BAB 2

LANDASAN TEORI

2.1 Teori *Database* dan *Modelling*

2.1.1 *Database*

Menurut C.J.Date (1995,p9) *Database* adalah suatu kumpulan dari data yang bersifat *persistent*, yaitu data yang berbeda dengan lainnya dan biasanya merupakan data yang bersifat sementara, dimana kumpulan data tersebut dapat digunakan oleh sistem – sistem aplikasi perusahaan.

Menurut McLeod (1995,p324) *Database* adalah suatu kumpulan data komputer yang terintegritasi, diorganisasikan dan disimpan dengan suatu cara yang memudahkan pengambilan data kembali (*retrieval*).

Menurut Connoly (2002, p14), *database* adalah suatu kumpulan data atau *file* yang dapat digunakan bersama yang secara logika terkait atau saling berhubungan satu sama lain, dan uraian tentang data tersebut dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi. *Database* disimpan di dalam media penyimpanan di luar memori komputer, seperti dalam *hard disk*, disket, dll. Jadi *Database* adalah kumpulan data yang bersifat *persistent*, terintegrasi, terkait, saling berhubungan dan disimpan dalam satu atau beberapa media

penyimpan (*storage device)* yang dapat digunakan bersama-sama secara logika.

2.1.2 *Database Management System*

Menurut Ramakrishnan (2000, p3), *Database Management System* atau yang biasa disingkat DBMS adalah perangkat lunak yang dirancang untuk membantu dalam pemeliharaan dan pemanfaatan sejumlah besar kumpulan data, dan kebutuhan bagi system tersebut, seperti kegunaan mereka, berkembang dengan cepat.

Fungsi – fungsi DBMS menurut Codd (1982) meliputi 8 pelayanan yang harus benar – benar dimiliki oleh DBMS dan diharapkan dapat ditambahkan lagi dua pelayanan yang memungkinkan.

1. Penyimpanan data, perolehan kembali, dan memperbaharui suatu DBMS harus dilengkapi para pemakai dengan kemampuan untuk menyimpan, mendapat kembali, dan memperbaharui data di dalam *database*.
2. Sebuah katalog yang dapat diakses; suatu DBMS harus dilengkapi suatu katalog di mana uraian data item disimpan dan dapat diakses oleh para pemakai. Katalog;menjadi dapat diakses ke para pemakai seperti halnya DBMS itu. Suatu sistem katalog, atau kamus data, adalah suatu tempat penyimpanan informasi yang menggambarkan data itu di dalam *database*; dimana hal itu adalah data tentang data atau meta-data

3. Transaksi pendukung suatu DBMS harus dilengkapi suatu mekanisme yang akan memastikan yang manapun bahwa semua update sesuai dengan transaksi ditentukan
4. Jasa Pengendalian *Concurrency* adalah suatu DBMS harus dilengkapi suatu mekanisme untuk memastikan bahwa *database* itu dipdate dengan tepat ketika para pemakai sedang memakai / mengupdate *database* itu secara bersamaan.
5. *Recovery services*; suatu DBMS harus dilengkapi suatu mekanisme untuk mengembalikan *database* seandainya *database* menjadi rusak bagaimanapun juga.
6. Jasa Otorisasi; suatu DBMS harus dilengkapi suatu mekanisme untuk memastikan bahwa hanya para pemakai yang diberi hak dapat mengakses *database* itu.
7. Pendukung komunikasi data, suatu DBMS harus mampu untuk mengintegrasikan dengan perangkat lunak komunikasi. Kebanyakan para pemakai mengakses *database* itu dari stasiun-kerja. Terkadang stasiun-kerja ini dihubungkan secara langsung ke komputer yang menjadi *hosting* DBMS.
8. Jasa Integritas; suatu DBMS harus melengkapi sebuah makna untuk memastikan bahwa kedua data ada di dalam *database* dan yang berubah ke data mengikuti aturan tertentu. *Database* intregrasi mengacu pada ketepatan dan konsistensi dari data disimpan; hal itu dapat diperlakukan sebagai perlindungan *database*.

