UNIVERSITAS PERSATUAN GURU
REPUBLIK INDONESIA SEMARANG

BUKU SISTEM BASIS DATA

PENULIS:
Andi Priyolistiyanto, S.Kom., M.Kom.
Denata Jayanti

KATA PENGANTAR

Segala puji kepada Allah SWT yang telah melimpahkan segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan Buku Sistem Basis Data ini. Buku Sistem Basis Data ini diharapkan dapat mendukung mahasiswa dalam memahami mata kuliah Sistem Basis Data. Buku ini dibuat berdasarkan sumber-sumber yang sudah banyak digunakan.

Pada buku ini berisi mengenai basis data, sistem basis data, model data, teknik normalisasi data, teknik normalisasi data lanjutan, bahasa *query* formal, bahasa *query* terapan, bahasa *query* terapan lanjutan, basis data terdistribusi, lingkungan basis data, lingkungan basis data lanjutan, serta penjelasan lain terkait dengan sistem basis data.

Akhir kata, penulis mengucapkan terima kasih secara tulus kepada pihak-pihak yang telah memberikan bantuan dan dukungannya sehingga penulis dapat menyelesaikan penulisan buku ini. Penulis juga memohon maaf sebesar-besarnya jika dalam penulisan buku ini masih banyak kekurangan dan kelemahannya. Permohonan sumbangan ide, kritik, dan saran juga diperlukan untuk pengembangan dan perbaikan penulisan buku agar lebih baik kedepannya.

Semarang, Maret 2025
Penulis

DAFTAR ISI

BAB 1 BASIS DATA

1.1 Pengenalan Basis Data

Basis data (*database)*, terdiri dari kata basis dan data. Basis dapat diartikan sebagai markas atau gudang. Sedangkan data adalah suatu kumpulan yang terdiri dari fakta-fakta untuk memberikan gambaran yang luas terkait dengan suatu keadaan atau objek yang dinyatakan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Basis data adalah suatu susunan/kumpulan data operasional lengkap dari suatu organisasi/perusahaan yang diorganisir/dikelola dan disimpan secara terintegrasi dengan menggunakan metode tertentu dengan menggunakan komputer sehingga mampu menyediakan informasi yang diperlukan pemakainya.

Satu basis data menunjukkan kumpulan data yang dipakai dalam satu lingkup informasi. Dalam satu file terdapat *record-record* yang sejenis, sama besar, sama bentuk, merupakan satu kumpulan entitas yang seragam. Satu *record* terdiri dari *field-field* yang saling berhubungan untuk menunjukan bahwa *field* tersebut dalam satu pengertian yang lengkap dan direkam dalam satu *record*.

1.2 Konsep Dasar Basis Data

Suatu Basis data merupakan kumpulan data terpusat dan terstruktur yang disimpan dalam sistem komputer, dimana sistem tersebut menyediakan fasilitas untuk mengambil, menambahkan, memodifikasi dan menghapus data saat diperlukan. Sistem juga menyediakan fasilitas untuk dapat mengubah data yang diambil menjadi informasi yang berguna.

Basis data *(database)* dapat dibayangkan atau digambarkan sebagai sebuah lemari arsip. Jika kita memiliki sebuah lemari arsip dan bertugas untuk mengelolanya, maka kemungkinan yang akan dilakukan adalah: memberi sampul/map pada kumpulan arsip yang disimpan, menentukan kelompok/jenis arsip, memberi penomoran dengan pola tertentu yang nilainya unik pada setiap sampul/map maupun penomoran kelompok atau jenis arsip, menempatkan arsip tersebut dengan cara/urutan tertentu di lemari arsip

Walaupun hal-hal tersebut tidak seluruhnya dilakukan, paling tidak semua lemari arsip menerapkan suatu aturan tertentu bagaimana keseluruhan arsip-arsip tersebut disusun. Tujuannya adalah jika pada saat data atau arsip tersebut dibutuhkan, maka dapat ditemukan dan diperoleh dengan cepat dan mudah.

1.3 Fungsi Basis Data

1. *Availability*, adalah untuk menyediakan data-data penting saat sedang diperlukan, meskipun tidak terletak dalam satu lokasi dan tersimpan dalam bentuk *disk*, akan tetapi dengan cara penyimpanan yang sistematik tersebut informasi menjadi mudah untuk didapatkan.

2. *Speed*, basis data mempunyai kemudahan dan kecepatan pada saat harus mengalokasikan waktu tertentu untuk memanggilnya dan memungkinkan untuk melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data dengan lebih cepat.

3. *Completeness,* basis data harus menyimpan data yang lengkap, yang bisa melayani keperluan penggunanya secara keseluruhan. Meski kata lengkap yang dipakai disini sifatnya relatif, namun setidaknya data tersebut membantu memudahkan untuk menambah koleksi data, dan menjamin mudahnya pengguna untuk memodifikasi struktur data yang ada.

4. *Security,* ada fasilitas pengaman data yang disediakan oleh sistem basis data yang baik sehingga data tidak bisa dimodifikasi, diakses, diubah maupun dihapus oleh yang tidak mendapatkan hak untuk melakukannya.

5. *Storage Efficiency,* pengorganisasian data dilakukan dengan baik dengan tujuan untuk menghindari duplikasi data yang berpengaruh pada bertambahnya ruang penyimpanan dari basis data tersebut. Pengkodean dan relasi data bermanfaat untuk menghemat *space* penyimpanan dalam basis data.

1.4 Prinsip dan Tujuan Basis Data

Prinsip kerja dan tujuan basis data sama dengan prinsip kerja dan tujuan dari lemari arsip. Prinsip utamanya adalah pengaturan data atau arsip, dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali dari data atau arsip tersebut. Tujuan utama dari basis data adalah untuk mengatur data sehingga diperoleh kemudahan, kecepatan dan ketepatan dalam memanggil kembali data yang diinginkan. Perbedaan basis data dengan lemari arsip hanya terletak pada media penyimpanan. Basis data menggunakan media penyimpanan elektronik.

1.5 Operasi Dasar Basis Data

Basis Data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. Ada beberapa operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data, yaitu:

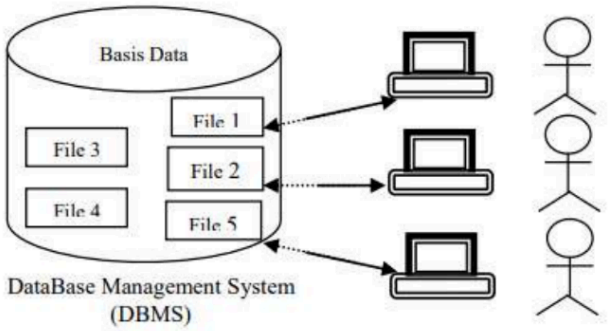
1. *Create database*, pembuatan basis data baru.
2. *Drop database*, penghapusan basis data.
3. *Create table*, pembuatan tabel baru ke suatu basis data.
4. *Drop table*, penghapusan tabel dari suatu basis data.
5. *Insert*, penambahan atau pengisian data baru ke sebuah tabel di sebuah basis data.
6. *Query*, pengambilan data dari sebuah tabel.
7. *Update*, pengubahan data dari sebuah tabel.
8. *Delete*, penghapusan data dari sebuah tabel.

BAB 2 SISTEM BASIS DATA

2.1 Definisi Sistem Basis Data

Pengertian dari sistem adalah sebuah tatanan/keterpaduan yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi/tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu. Sistem basis data adalah kumpulan dari program aplikasi yang berinteraksi dengan basis data bersama dengan *Database Management System (DBMS)* dan basis data itu sendiri.

Sistem yang terdiri atas kumpulan file/tabel yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program *DBMS* yang memungkinkan beberapa pemakai yang menggunakan basis data secara bersama-sama, personil yang merancang dan mengelola basis data, teknik-teknik untuk merancang dan mengelola basis data, serta sistem komputer yang mendukungnya dan/atau program lain untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut, seperti terlihat pada gambar di bawah ini.



