Telesales
- Mail Order
- Electronic
- Web Business-to-Business (B2B), Business-to-Customer (B2C)
- Electronic Data Interchange (EDI)
- Consignment
- Multi-level
- Sales Agents (real estate, manufacturing)

2.2.2 Teori Pembelian

Pembelian (purchasing) adalah salah satu fungsi dasar yang umum ada di semua jenis perusahaan.

Dikatakan fungsi dasar karena perusahaan tidak dapat beroperasi dengan baik tanpanya. Dari sifatnya, pembelian adalah bagian dasar dan integral dari manajemen bisnis.

Pembelian Angsuran adalah pembelian barang yang pelunasan secara berangsur, yang penyerahan dan perpindahan hak milik dilakukan setelah pelunasan angsuran terakhir.

2.2.3 Teori Persediaan

Persediaan adalah jumlah barang yang akan dapat diproduksi. Persediaan juga merupakan :

- suatu jumlah barang yang tersedia dan dapat digunakan
- menawarkan barang dan jasa untuk penjualan
- Suatu kegiatan menyediakan dan menyimpan sesuatu

Persediaan merupakan sejumlah produk yang akan dijual, dengan harga yang sudah ditentukan. Contoh: sebuah pabrik chip ingin memproduksi 1 juta paket chip dengan harga $1, dengan harga pasaran $2. Hal utama yang mendasari jumlah produksi chip adalah keinginan memproduksi akan menjadi dasar harga market barang dan biaya untuk memproduksinya.