9. Jasa untuk mempromosikan ketidaktergantungan data; suatu DBMS harus meliputi fasilitas untuk mendukung ketidaktergantungan program dari struktur yang nyata *database* itu.
10. *Utility services*; suatu DBMS perlu menyediakan seperangkat *utility services*. Utility program membantu DBA untuk mengurus *database* secara efektif.

Menurut Ramakrishnan (2000, p8), keuntungan yang diperoleh dari penggunaan DBMS untuk mengolah data antara lain adalah :

- *Data independence*; program aplikasi harus sebebas mungkin dari rincian representasi dan penyimpanan data. DBMS mampu menampilkan tampilan abstrak dari data untuk mengisolasi kode aplikasi dari rincian tersebut.
- Akses data yang efisien; DBMS mengutilisasi bermacam teknik yang menipu untuk menyimpan dan memakai kembali data secara efisien.fitur ini sangat penting jika data disimpan pada alat penyimpanan eksternal.
- *Data integrity* dan *security*; jika data selalu diakses melalui DBMS, maka DBMS dapat melakukan *integrity constraint* pada data. DBMS juga dapat melakukan kontrol akses yang mengatur data apa yang dapat dilihat pada kelas user yang berbeda.
- *Data administration*; saat beberapa *user share* data, memusatkan administrasi data dapat memungkinkan perbaikan yang signifikan.
- Akses yang bersamaan dan perbaikan kerusakan; DBMS menjadwalkan akses yang berbarengan ke data dalam cara dimana user menganggap

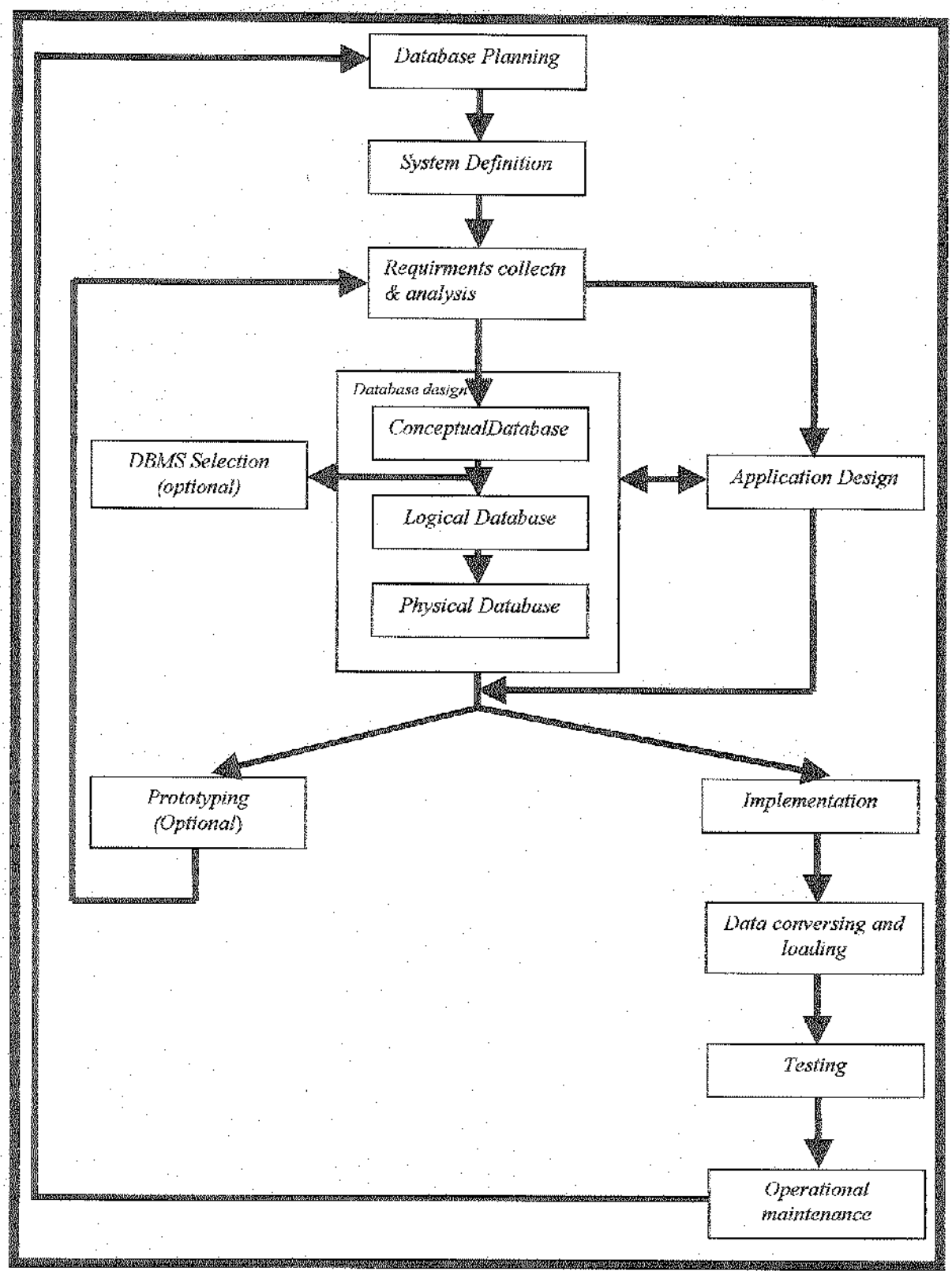
bahwa data hanya diakses oleh satu user pada satu saat. Selain itu, DBMS melindungi *user* dari akibat kegagalan sistem.