Sehingga dapat disimpulkan bahwa basis data adalah kumpulan informasi yang disimpan didalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut.

2.2 Komponen Sistem Basis Data

A. Data, disimpan secara terintegrasi, yaitu *database* merupakan kumpulan dari berbagai macam file dari aplikasi-aplikasi yang berbeda yang disusun dengan cara menghilangkan bagian-bagian yang rangkap. Kemudian data dapat dipakai secara bersama-sama, yaitu masing-masing bagian dari *database* dapat diakses oleh pemakai dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

B. Perangkat keras/*hardware*, terdiri dari semua peralatan perangkat keras komputer yang digunakan untuk pengelolaan sistem *database*. Perangkat keras yang terdapat dalam sebuah sistem basis data adalah: komputer (satu untuk *sistem stand-alone* atau lebih dari satu untuk sistem jaringan), memori sekunder *online* (*hard disk*), memori sekunder *offline* (*tape* atau *removable disk*) untuk *backup* data, media/perangkat komunikasi (untuk sistem jaringan).

C. Perangkat lunak/*software,* berfungsi sebagai perantara/*interface* antara pemakai dengan data fisik pada *database*, dapat berupa : *DBMS*, program-program aplikasi & prosedur-prosedur.

D. Pemakai/*user* adalah pengguna basis data yang berinteraksi secara tidak langsung dengan basis data melalui program aplikasi basis data dan *DBMS*. Terbagi menjadi 3 klasifikasi : *Data Base Administrator (DBA)* yang membuat basis data dan mengontrol akses ke basis data, *programmer* yang membuat aplikasi basis data yang digunakan oleh *DBA* dan pemakai akhir, pemakai akhir yang melakukan penambahan, penghapusan, pengubahan, dan pengaksesan data.

2.3 Perancangan Basis Data

Tujuan perancangan basis data adalah untuk memenuhi informasi yang berisi kebutuhan–kebutuhan user secara khusus dan aplikasinya, memudahkan pengertian struktur informasi, mendukung kebutuhan–kebutuhan pemrosesan dan beberapa objek penampilan (*response time, processing time* dan *storage space*).

Dalam perancangan basis data terdiri dari dua tahapan:

A. Tahap analisis dan perancangan. Pada tahap ini dilakukan pemetaan atau pembuatan model dari dunia nyata menggunakan notasi perancangan basis data tertentu serta pembuatan deskripsi implementasi basis data. Tahapan analisis dan perancangan dibagi menjadi:
   1) Perancangan basis data secara konsep, proses pembuatan data model dan tidak bergantung pada seluruh aspek fisik basis data.
   2) Perancangan basis data secara logis, proses pembuatan data model berdasarkan data model tertentu, tidak bergantung pada *DBMS* tertentu/implementasi fisik basis data.
   3) Perancangan basis data secara fisik, proses pembuatan deskripsi implementasi basis data pada media penyimpanan sekunder (*disk*), menjelaskan tabel tabel dasar, organisasi file, indeks untuk mendapatkan akses data secara efisien, *integrity constraints,* sampai dengan langkah-langkah keamanan.

B. Tahap implementasi. Tahapan ini mengimplementasikan rancangan basis data yang telah dibuat. Implementasi dilakukan dengan aplikasi *client* yang disediakan oleh *DBMS* yang telah dipilih.

2.4 Definisi *Database Management System (DBMS)*

Data Base Management System (DBMS) adalah perangkat lunak yang memungkinkan pemakai untuk mendefinisikan, mengelola, dan mengontrol akses ke basis data. *DBMS* yang mengelola basis data *relational* disebut dengan *Relational Data Base Management System (RDBMS)*. Berikut beberapa contoh *software DBMS* seperti: Microsoft-Access, MySQL, MSSQL Server, Oracle, dBase, FoxBase, Rbase, Borland Paradox, dll.

2.5 Komponen *Database Management System (DBMS)*

1. *Query Processor*, komponen yang mengubah bentuk *query* kedalam instruksi ke dalam *database manager*.
2. *Database Manager*, menerima *query* & menguji eksternal & konseptual untuk menentukan apakah *record–record* tersebut dibutuhkan untuk memenuhi permintaan kemudian *database manager* memanggil *file manager* untuk menyelesaikan permintaan.
3. *File Manager*, memanipulasi penyimpanan *file* dan mengatur alokasi ruang penyimpanan *disk*.
4. *DML Precompiler*, modul yang mengubah perintah *DML* yang ditempelkan kedalam program aplikasi dalam bentuk fungsi-fungsi.
5. *DDL Compiler*, merubah statement *DDL* menjadi kumpulan table atau file yang berisi data *dictionary* / *meta data*.
6. *Dictionary Manajer*, mengatur akses dan memelihara *data dictionary*.

2.6 Kelebihan dan Kekurangan *Database Management System (DBMS)*

Kelebihan *DBMS*	Kekurangan *DBMS*
Mengurangi pengulangan data dan konsistensi data jika ada perubahan	Harga *DBMS* yang mahal
Meningkatnya pemeliharaan karena independensi data	Biaya perangkat keras tambahan
Mengintegrasikan data beberapa *file* sehingga lebih banyak informasi dari jumlah data yang sama	Biaya konversi teknologi
Mengambil data dan informasi dengan cepat	Kompleksitas
Meningkatkan keamanan dan pengontrolan kerangkapan data	Ukuran karena tambahan kompleksitas yang memerlukan banyak tempat penyimpanan
Sharing data	Dampak kegagalan yang lebih besar

BAB 3 MODEL DATA

3.1 Definisi Model Data

Model data merupakan suatu cara untuk menjelaskan bagaimana pemakai dapat melihat data secara logik. Berisi sekumpulan konsep-konsep untuk menerangkan data, hubungan-hubungan antara data dan batasan-batasan data yang terintegrasi.

Model data dapat didefinisikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantic (makna) data dan batasan data. Pada umumnya sebuah model dinyatakan dalam bentuk diagram yang dibuat di awal akan lebih mudah untuk dievaluasi maupun dianalisis untuk kemudian dilakukan perbaikan perbaikan untuk mendapatkan sebuah model data yang lebih permanen dan lebih mendekati kenyataan sesungguhnya.

3.2 Jenis Model Data

1. Model data berbasis objek, menggunakan konsep entitas, atribut dan hubungan antar entitas (*relationship*). *Entity* adalah sesuatu apa saja yang ada didalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokkan dalam empat jenis nama. Atribut adalah relasi fungsional dari satu objek set ke objek set yang lain. Sedangkan, *Relationship* adalah hubungan alamiah yang terjadi antara entitas. Terdiri dari:

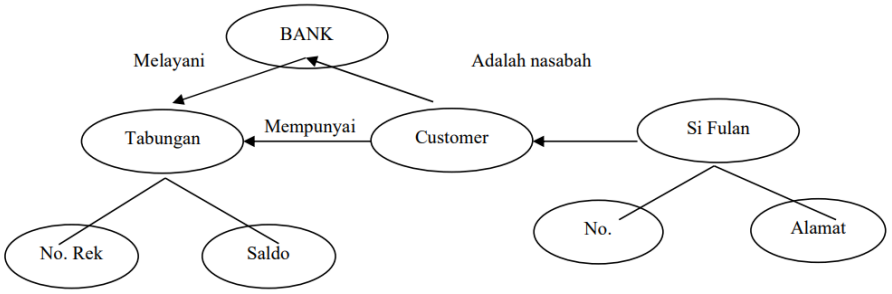
   a. Model Keterhubungan Entitas (*Entity-Relationship Model*)
   Entitas merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan atau relasi antara objek-objek tersebut. Model ini termasuk model yang paling populer digunakan dalam perancangan basis data. Komponen utama pembentuk *Model Entity-Relationship* yaitu Entitas (*Entity*), Relasi (*Relation*).

Arti simbol:

Entitas Atribut Relasi

b. Model Berorientasi Objek (*Object-Oriented Model*)
Penggambaran model berbasis objek menggunakan *Unified Modeling Language*, digambarkan dengan:
- *Structural Diagram*, terdiri dari: *Class Diagram, Object Diagram, Component Diagram, Deployment Diagram*.
- *Behaviour Diagram*, terdiri dari: *Use case Diagram, Sequence Diagram, Communication Diagram, Statechart Diagram, Activity Diagram*.

c. Model Data Semantik (*Semantic Data Model*)
Hampir sama dengan *Entity Relationship Model* dimana relasi antara objek dasar tidak dinyatakan dengan simbol tetapi menggunakan kata-kata (*Semantic*).



Arti simbol:

d. Model Data Fungsional (*Functional Data Model*)

2. Model Data Berdasarkan *Record*, berdasarkan pada *record* untuk menjelaskan kepada *user* tentang hubungan logis antar data dalam basis data. Perbedaan dengan model data berbasis objek adalah pada *record based* data model yang digunakan untuk menguraikan struktur logika keseluruhan dari suatu *database*, juga digunakan untuk menguraikan implementasi dari sistem *database*. Berikut adalah jenis-jenis model data berbasis *record*:

a. *Relational Model*
Dimana data serta hubungan antar data direpresentasikan oleh sejumlah tabel dan masing masing tabel terdiri dari beberapa kolom yang namanya unik.

b. *Hierarki Model*
Dimana data serta hubungan antar data direpresentasikan dengan *record* dan *link* (pointer), dimana record-record tersebut disusun dalam bentuk *tree* (pohon), dan masing-masing node pada *tree* tersebut merupakan *record/group* data elemen dan memiliki hubungan kardinalitas 1:1 dan 1:*Many*.

c. Model Jaringan
Model dimana data dan hubungan antar data direpresentasikan dengan *record* dan *links*. Perbedaannya terletak pada susunan *record* dan *link*-nya yaitu model jaringan menyusun *record-record* dalam bentuk *graph* dan menyatakan hubungan kardinalitas 1:1, 1:*Many* dan *Many:Many*.

BAB 4 TEKNIK NORMALISASI

4.1 Definisi Normalisasi

Normalisasi adalah suatu proses memperbaiki atau membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika. Normalisasi adalah proses pengelompokan *atribute-atribute* dari suatu relasi sehingga membentuk *well structure relation*.

Normalisasi merupakan cara pendekatan lain dalam membangun desain logikal basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal.

Normalisasi merupakan bentuk atau memecah tabel dalam suatu *database* dari *record-record* data yang komplek menjadi lebih sederhana. Normalisasi juga merupakan proses pengelompokan elemen data menjadi tabel yang menunjukkan entitas sekaligus relasinya. Teknik memecah tabel yang komplek menjadi sederhana biasa dikenal dengan istilah dekomposisi.

4.2 Bentuk Normalisasi

Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi dalam basis data, berikut beberapa level yang biasa digunakan pada normalisasi adalah:

4.2.1 Bentuk Normal Pertama (*First Normal Form* / 1NF)

Pada tahap ini dilakukan penghilangan grup elemen yang berulang agar menjadi satu harga tunggal yang berinteraksi diantara setiap baris pada suatu tabel, dan setiap atribut harus mempunyai nilai data yang *atomic* (*atomic value*). Sebuah tabel dikatakan pada bentuk 1NF ketika masing masing *record* data pada tabel memiliki pasangan satu *record* data. Suatu relasi memenuhi 1NF jika dan hanya jika setiap atribut dari relasi hanya memiliki nilai tunggal dalam baris *record*. Bentuk 1NF bertujuan untuk menghilangkan ketergantungan parsial.

4.2.2 Bentuk Normal Kedua (*Second Normal Form* / 2NF)

Suatu relasi memenuhi 2NF jika dan hanya jika memenuhi 1NF. Setiap atribut yang bukan kunci utama tergantung secara fungsional terhadap semua atribut kunci dan tidak hanya sebagian atribut kunci. Bentuk 2NF bertujuan untuk menghilangkan ketergantungan transitif.

Harus berlaku : A, B → C, D, E	artinya: A, B → C A, B → D A, B → E

Non 2NF apabila	diketahui R = (A,B,C,D,E) A, B → C, D B → E	Atribut E hanya tergantung secara fungsional terhadap B saja dan bukan terhadap A, B. Maka R menjadi Non 2NF.

2NF apabila	diketahui R1 = (A,B,C,D) R2 = (B,E)	Maka dapat didekomposisi jadi 2 relasi. Relasi R1 dan R2 secara keseluruhan memenuhi 2NF, karena semua atribut bukan kunci dan tergantung secara fungsional terhadap semua atribut kunci.

Langkah melakukan normalisasi 2NF adalah mencari, temukan dan hilangkan atribut yang hanya memiliki ketergantungan pada sebagian *key* tidak pada semua *key*. Kemudian letakkan atribut tersebut pada tabel yang berbeda. Setelahnya kelompokkan atribut lainnya.

4.2.3 Bentuk Normal Ketiga (*Third Normal Form* / 3NF)

Bentuk normal ketiga adalah jika suatu relasi memiliki syarat: berada dalam bentuk 2NF, setiap atribut bukan kunci haruslah tidak memiliki dependensi transitif (ketergantungan transitif) terhadap kunci primer, dengan kata lain suatu atribut bukan kunci tidak boleh memiliki ketergantungan fungsional terhadap atribut bukan kunci lainnya, seluruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap *primary key* di relasi itu saja.

Non 3NF apabila	diketahui R = (A,B,C,D,E) A, B → C, D, E C → D, E	Atribut D, E yang bukan kunci tergantung secara fungsional kepada C yang juga bukan atribut kunci. Maka R menjadi Non 3NF.

3NF apabila	diketahui R1 = (A,B,C) R2 = (C,D,E)	Maka dapat didekomposisi jadi 2 relasi. Relasi R1 dan R2 secara keseluruhan memenuhi 3NF. Suatu relasi memenuhi 2NF dan hanya memiliki 1 atribut bukan kunci selalu memenuhi 3NF.

Langkah melakukan normalisasi 3NF adalah mencari dan pisahkan atribut yang memiliki ketergantungan fungsional pada atribut yang bukan non-key dan letakkan pada tabel yang berbeda. Kelompokkan sisa atribut lainnya pada tabel.

4.2.4 Bentuk Normal *Boyce-Codd* (*Boyce-Codd Normal Form* / BCNF)

BCNF merupakan bentuk normal sebagai perbaikan terhadap 3NF. Suatu relasi yang memenuhi BCNF selalu memenuhi 3NF, tetapi tidak untuk sebaliknya. Jika semua ketergantungan fungsional A → B, maka A harus merupakan *super key* pada tabel tersebut. Jika tidak maka tabel tersebut harus di dekomposisi berdasarkan ketergantungan fungsional yang ada, sehingga A menjadi *super key* dari tabel-tabel hasil dekomposisi.

Bentuk BCNF bertujuan untuk menghilangkan ketergantungan multi nilai. Suatu relasi memenuhi BCNF jika dan hanya jika setiap determinan yang ada pada relasi tersebut adalah *candidate key*. Determinan adalah atribut dimana satu atau lebih atribut lain tergantung secara fungsional.

Non BCNF apabila	diketahui R = (A,B,C,D,E) A, B → C, D, E C → D, E	Ada determinan yang bukan *candidate key* yaitu B, C, sedangkan A, B adalah determinan yang juga merupakan *candidate key*. A, B → C, D, E C → A, B A, B → A, B, C, D, E Jadi A, B merupakan *candidate key*. Maka R menjadi Non BCNF.

BCNF apabila	diketahui R1 = (A,B,C) R2 = (B,C,D,E)	Maka dapat didekomposisi jadi 2 relasi. Relasi R1 dan R2 secara keseluruhan memenuhi BCNF.