- Mengurangi waktu pengembangan aplikasi; DBMS mendukung banyak fungsi yang penting yang umum bagi aplikasi yang mengakses data tersimpan dalam DBMS. Hal ini besamaan dengan *high-level interface* ke data, menfasilitasi perkembangan aplikasi yang cepat.

2.2 Teori *System Database Lifecycle*

Menurut Connoly (2002, p271), sistem *database* merupakan komponen yang fundamental dalam sistem informasi luas pada organisasi yang besar. Karena itu, *database application lifecycle* sudah sewajarnya berhubungan dengan *lifecycle* dari sistem informasi organisasi tersebut.

Berikut tahapan dalam *database application lifecycle*



Gambar 2.1 diagram SDLC
(Sumber : Connoly, *"Database System"*, 2002, p272)

2.2.1 *Database planning*

Menurut Connoly (2002, p273), *database planning* adalah kegiatan manajemen yang memungkinkan tahapan-tahapan dalam *database apliaction* untuk direalisasikan seefisien dan seefektif mungkin. *Database planning* harus diintegrasikan dengan keseluruhan strategi sistem informasi organisasi. Terdapat tiga hal utama yang terlibat dalam merumuskan strategi sistem informasi, yaitu :

- Identifikasi rencana dan tujuan perusahaan dengan keputusan selanjutnya dari kebutuhan sistem informasi.
- Evaluasi dari sistem informasi terakhir untuk menentukan kekuatan dan kelemahan yang ada.
- Penaksiran kesempatan IT yang mungkin menghasilkan keuntungan yang kompetitif.

2.2.2 *System definition*

Menurut Connoly (2002, p274), *System definition* menggambarkan jangkauan dan batasan dari aplikasi *database* dan tampilan *user* yang utama. Sedangkan tampilan *user* sendiri berarti apa yang diminta dari aplikasi *database* dari sudut pandang pekerjaan tertentu (seperti manjer, *supervisor*) atau area aplikasi perusahaan (seperti *marketing*, personalian, dll). Sebuah aplikasi *database* dapat mempunyai lebih dari satu tampilan *user*.