BCNF dibutuhkan jika 3NF masih terjadi anomali pada perubahan dan penghapusan data. Ciri-ciri yang dilakukan pada BCNF adalah: tabel banyak memiliki *candidate key*, *candidate key* memiliki sifat komposit, *candidate key* overlap.

Langkah melakukan normalisasi BCNF adalah mencari dan hilangkan *candidate key* yang overlap. Kemudian letakkan sebagian *candidate key* dan atribut yang memiliki ketergantungan fungsional pada tabel yang berbeda. Kelompokkan item lainnya ke dalam sebuah tabel.

4.2.5 Bentuk Normal Keempat (Fourth Normal Form / 4NF)

4NF berkaitan sifat ketergantungan banyak nilai (*Multivalued Dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Suatu relasi memenuhi 4NF jika dan hanya jika memenuhi BCNF dan tidak memiliki ketergantungan multi nilai atribut. Setiap *multi dependencies* merupakan *functional dependencies*. Bentuk 4NF bertujuan untuk menghilangkan anomali yang tersisa. Bentuk 4NF ini tidak mengandung dua atribut atau lebih yang bernilai banyak.

4NF apabila	diketahui R = (A,B,C)	artinya	A → B A → C A → B | C

4.2.6 Bentuk Normal Kelima (Fifth Normal Form / 5NF)

Suatu relasi memenuhi 5NF jika dan hanya jika setiap dependensi gabungan dalam R. 5NF tidak dapat memiliki sebuah *lossless decomposition* dan memenuhi 4NF. 5NF dibentuk berdasarkan konsep *join dependencies*.

5NF apabila	diketahui R = (V,W,X,Y,Z) Memenuhi dependensi gabungan, dari proyeksi A, B, C merupakan sub himpunan dari atribut R. Dependensi gabungan dinyatakan dengan notasi *(A, B, C)	artinya	A → X, Y B → W, Z C → Z, V

4.2.7 Bentuk *Project-Join (Project-Join Normal Form / PJNF)*

PJNF merupakan bentuk lain dari 5NF yang berhubungan dengan ketergantungan relasi antar tabel (*Join Dependency*).

4.3 Keuntungan dan Syarat Normalisasi

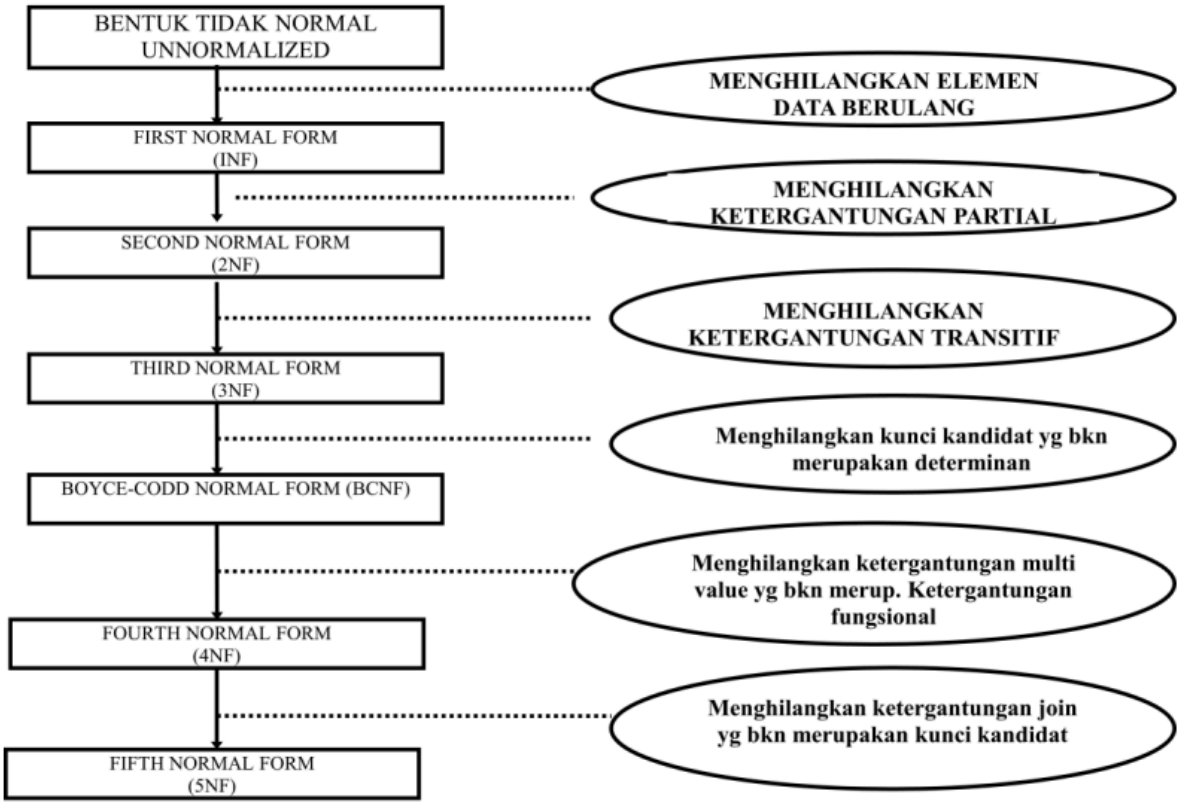
Normalisasi adalah sebuah metode yang urut dalam penerapan aturan-aturan untuk mendesain database dengan tujuan agar (table menjadi normal), meminimalkan redundancy dan pengulangan data, mempertahankan integritas data, menambah konsistensi dan stabilitas, menghilangkan potensi anomaly ketika mengolah data.

Keuntungan dari normalisasi: meminimalkan ukuran penyimpanan yang diperlukan untuk menyimpan data, meminimalkan resiko inkonsistensi data pada basis data, meminimalkan kemungkinan anomali pembaruan, memaksimalkan stabilitas struktur data.

Syarat perlunya normalisasi	
Fleksibilitas	Struktur database harus menunjang semua cara untuk menampilan data, sehingga ketika user menjalankan aplikasi dan meminta sesuatu dalam database, database harus dapat berjalan memenuhi permintaan user.
Integritas Data	Semua data dalam database yang berkaitan harus terhubung dalam suatu relationship. Sehingga ketika suatu data berubah, maka semua data yang berkaitan dengan data tersebut harus dapat berubah secara otomatis.

Efficiency	Pada database, ukuran suatu database merupakan hal yang penting. Maka dari itu kita harus mengurangi redudansi data yang bisa menyebabkan ukuran database menjadi besar.
Menghindari Modifikasi Anomali	Desain database yang baik menyajikan suatu keyakinan bahwa ketika user melakukan perubahan dalam database, maka tidak terjadi hal yang tidak diinginkan.

4.4 Langkah-Langkah Pembuatan Normalisasi



BAB 5 TEKNIK NORMALISASI LANJUTAN

5.1 Definisi *Anomaly*

Anomaly merupakan penyimpangan-penyimpangan atau *error* atau inkonsistensi data yang terjadi pada saat dilakukan proses *insert*, *delete* maupun *update* dalam suatu basis data.

5.2 Jenis *Anomaly*

Terdapat 3 jenis anomaly :

1. *Insertion Anomaly*, merupakan *error* atau kesalahan yang terjadi sebagai akibat operasi *insert record/tuple* pada sebuah *relation*.
2. *Deletion Anomaly* merupakan *error* atau kesalahan yang terjadi sebagai akibat operasi *delete record/tuple* pada sebuah *relation*.
3. *Update Anomaly* merupakan *error* atau kesalahan yang terjadi sebagai akibat inkonsistensi data yang terjadi sebagai akibat dari operasi *update record/tuple* dari sebuah *relation*.

5.3 Kebergantungan Fungsi

1. Ketergantungan Fungsional (*Functionality Dependent*)
   Keterkaitan antar hubungan antara 2 *atribute* pada sebuah relasi. Dituliskan dengan cara : A → B, yang berarti :
   Attribute B *functionality dependent* terhadap *attribute* A atau isi (*value*) *attribute* A menentukan isi *attribute* B.
   Definisi dari functional dependent:
   Diketahui sebuah relasi R, *attribute* Y dari R adalah FD pada *attribute* X dari R ditulis R.X → R.Y jika dan hanya jika tiap harga X dalam R bersesuaian dengan tepat satu harga Y dalam R.

2. *Fully Functionally Dependent (FFD)*
   Suatu rinci data dikatakan *fully functional dependent* pada suatu kombinasi rinci data jika *functional dependent* pada kombinasi rinci data dan tidak *functional dependent* pada bagian lain dari kombinasi rinci data.
   Definisi dari FDD:
   Attribute Y pada relasi R adalah FFD pada *attribute* X pada relasi R jika Y FD pada X tida FD pada himpunan bagian dari X. Contoh:
   Bukan FDD: PersonID,Project, Project_budget → time_spent_byperson_ onProject
   FDD : PersonID, Project → time_spent_byperson_onProject

3. Ketergantungan Partial
   Sebagian dari kunci dapat digunakan sebagai kunci utama.

4. Ketergantungan Transitif
   Menjadi *attribute* biasa pada suatu relasi tetapi menjadi kunci pada relasi lain.

5. Determinan
   Suatu *attribute* (*field*) atau gabungan *attribute* dimana beberapa *attribute* lain bergantung sepenuhnya pada *attribute* tersebut.

5.4 *Key* dan Atribut Deskriptif

5.4.1 Atribut Deskriptif

Key merupakan satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*record*) dalam tabel secara unik. Jika suatu atribut dijadikan sebuah *key*, maka tidak diperkenankan ada dua atau lebih data dengan nilai yang sama untuk atribut tersebut. Macam-macam atribut yaitu :

1. Atribut Sederhana, yaitu atribut *atomic* yang tidak dapat dipisahkan lagi.
2. Atribut Komposit, atribut yang dapat dipisahkan menjadi sub atribut masing-masing memiliki makna, dapat didekomposisi menjadi atribut sederhana.
3. Atribut Bernilai Tunggal, atribut memiliki paling banyak satu nilai setiap *record*.
4. Atribut Bernilai Banyak, adalah atribut yang dapat memiliki lebih dari satu nilai, tetapi jenisnya sama.
5. Atribut Harus Bernilai, sejumlah atribut pada sebuah tabel yang harus berisi data dan tidak boleh kosong.
6. Atribut Turunan, atribut yang nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan.

5.4.2 *Super-Key*

Salah satu atau lebih atribut (kumpulan atribut) dari suatu tabel yang dapat digunakan untuk mengidentifikasi *entity/record* dari tabel tersebut secara unik (tidak semua atribut dapat menjadi *super key*). Merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data (*record*) dalam sebuah tabel secara unik.

5.4.3 *Candidate-Key*

Tidak boleh berisi atribut dari tabel yang lain sehingga *candidate key* sudah pasti *super key* namun belum tentu sebaliknya. Ciri-ciri *candidate key* :
Tidak dapat berisi atribut, kumpulan atribut yang telah menjadi *super key*, sebuah tabel dimungkinkan adanya lebih dari satu *candidate key*, apabila *candidate key* lebih dari satu, maka salah satu *candidate key* dapat dijadikan sebagai *primary key*.

5.4.4 *Primary-Key*

Salah satu atribut *candidate key* dapat dipilih menjadi *primary key* dengan 3 (tiga) kriteria yaitu *key* tersebut lebih natural untuk digunakan sebagai acuan, lebih sederhana, terjamin keunikannya.

Apabila ada lebih dari satu yang menjadi *candidate key*, maka *key* dapat dijadikan *primary key*. Dasar pemilihan *primary key* adalah: *key* tersebut lebih sering untuk dijadikan sebagai acuan, *key* tersebut lebih ringkas, jaminan keunikan *key* tersebut lebih baik.

5.5.5 *Alternate-Key*

Setiap atribut *candidate key* yang tidak terpilih menjadi *primary key* maka atribut-atribut tersebut dinamakan *alternate key*.

BAB 6 BAHASA *QUERY* FORMAL

6.1 Definisi Bahasa *Query* Formal

Dalam bahasa *Query* Formal, ada dua dasar pembentukan bahasa *Query*, yaitu:

1. Aljabar Relasional
2. Kalkulus Relasional

Bahasa *query* merupakan bahasa yang termasuk dalam kategori bahasa tingkat tinggi (*high level language*) yang digunakan *user* untuk mendapatkan informasi atau data dari basis data.

6.2 Kelompok Bahasa *Query*

Bahasa *query* dikelompokkan menjadi dua, yaitu:

1. Bahasa *procedural*, *user* meminta sistem untuk melakukan serangkain operasi terhadap basis data dalam rangka mendapatkan data atau informasi yang diinginkan.
2. Bahasa *non procedural*, *user* menunjukkan data atau informasi yang diinginkan tanpa menyatakan suatu cara atau prosedur tertentu untuk memperoleh data atau informasi tersebut.

6.3 Operator *Aljabar Relational*

Aljabar relasional merupakan salah satu bahasa manipulasi untuk *database* relasional. Aljabar relasional merupakan kumpulan operasi terhadap relasi dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru. Aljabar relasional termasuk dalam kategori bahasa *procedural* yang menyediakan seperangkat operasi untuk memanipulasi data.

Operator dasar:

a. *Selection* (σ) *Lower Case Omega*. Operasi *selection* menyeleksi tupel-tupel pada sebuah relation yang memenuhi *predicate*/syarat yang sudah ditentukan.
b. *Projection* (π) Operator *projection* beroperasi pada sebuah *relation*, yaitu membentuk *relation* baru dengan meng-*copy atribute-atribute* dan domain-domain dari *relation* tersebut berdasarkan argumen-argumen pada operator tersebut.
c. *Cartesian product* (X) Operator dengan dua relasi untuk menghasilkan tabel hasil perkalian kartesian.
d. *Union* (∪) Operasi untuk menghasilkan gabungan tabel dengan syarat kedua tabel memiliki atribut yang sama yaitu domain atribut ke-i masing-masing tabel harus sama.
e. *Set difference* (-) Operasi untuk mendapatkan tabel di suatu relasi tapi tidak ada di relasi lainnya.

Operator tambahan:

a. *Set Intersection* (∩) Operasi untuk menghasilkan irisan dua tabel dengan syarat kedua tabel memiliki atribut yang sama, domain atribut ke-i kedua tabel tersebut sama.
b. *Theta Join* (θ), Operasi yang menggabungkan operasi *cartesian product* dengan operasi *selection* dengan suatu kriteria.
c. *Natural Join* (⋈), Operasi menggabungkan operasi *selection* dan *cartesian product* dengan suatu kriteria pada kolom yang sama.
d. *Division* (÷), Merupakan operasi pembagian atas tuple-tuple dari 2 *relation*.

6.4 Operator *Kalkulus Relational*

Kalkulus relasional merupakan bahasa manipulasi teoritis yang non *procedural*. Kalkulus relasional dilandasi dengan teori *predicate calculus* yang menggunakan fungsi sebagai suatu ekspresi *logic*. Predikat adalah suatu fungsi yang dapat mengambil nilai benar atau salah tergantung dari substitusi nilai argumen dari fungsi tersebut. Jadi, bila semua argumen dari sebuah fungsi disubstitusi dengan suatu nilai, maka fungsi tersebut menjadi suatu ekspresi yang disebut preposisi, yaitu suatu ekspresi yang hanya bernilai.

BAB 7 BAHASA *QUERY* TERAPAN

7.1 Definisi *Structure Query Language (SQL)*

Structured Query Language (SQL) merupakan bahasa *query* terapan yang banyak digunakan oleh berbagai DBMS, diterapkan dalam berbagai *development tools* dan program aplikasi untuk berinteraksi dengan basis data. Bahasa ini dibangun dengan dasar *Aljabar Relational* dan sedikit *Kalkulus Relational*. SQL merupakan bahasa yang mudah dipelajari karena termasuk ke dalam bahasa *non procedural*, cukup menspesifikasikan informasi apa yang dibutuhkan daripada bagaimana mendapatkannya.