2.2.3 *Requirements collection and analysis*

Menurut Connoly (2002, p276), *Requirements collection and analysis* adalah proses mengumpulkan dan menganalisis informasi mengenai bagian dari organisasi yang didukung oleh aplikasi *database*, dan menggunakan informasi ini untuk mengidentifikasi permintaan *user* dari sistem baru.

Tahap ini terdiri dari kumpulan dan analisis dari informasi mengenai bagian dari perusahaan yang didukung oleh *database*. Ada banyak teknik untuk mengumpulkan informasi tersebut, yang disebut dengan *fact finding techniques.*

Informasi dikumpulkan untuk tiap tampilan *user* utama termasuk :

- Sebuah deskripsi dari data yang digunakan atau dibangkitkan
- Detil dari bagaimana data digunakan atau dibangkitkan
- Permintaan tambahan untuk aplikasi *database* baru

2.2.4 *Database design*

Menurut Connoly (2002, p279), *database* ***design*** adalah proses membuat desain untuk *database* yang akan mendukung operasi dan tujuan perusahaan. Menurut Connoly (2002, p281), fase dari *database* ***design*** antara lain :

2.2.4.1 Desain *database* konseptual

Yaitu proses konstruksi model informasi yang dipakai dalam sebuah perusahaan, bebas dari semua pertimbangan fisikal. Tahap ini merupakan tahap pertama dari desain

database, dan termasuk menciptakan model data konseptual. Model data dibuat dengan menggunakan informasi yang didokumentasikan dalam spesifikasi permintaan *user*. Tahap ini terbebas dari detil implementasi seperti *software* DBMS target, program aplikasi, bahasa pemrograman, *platform hardware*, dan lain-lain.

2.2.4.2 **Desain *database* logikal**

Yaitu proses konstruksi model informasi yang dipakai dalam sebuah perusahaan berdasarkan pada model data tertentu, tapi terbebas dari DBMS tertentu dan pertimbangan fisikal lainnya. Tahap ini merupakan tahap kedua dari desain *database*. Model data konseptual yang telah dihasilkan dari tahap sebelumnya diperbaiki dan dipetakan kepada model data logikal. Model data ini berdasarkan pada model data target untuk *database*. Di mana *conceptual* data *model* independen dari semua pertimbangan fisik, *logical model* diturunkan dari pengetahuan akan model data dasar dari DBMS yang akan digunakan. Apakah DBMS itu *relational*, *network*, *hierarchical*, atau *object-oriented*. Tetapi aspek lain dari DBMS tidak diperhatikan.

Sepanjang proses pengembangan model data logikal, model diuji dan divalidasi terhadap permintaan *user*. Teknik

normalisasi digunakan untuk menguji kebenaran model data logikal. Normalisasi memastikan bahwa relasi yang diperoleh dari model data tidak menampilkan redundansi data, yang dapat menyebabkan *update anomalies* saat diimplementasikan.

2.2.4.3 Desain *database* fisikal

Yaitu proses memproduksi suatu uraian implementasi *database* pada media penyimpanan sekunder; menguraikan hubungan data, mengorganisasikan *file*, dan indeks yang digunakan untuk mencapai efisiensi pada data, dan semua integritas hubungan dan ukuran keamanan.

Physical database design adalah tahap ketiga dan akhir dari proses desain *database*, di mana desainer memutuskan bagaimana *database* diimplementasikan. Dalam mengembangkan *physical database design*, harus ditentukan lebih dulu sistem *database* yang akan digunakan. Sehingga desain *physical* disesuaikan pada sistem DBMS tertentu. Tujuan utama dari *Physical database design* adalah menggambarkan bagaimana *Logical design* diterapkan secara fisik. Untuk model *relational*, hal ini terdiri dari :

- untuk mencapai performa yang optimal bagi sistem *database*.

  Membuat sekumpulan *relational table* dan *constraint* pada

tabel-tabel ini dari informasi yang ditampilkan dalam *logical* data *model*.