7.2 Pengelompokkan *Statement Query Language (SQL)*

Data Definition Language (DDL)	CREATE DATABASE DROP DATABASE CREATE TABLE DROP TABLE CREATE INDEX DROP INDEX CREATE VIEW DROP VIEW ALTER TABLE
Data Manipulation Language (DML)	INSERT, SELECT, UPDATE, DELETE
Data Access	GRANT, REVOKE
Data Integrity	RECOVER TABLE
Auxiliary	SELECT INTO OUTFILE, LOAD, RENAME TABLE

7.2.1 *Data Definition Language (DDL)*

Data Definition Language (DDL) adalah kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, atribut (kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Termasuk kelompok DDL adalah:

a. *Create* (Pembuatan) Pembuatan Database
   Pembuatan Tabel
   Pembuatan Index
   Pembuatan View
b. *Drop* (Menghapus) Menghapus Database
   Menghapus Tabel
   Menghapus Index
   Menghapus View
c. *Alter Table* Merubah struktur tabel

7.2.2 *Data Manipulation Language (DML)*

Data Manipulation Language (DML) adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan. Yang termasuk dalam kelompok DML sebagai berikut:

a. *Insert* — Sintaks SQL yang digunakan untuk penambahan record baru ke dalam sebuah tabel.
b. *Select* — Mempunyai fungsi untuk menampilkan isi tabel.
c. *Update* — Sintaks SQL yang digunakan untuk mengubah nilai atribut pada suatu record dari sebuah tabel.
d. *Delete* — Sintaks SQL yang digunakan untuk menghapus record dari sebuah tabel.

7.2.3 *Data Access (Data Control Language/DCL)*

Data Access (Data Control Language / DCL) adalah suatu perintah-perintah untuk mengendalikan pengaksesan data. Pengendalian dapat dilakukan per tabel, per kolom maupun per operasi. Yang termasuk kelompok *DCL* atau *Data Access* yaitu:

a. *Grant* — Mempunyai fungsi untuk memberikan hak akses.
b. *Revoke* — Mempunyai fungsi untuk mencabut kembali hak akses yang sudah diberikan.

7.2.4 *Data Integrity*

Data Integrity adalah suatu perintah yang digunakan untuk mengembalikan data sebelum terjadi kerusakan.Yang termasuk di dalam kelompok Data Integrity adalah Recover Table.

7.2.5 *Data Auxiliary*

Data Auxiliary adalah suatu perintah yang digunakan untuk mengubah data maupun kolom pada tabel. Yang termasuk kelompok *data auxiliary* sebagai berikut:

a. *Select ... Into Outfile 'filename'* — Sintaks ini digunakan untuk mengekspor data dari tabel ke file lain.
b. *Load* — Berfungsi untuk mengimpor data dari file lain ke tabel.
c. *Rename Table* — Berfungsi untuk mengganti nama tabel.

BAB 8 BAHASA *QUERY* TERAPAN LANJUTAN

8.1 *Join*

Join merupakan operasi yang digunakan untuk menggabungkan dua tabel atau lebih dengan hasil berupa gabungan dari kolom-kolom yang berasal dari tabel-tabel tersebut. Join juga digunakan untuk memilih data dari dua tabel atau lebih. Ada beberapa tipe *Join*, yaitu :

1. *Inner Join*
   Menggabungkan dua tabel dimana di antara dua tabel datanya bersesuaian.
2. *Left Join* atau *Left Outer Join*
   Menggabungkan dua tabel dimana di antara dua tabel datanya bersesuaian dan juga semua *record* pada tabel sebelah kiri.
3. *Right Join* atau *Right Outer Join*
   Menggabungkan dua tabel dimana di antara dua tabel datanya bersesuaian dan juga semua *record* pada tabel sebelah kanan.

8.2 Fungsi Agregat

Selain menampilkan nilai-nilai atribut yang ada di dalam tabel, sering pula ada kebutuhan untuk menampilkan data-data agregasi, seperti banyaknya *record*, total nilai suatu atribut, rata-rata nilai atribut, nilai atribut terbesar ataupun nilai atribut terkecil. Berikut yang termasuk fungsi agregasi:

1. *Count*, digunakan untuk menghitung jumlah atau untuk mendapatkan nilai banyaknya *record* hasil *query*.
2. *Sum* digunakan untuk menghitung total dari kolom yang mempunyai tipe data numerik.
3. *Avg* digunakan untuk menghitung rata-rata dari data-data dalam sebuah kolom.
4. *Min* digunakan untuk menghitung nilai minimal dalam sebuah kolom.
5. *Max* digunakan untuk menghitung nilai maksimum dalam sebuah kolom.

8.3 *Sub Query*

Sub query mempunyai arti suatu *query* berada di dalam *query*. Dengan menggunakan *sub query*, maka hasil dari *query* akan menjadi bagian dari *query* diatasnya. *Sub query* terletak di dalam klausa *where* atau *having*. Klausa *where*, digunakan untuk memilih baris-baris tertentu yang kemudian digunakan untuk *query*. Sedangkan pada klausa *having*, *sub query* digunakan untuk memilih kelompok baris yang kemudian digunakan oleh *query*.

Aturan membuat *sub query*:

1. Klausa Order By tidak boleh digunakan di *sub query*, Order By hanya dapat digunakan di pernyataan *select* luar.
2. Klausa *sub query select* harus berisi satu nama kolom tunggal atau ekspresi kecuali untuk *sub query* menggunakan kata kunci *exist*.

3. Secara *default* nama kolom di *sub query* mengacu ke nama tabel di klausa rrom dari *sub query* tersebut.
4. Saat *sub query* adalah salah satu dua operan dilibatkan di pembandingan, *sub query* harus muncul di sisi kanan perbandingan.

Penggunaan Any & All, berkaitan dengan *sub query*. Maka:

1. Jika *sub query* diawali kata kunci Any, syaratnya akan bernilai *true* jika dipenuhi sedikitnya satu nilai yang dihasilkan *sub query* tersebut atau dapat pula dikatakan menghasilkan *true* kalau paling tidak salah satu perbandingan dengan hasil subquery menghasilkan nilai *true*.
2. Jika *sub query* diawali kata kunci All, syarat hanya akan bernilai *true* jika dipenuhi semua nilai yang dihasilkan *sub query* itu.

Penggunaan Exist & Not Exist

Akan mengirim nilai *true* jika dan hanya jika terdapat sedikitnya satu baris di tabel hasil yang dikirim oleh *sub query* dan exist mengirim nilai False jika *sub query* mengirim tabel kosong. Untuk not exist kebalikan dari exist.

BAB 9 BASIS DATA TERDISTRIBUSI

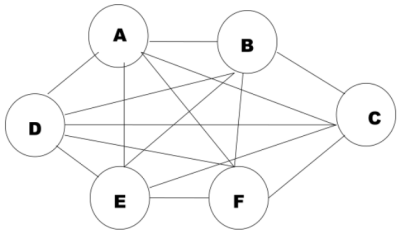
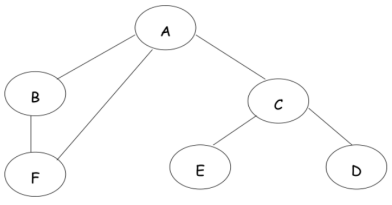
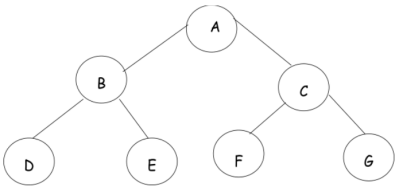
9.1 Definisi Basis Data Terdistribusi

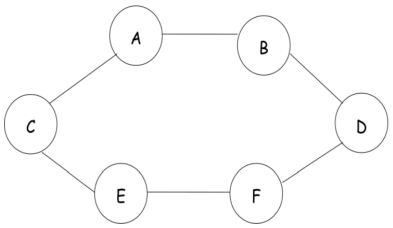
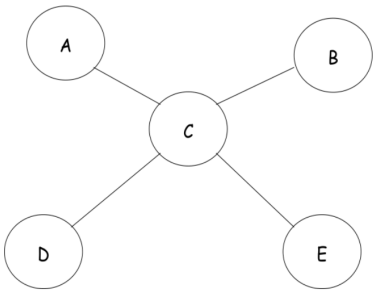
Yaitu kumpulan data yang digunakan bersama dan saling terhubung secara logik tetapi tersebar secara fisik pada suatu jaringan komputer. Karakteristik *database* terdistribusi:

1. Kumpulan data yang digunakan bersama secara logik tersebar pada sejumlah komputer yang berbeda.
2. Komputer yang dihubungkan menggunakan jaringan komunikasi
3. Data masing-masing situs dapat menangani aplikasi-aplikasi lokal secara otonom.
4. Data pada masing situs dibawah kendali satu DBMS.
5. Masing-masing DBMS berpartisipasi dalam sedikitnya satu aplikasi global.

Sistem basis data terdistribusi hanya mungkin dibangun dalam sebuah sistem jaringan komputer. Dalam sebuah sistem jaringan computer kita mengenal adanya topologi, yang akan menentukan bagaimana konfigurasi/ keterhubungan antara satu simpul jaringan (node/site) dengan simpulsimpul lainnya. Setiap simpul, dalam kaitannya dengan sistem basis data terdistribusi mewakili sebuah server, yang memiliki disk dengna sistem data sendiri (lokal). Setiap server ini juga membuat sebuah LAN (Local area Network) sendiri untuk mengamodasi sejumlah workstation dan sekaligus user lokal.

9.2 Topologi Distribusi Data

	<u>*Fully Connected Network*</u> Yaitu jenis topologi di mana setiap perangkat komputer dalam jaringan ini saling berhubungan. Apabila ada 5 komputer di jaringan ini, maka setiap komputer terhubung ke 4 komputer lainnya.
	<u>*Partially Connected Network*</u> Terdiri dari: *Tree-structured network*. Biaya instalasi dan komunikasi pada topologi jenis ini biasanya rendah. Namun, jika terjadi *failure link* atau *failure site* maka pengaksesan data menjadi terhambat dan mengakibatkan *availibilitas*/ketersediaan menjadi rendah.
	<u>*Tree Structured Network*</u> Merupakan penggabungan antara topologi bus dengan topologi star. Topologi tree memiliki satu kabel utama (*backbone*) yang menghubungkan beberapa hub dalam jaringan. Hub menghubungkan beberapa client, dan merupakan pusat kendali jaringan. Komunikasi data dari atau untuk client juga harus melalui hub.

	*Ring Network* Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk jalur melingkar membentuk cincin.
	*Star Network* Topologi bintang merupakan bentuk topologi jaringan yang berupa konvergensi dari node tengah ke setiap node atau pengguna. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah.

9.3 Fragmentasi Data

Fragmentasi merupakan sebuah proses pembagian atau pemetaan *database* dimana database dipecah-pecah berdasarkan kolom dan baris yang kemudian disimpan didalam site atau unit komputer yang berbeda dalam suatu jaringan data, sehingga memungkinkan untuk pengambilan keputusan terhadap data yang telah terbagi. Fragmentasi data merupakan langkah yang diambil untuk menyebarkan data dalam basis data terdistribusi. Fragmentasi diperlukan karena penggunaan dan efisiensi, parallelisme, keamanan.

Beberapa aturan yang harus didefinisikan saat fragmentasi:

a. Kondisi lengkap (*Completeness*) sebuah unit data yang masih dalam bagian dari relasi utama, maka data harus berada dalam satu fragmen.
b. Rekonstruksi (*Reconstruction*) sebuah relasi asli dapat dibuat kembali atau digabungkan kembali dari sebuah fragmen. Ketika telah dipecah-pecah, data masih memungkinkan untuk digabungkan kembali dengan tidak mengubah struktur data.
c. *Disjointness*, data didalam fragmen tidak boleh diikutkan dalam fragmen lain agar tidak terjadi *redundancy data*, kecuali untuk atribut *primary key* dalam fragmentasi vertikal.

Jenis Fragmentasi

1. Fragmentasi horizontal, terdiri dari tuple dari fragment global yang kemudian dipecah-pecah atau disekat menjadi beberapa sub-sets.
2. Fragmentasi vertikal, membagi atribut-atribut dari fragment global yang tersedia menjadi beberapa grup.
3. Fragmentasi campuran, menggunakan fragmentasi horizontal pada fragmentasi vertikal dan menggunakan fragmentasi vertikal pada fragmentasi horizontal.

BAB 10 LINGKUNGAN BASIS DATA

10.1 *Concurrency*

Ada 3 masalah yang disebabkan oleh *Concurrency* :

- Masalah kehilangan modifikasi (*Lost Update Problem*) masalah ini timbul jika dua transaksi mengakses item *database* yang sama yang mengakibatkan nilai dari database tersebut menjadi tidak benar.
- Masalah Modifikasi Sementara (*Uncommitted Update Problem*) masalah ini timbul jika transaksi membaca suatu *record* yang sudah dimodifikasi oleh transaksi lain tetapi belum terselesaikan (*uncommitted*), terdapat kemungkinan kalau transaksi tersebut dibatalkan (*rollback*).
- Masalah Analisa yang tidak konsisten (*Problem of inconsistency Analysis*) Masalah ini timbul jika sebuah transaksi membaca suatu nilai tetapi transaksi yang kedua mengupdate beberapa nilai tersebut selama eksekusi transaksi pertama.

10.2 *Locking*

Locking adalah salah satu mekanisme pengontrol konkurensi, konsep dasar ketika sebuah transaksi memerlukan jaminan kalau *record* yang diinginkan tidak akan berubah secara mendadak, maka diperlukan kunci untuk *record* tersebut. *Locking* berfungsi untuk menjaga *record* tersebut agar tidak dimodifikasi oleh transaksi lain. Jenis- Jenis *Lock*:

- *Share* (S)
  Kunci ini memungkinkan pengguna dan para pengguna konkuren yang lain dapat membaca *record* tetapi tidak mengubahnya.
- *Exclusive* (X)
  Kunci ini memungkinkan pengguna untuk membaca dan mengubah *record*. Sedangkan pengguna konkuren lain tidak diperbolehkan membaca ataupun mengubah *record* tersebut.

10.3 *Timestamping*

Timestamping adalah salah satu alternatif mekanisme kontrol konkurensi yang dapat menghilangkan masalah *deadlock*. Dua masalah yang timbul pada *Timestamping* :

- Suatu transaksi memerintahkan untuk membaca sebuah item yang sudah di *update* oleh transaksi yang belakangan.
- Suatu transaksi memerintahkan untuk menulis sebuah item yang nilainya sudah dibaca atau ditulis oleh transaksi yang belakangan.

BAB 11 LINGKUNGAN BASIS DATA LANJUTAN

11.1 *Crash and Recovery*

Crash adalah suatu *failure* atau kegagalan dari suatu sistem.

Penyebab kegagalan:

1. Disk Crash, yaitu informasi yang ada di disk akan hilang.
2. Power Failure yaitu informasi yang disimpan pada memori utama dan register akan hilang.
3. Software Error yaitu output yang dihasilkan tidak betul dan sistem databasenya sendiri akan memasuki suatu kondisi tidak konsisten berdasarkan jenis storage.
4. Volatile Storage, biasanya informasi yang terdapat pada volatile akan hilang, jika terjadi kerusakan sistem (system crash).
5. Non Volatile Storage, biasanya informasi yang terdapat pada non volatile storage tidak akan hilang jika terjadi kerusakan sistem.
6. Stable Storage, informasi yang terdapat dalam stable storage tidak pernah hilang.