- Menentukan struktur penyimpanan yang akan dipakai dan metode akses data
- Mendesain proteksi sekuriti bagi sistem.

2.2.5 *DBMS selection*

Menurut Connoly (2002, p284), ***DBMS selection*** adalah seleksi DBMS yang sesuai untuk mendukung aplikasi *database*. Langkah-langkah utama dalam menyeleksi sebuah DBMS adalah :

- Menggambarkan cakupan tugas studi , menyatakan sasaran hasil dan raung lingkup studi, dan tugas yang perlu untuk dikerjakan.
- Shortlist dua atau tiga produk DBMS yang akan kita gunakan.
- Mengevaluasi produk, bisa dari segi accessibility, data definition, physical definition, dan lain – lain.
- Recommend pemilihan dan laporan hasil

2.2.6 *Application design*

Menurut Connoly (2002, p287), ***application design*** adalah desain/perancangan *user interface* dan program aplikasi yang menggunakan dan memroses *database*.

2.2.7 *Prototyping*

Menurut Connoly (2002, p291), *prototyping* adalah membangun model kerja dari aplikasi *database*. Sebuah prototipe adalah model bekerja yang secara tidak normal memiliki semua fitur yang diminta atau menyediakan semua fungsi dari sistem akhir. Tujuan utama dari membangun *prototyping* dari aplikasi *database* adalah untuk memungkinkan *user* menggunakan protoipe untuk mengidentifikasi fitur sistem yang berjalan.

2.2.8 *Implementation*

Menurut Connoly (2002, p292), *implementation* adalah realisasi fisikal dari *database* dan aplikasi desain. Impelementasi *database* ini tersedia dengan menggunakan *Data Definition Language (DDL)* dari DBMS yang dipilih atau *Graphical ser Interface (GUI)*.

2.2.9 *Data conversion and loading*

Menurut Connoly (2002, p292), data ***conversion and loading*** adalah mengubah atau mentransfer berbagai data yang ada ke dalam *database* yang baru dan mengubah berbagai aplikasi yang sedang berjalan agar sesuai dengan *database* baru.

2.2.10 *Testing*

Menurut Connoly (2002, p293), *testing* adalah proses pelaksanaan program aplikasi dengan tujuan menemukan kesalahan. Sebelum

dihidupkan, aplikasi *database* yang baru dikembangkan sebaiknya diuji terdahulu. Hal ini tersedia dengan menggunakan strategi pengujian yang terencana dengan hati – hati dan data yang realistis.

2.2.11 Operational maintenance

Menurut Connoly (2002, p293), ***Operational maintenance*** adalah proses *monitoring* dan memelihara sistem mengikuti instalasi, agar sesuai dengan yang kita inginkan. Aktivitas pemeliharaan sebagai berikut :

- Memonitor tampilan sistem. Jika tampilan berada di bawah tingkat yang diinginkan, maka mengubah atau mereorganiasi *database* mungkin diminta.
- Memelihara dan meng*upgrade* aplikasi *database*

2.3 Normalisasi

Menurut Connolly (2002, p411), normalisasi adalah sebuah teknik untuk menghasilkan sekumpulan relasi dengan properti yang diinginkan, diberi permintaan permintaan data dari perusahaan.

2.3.1 Unnormalized Form (UNF)

Menurut Connoly (2002, p387), *unnormalized form* adalah sebuah table yang mengandung satu atau lebih *repeating group*. Ada dua

pendekatan yang dapat digunakan untuk menghilangkan *repeating groups* dari UNF :

- Dalam pendekatan pertama, *repeating group* dihilangkan dengan memasukkan data yang dalam kolom-kolom yang kosong baris-baris yang mengandung data yang berulang. Pendekatan ini sering disebut sebagai *"flattening the table"*
- Pendekatan kedua, *repeating group* dihilangkan dengan menempatkan data yang berulang bersamaan dengan salinan dari atribut *key* aslinya dalam relasi terpisah. Terkadang UNF dapat mengandung lebih dari satu *repeating group*, atau *repeating group* dalam *repeating group*. Dalam kasus seperti ini, pendekatan ini dilakukan berulang-ulang sampai tidak ada *repeating group* yang tersisa.

2.3.2 *First normal form (1NF)*

Menurut Connoly (2002, p388), *First normal form (1NF)* adalah sebuah *table* yang di dalam perpotongan kolom dan baris hanya memiliki satu nilai saja. Kita dapat menyebut skema relasi R sebagai *First Normal Form* jika domain dari seluruh atributnya atomik. 1NF memberikan permintaan yang sangat mendasar pada relasi dan tidak membutuhkan informasi tambahan seperti *functional dependency*. Untuk mengubah *unormalized table* ke 1NF, kita harus mengindentifikasikan dan membuang/memindahkan semua kumpulan data yang berulang.

2.3.3 *Second normal form (2NF)*

Menurut Connoly (2002, p392), *Second normal form (2NF)* adalah sebuah relasi yang berada dalam bentuk normal pertama dan setiap *non-primary-key* bergantung penuh (*full functionally dependent)* pada *primary key*. 2NF didasarkan pada konsep dari *full functional dependency*.

2.3.4 *Third normal form*

Menurut Connoly (2002, p394), *Third normal form (3NF)* adalah *table* yang setiap relasi yang terdapat didalamnya ada didalam *first normal form* dan *second normal form*, dimana tidak ada *non-primary-key* yang *transitive dependent* di *candidate key*.

3NF adalah salah satu solusi untuk memeriksa jika peng*update*an mengganggu *functional dependecy* mana saja. Dekomposisi 3NF dapat saja terdiri dari redundasi dalam skema yang terdekomposisi. Relasi skema R ada dalam 3NF berkenaan dengan sekumpulan F dari ketergantungan fungsional jika semua ketergantungan fungsional dalam F dari form $\alpha \rightarrow \beta$, setidaknya salah satunya memiliki :

- $\alpha \rightarrow \beta$ merupakan *functional dependency* yang diabaikan
- α adalah superkey bagi R
- setiap atribut A dalam $\beta \rightarrow \alpha$ terdapat dalam *candidate key* untuk R

2.3.5 *Boyce-Codd normal form (BCNF)*

Menurut Connoly (2002, p398), sebuah *table* dikatakan BCNF, jika dan hanya jika, semua *determinant* adalah *candidate-key*. Jika sebuah

skema relasi ada di dalam BCNF, maka yang menjadi satu-satunya *nontrivial functional dependencies* adalah *key constraint*. Sedangkan jika sebuah relasi ada di dalam 3NF, maka semua yang menjadi *nontrivial functional dependencies* adalah *key constraint* atau sebelah kanannya merupakan bagian dari *candidate key*. Karena hal inilah maka setiap relasi yang ada di dalam BCNF juga ada di dalam 3NF, namun tidak berlaku hal sebaliknya.

Sebuah relasi berpotensi menyalahi aturan BCNF bila :

- Memiliki dua atau lebih *composite candidate key*
- *Candidate Key* dari relasi *overlap*

2.3.6 *Fourth normal form (4NF)*

Menurut Connoly (2002, p408), 4NF adalah sebuah *table* yang ada di BCNF. Kita mungkin saja menggunakan *multivalue dependencies* untuk mendefinisikan *normal form* untuk skema relasi. *Normal form* ini yang disebut *Fourth Normal Form* atau 4NF. Kita akan menemukan bahwa setiap skema 4NF terdapat dalam BCNF, tapi ada skema BCNF yang tidak terdapat di dalam 4NF. Relasi skema R ada dalam 4NF berkenaan dengan sekumpulan D dari *multivalue dependency* dan fungsional jika untuk semua ketergantungan *multivalue* dalam D^+ pada form $\alpha \rightarrow \beta$, setidaknya salah satu memiliki :

- $\alpha \rightarrow \beta$ adalah *multivalue dependency* yang diabaikan
- α adalah *superkey* dari skema R

Menurut Connoly (2002, p409), *Lossless-Join Dependency* adalah sebuah aturan dekomposisi yang menjamin tidak ada *tuple* yang salah saat relasi disatukan melalui operasi *join*

2.3.7 Fifth normal form (5NF)

Menurut Connoly (2002, p410), 5NF adalah sebuah *table* yang tidak punya *join dependecy*.