Jenis-jenis kegagalan:

1. Logical Error, program tidak dapat lagi dilaksanakan disebabkan oleh kesalahan input, data tidak ditemukan, overflow.
2. System Error, sistem berada pada keadaan yang tidak diinginkan, seperti terjadi deadlock, sebagai akibat program tidak dapat dilanjutkan namun setelah beberapa selang waktu program dapat dijalankan kembali.
3. System Crash, kegagalan fungsi perangkat keras, menyebabkan hilangnya data pada volatile storage, tetapi data pada non volatile storage masih tetap ada.
4. Disk Failure, hilangnya data dari sebuah blok disk disebabkan oleh kerusakan head atau kesalahan pada waktu pengoperasian transfer data.

11.2 *Security*

Security adalah suatu proteksi data terhadap perusakan data dan pemakaian oleh pemakai yang tidak mempunyai ijin. Ada beberapa masalah security secara umum yaitu di dalam suatu perusahaan siapa yang diizinkan untuk mengakses suatu sistem. Kemudian bila sistem tersebut menggunakan password, bagaimana kerahasian dari password tersebut dan berapa lama password tersebut harus diganti. Serta di dalam pengontrolan hardware, apakah ada proteksi untuk penyimpanan data (data storage).

Tingkatan masalah security:

Phisical, berkaitan dengan pengamanan lokasi fisik database. → Man, berkaitan dengan wewenang user. → Sistem operasi, berkaitan dengan kemanan sistem operasi yang digunakan dalam jaringan. → Sistem database, sistem dapat mengatur hak akses user.

11.3 Pemberian Wewenang dan *View*

Konsep View adalah cara yang diberikan pada seorang pemakai untuk mendapatkan model database yang sesuai dengan kebutuhan perorangan. Database relational membuat pengamanan pada level:

1. Relasi, seorang pemakai diperbolehkan atau tidak mengakses langsung suatu relasi.
2. View, seorang pemakai diperbolehkan atau tidak mengakses data yang terdapat pada view.
3. Read Authorization, data dapat dibaca tapi tidak boleh dimodifikasi.
4. Insert Authorozation, pemakai boleh menambah data baru, tetapi tidak dapat memodifikasi data yang sudah ada.
5. Update Authorization, pemakai boleh memodifikasi tetapi tidak dapat menghapus data.
6. Delete Authorization, pemakai boleh menghapus data.
7. Index Authorization, pemakai boleh membuat atau menghapus index.
8. Resource Authorization, mengizinkan pembuatan relasi – relasi baru.
9. Alternation Authorization, mengizinkan penambahan atau penghapusan atribute dalam satu relasi.
10. Drop Authorization, pemakai boleh menghapus relasi yang ada.

11.4 *Integrity*

Berarti memeriksa keakuratan dan validasi data. Ada beberapa jenis integrity diantaranya:

1. Integrity Konstains, memberikan suatu sarana yang memungkinkan pengubahan database oleh pemakai berwenang sehingga tidak akan menyebabkan data inkonsistensi.
2. Integrity Rule (pada basis data relational), terbagi menjadi:
   Integrity Entity, contoh: tidak ada satu komponen kunci primer yang bernilai kosong (null).
   Integrity Referensi, suatu domain dapat dipakai sebagai kunci primer bila merupakan atribut tunggal pada domain yang bersangkutan.

DAFTAR PUSTAKA

Fathansyah. (2012). Basis Data. Bandung: Informatika.

Hariyanto, B. (2004). Sistem Manajemen Basis Data (Pemodelan, Perancangan dan Terapannya). Bandung: Informatika.

Indrajani. (2009). Sistem Basis Data Dalam Paket Five In One. Jakarta: PT Elex Media Komputindo.

Ladjamudin, A. B. Bin. (2004). Konsep Sistem Basis Data dan Implementasinya. Yogyakarta: Graha Ilmu.

Pahlevi, S. M. (2013). Tujuh Langkah Praktis Pembangunan Basis Data. Jakarta: PT Elex Media Komputindo.

Hartono, Jogiyanto. (2005). Basis Data. Jakarta: Salemba Empat.

Adi Nugroho. (2004). Konsep Pengembangan Sistem Basis Data. Bandung: Informatika.

Simarmata, J., & Paryudi, I. (2011). Basis Data. Yogyakarta: CV Andi Offset.

Connolly, T., & Begg, C. (2010). Database System: A Practical Approach to. United States: Pearson Education.

TENTANG PENULIS

Andi Priyolistiyanto, S.Kom., M.Kom. lahir di Pemalang, Jawa Tengah pada 7 April 1973.

Menempuh pendidikan S1 Teknik Informatika di IST "AKPRIND" Yogyakarta dan S2 Magister Sistem Informasi di Universitas Diponegoro Jawa Tengah.

Karir mengajarnya diawali pada tahun 2000 dengan menjadi instruktur/tenaga pengajar pada LP3I Semarang. Tahun 2004 menjadi Dosen Tetap pada Universitas Sultan Fatah Demak. Di tahun 2005 dipekerjakan sebagai Dosen PNS Kopertis Wilayah VI Jawa Tengah, di Universitas Muhammadiyah Magelang, sejak tahun 2013 mutasi ke IKIP PGRI SEMARANG, yang kemudian berubah nama menjadi Universitas Persatuan Guru Republik Indonesia Semarang (UPGRIS) sebagai Dosen PNS LLDIKTI Wilayah VI Jawa Tengah.

Kegiatan di luar kampus tercatat sebagai anggota pada organisasi profesi Persatuan Guru Republik Indonesia (PGRI), sebagai pemegang SRTI dari Persatuan Insinyur Indonesia (PII), relawan TIK, Advisor PT. Lentera Vokasee Indonesia, dan pengurus Yayasan Dharma Bakti Pancasila Pemalang.

Beberapa mata kuliah yang diampu, antara lain : Sistem Basis Data, Sistem Informasi, Sistem Pendukung Keputusan, Sistem Operasi, Data Mining, Mikrokontroller, Struktur Data, Pemrograman Java.

Denata Jayanti, akrab dipanggil Dena lahir di Banjarnegara, Jawa Tengah pada 29 Mei 2006.

Sedang menempuh pendidikan S1 dan merupakan mahasiswa Pendidikan Teknologi Informasi, Fakultas Pendidikan Matematika Ilmu Pengetahuan Alam dan Teknologi Informasi di Universitas Persatuan Guru Republik Indonesia Semarang (UPGRIS), Jawa Tengah.

Pada tahun 2018, lulus dari Sekolah Dasar Islam Terpadu Permata Hati, Banjarnegara. Kemudian melanjutkan pendidikan di Sekolah Menengah Pertama Negeri 2 Bawang, Banjarnegara dan lulus pada tahun 2021. Setelahnya menempuh pendidikan di Sekolah Menengah Kejuruan Negeri 1 Bawang, Banjarnegara jurusan Pengembangan Perangkat Lunak dan Gim hingga lulus pada tahun 2024.

Selain mengikuti kegiatan pembelajaran formal, juga menjadi pengurus Badan Eksekutif Mahasiswa Fakultas Pendidikan Matematika Ilmu Pengetahuan Alam dan Teknologi Informasi, anggota Unit Kegiatan Mahasiswa Kajian Ilmiah dan Penelitian Mahasiswa. Selain itu, pernah mengikuti beberapa organisasi juga kegiatan kepemimpinan dan kemanusiaan serta beberapa tentang bahasa dan seni. Pengalaman tersebut turut mendukung proses penulisan buku ini.

UNIVERSITAS PERSATUAN GURU
REPUBLIK INDONESIA SEMARANG

Pada buku ini berisi mengenai basis data, sistem basis data, model data, teknik normalisasi data, teknik normalisasi data lanjutan, bahasa query formal, bahasa query terapan, bahasa query terapan lanjutan, basis data terdistribusi, lingkungan basis data, lingkungan basis data lanjutan, serta penjelasan lain terkait dengan sistem basis data.