4NF dan 5NF memiliki aturan yang lebih ketat daripada BCNF dan menghilangkan bak *redundancy* pada *multivalued* dan *join dependencies*.

Menurut Connoly (2002, p7),*File based system* adalah suatu kumpulan aplikasi yang melaksanakan jasa untuk *end-users* seperti menghasilkan laporan. Masing - masing program menggambarkan dan mengatur data sendiri.

2.4 *Relational Model*

2.4.1 Relation

Menurut Connoly (2002, p72), *relation* adalah suatu tabel dengan beberapa kolom dan baris.

2.4.2 Attribute

Menurut Connoly (2002, p72), *attribute* adalah kolom dari suatu tabel/relasi. Dalam model relasional, relasi digunakan untuk memegang informasi mengenai objek untuk ditampilkan di dalam *database*. Sebuah relasi ditampilkan sebagai tabel dua dimensi dimana baris dalam tabel cocok dengan *record* individual dan kolom tabel cocok dengan atribut.

2.4.3 *Domain*

Menurut Connoly (2002, p72), *domain* adalah seperangkat nilai-nilai yang memungkinkan untuk satu atau lebih atribut. Domain adalah fitur yang kuat dari model relasional. Setiap atribut di dalam relasi didefinisikan pada domain.

2.4.4 Tuple

Menurut Connoly (2002, p73), *tuple* adalah baris dari suatu *table*. *Tuple* merupakan elemen sebuah relasi.

2.4.5 Degree

Menurut Connoly (2002, p74), *degree* adalah banyaknya atribut/kolom dalam suatu *table*. Relasi dengan hanya satu atribut memiliki *degree* satu dan disebut relasi *unary* atau *one tuple*. Relasi dengan dua atribut disebut *binary*, sedangkan dengan tiga atribut disebut *ternary*, selanjutnya disebut *n-ary*.

2.4.6 Cardinality

Menurut Connoly (2002, p74), *cardinality* adalah banyaknya baris dalam suatu *table*. *Cardinality* dapat berubah-ubah mengikuti perubahan *tuple*, ditambah atau diapus. Kebalikan dari *degree*.

2.4.7 Relational database

Menurut Connoly (2002, p74), *relational database* adalah suatu koleksi *table* yang telah dinormalisasi dengan nama relasi beda. *Relational database* berisi relasi yang terstruktur secara tepat.

2.4.8 Relational key

Menurut Connoly (2002, p78), *relational key* digunakan untuk mengidentifikasi satu atau lebih atribut. Terminologi yang digunakan untuk *relational key* adalah:

- *Superkey*

  Menurut Connoly (2002, p78), *superkey* adalah suatu atribut, atau kempulan atribut, yang dengan uniknya mengidentifikasi suatu *tuple* di dalam suatu relasi.

- *Candidate key*

  Menurut Connoly (2002, p78), *candidate key* adalah suatu *super key* yang sedemikian rupa dimana tidak ada subset yang sesuai untuk menjadi *super key* di dalam *table*.

- *Primary key*

  Menurut Connoly (2002, p79), *primary key* adalah *candidate key* yang terpilih untuk mengindentifikasi *tuple* dengan uniknya di dalam *table*.

- *Foreign key*

  Menurut Connoly (2002, p79), *foreign key* adalah suatu atttribute, atau kumpulan atribut, di dalam satu relasi yang memenuhi *candidate key* dari beberapa (mungkin yang sama) tabel.

2.5 Metodologi

Bottom Up

Menurut Connolly (2002, p279), pendekatan *bottom up* dimulai pada tingkat fundamental atribut-atribut (yaitu properti dari entiti dan relasi), yang melalui analisis hubungan antar atribut dikelompokkan ke dalam relasi-relasi yang mewakilkan tipe entiti dan relasi-relasi antar entiti. Pada proses normalisasi, proses ini mewakili pendekatan desain *database* bottom up. Normalisasi mengikutsertakan identifikasi atribut-atribut yang diperlukan dan agregasinya pada relasi yang ternormalkan berdasarkan *functional dependencies* antar atribut.

Pendekatan ini tepat untuk desain *database* yang sederhana dengan atribut yang sedikit. Tetapi pendekatan ini menjadi sulit jika diterapkan pada desain

database yang kompleks dengan jumlah atribut yang besar, di mana model data logikal dan konseptual untuk *database* yang kompleks mengandung ratusan sampai ribuan atribut.

Top Down

Menurut Connolly (2002, p279), metodologi ini dimulai dengan pengembangan model data yang mengandung beberapa entiti tingkat tinggi dan relasinya kemudian mengaplikasikan pendetilan *top down* untuk identifikasi entiti tingkat rendah, relasinya dan atribut-atribut yang berhubungan. Model *Top Down* diilustrasikan menggunakan model konsep ER. Dimulai dengan indentifikasi entiti dan relasi antara entiti.

2.6 *Security Database*

2.6.1 Windows and SQL Authentication Mode

Menurut Robert Patton (p207), model autentikasi gabungan memungkinkan dukungan bagi kedua mekanisme autentikasi Windows dan SQL Server. Jika *option* ini dipilih, *user* akan diminta untuk memasukkan *password* untuk *sa account*. *Sa account* adalah login autentikasi SQL yang dibuat di dalam. Saat dilogin sebagai *sa*, *user* mempunyai hak penuh kepada SQL Server, yaitu *database*nya dan semua objek yang ada. Karena *sa account* SQL Server merupakan login yang berkuasa, maka dibutuhkan *password* yang aman.

Menurut Rankins et al (2002, p389), mode gabungan ini berguna untuk mendukung aplikasi peninggalan yang konek dengan menggunakan *account SQL Server* dan berada dalam lingkungan dimana pengendali domain *Windows* tidak mengendalikan akses jaringan.

2.6.2 Windows-only Authentication Mode

Menurut Robert Patton (2003,p210), *Windows-only authentication mode* adalah alternatif dari mode autentikasi *SQL Server* dan *Windows*. Autentikasi ini merupakan pilihan autentikasi *default* pada instalasi SQL Server 2000. Jika mode autentikasi *windows-only* digunakan, maka mekasnisme autentikasi SQL tidak dapat dipakai, dan hanya *login Windows* yang dapat mengakses SQL Server.

Beberapa keuntungan dari penggunaan mode autentikasi ini antara lain menurut Robert Patton (2003, p203 dan p210) , adalah: mudah dalam pemakaian mekanisme autentikasi Windows yang ada untuk mengautentikasi *user* saat *user* melakukan konek kepada SQL Server. Hal ini akan mengurangi kegiatan memasukkan berulang-ulang semua *username* dan *password*nya ke dalam penyimpanan data yang terpisah-pisah. Seorang *user* hanya mempunyai satu satu *user account*. Tidak akan ada duplikat *user account* yang membutuhkan *password* yang sinkron. Jika seorang user meninggalkan perusahaan dan *account*nya diapus, maka akses ke semua sumber dibatasi.

Keuntungan lain adalah keamanan jaringan. Mekanisme autentikasi Windows tidak mentransmit password pada jaringan. Hal ini memberi tingkat pengamanan tambahan saat klien melakukan konek kepada sumber seperti *SQL Server* melalui jaringan yang rawan. Keuntungan dari membatasi login pada *login Windows-only* adalah setidaknya satu jalan masuk keamanan pada server sudah ditutup.