MODUL PEMBELAJARAN

SISTEM BASIS DATA

Disusun Oleh:

Ade Davy Wiranata, S.Kom.,M.Kom

NPD: D.20.1431

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PROF. DR. HAMKA
2019/2020

KATA PENGANTAR

Alhamdulillah, pertama penulis mengucapkan rasa syukur dan segala puji bagi Allah SWT yang telah melimpahkan segala Rahmat dan Karunianya, sehingga modul Sistem Basis Data ini dapat diselesaikan. Modul Sistem Basis Data ini diharapkan dapat mendukung mahasiswa dalam memahami matakuliah Sistem Basis Data. Modul ini dibuat berdasarkan sumber-sumber yang sudah banyak digunakan. Pada modul ini membahas mengenai konsep Sistem Basis Data secara umum. Modul ini membahas mengenai Konsep Dasar Basis Data, Komponen Basis Data, Relational & Perancangan Basis Data, Model Data, Entity Relationship Diagram (ERD), Normalisasi, Bahasa Query Formal dan Terapan, Basis Data Terdistribusi, dan Lingkup Basis Data. Akhir kata, penulis menyampaikan terimakasih yang tulus kepada pihak- pihak yang telah memberikan bantuan dan dukungannya sehingga penulis dapat menyelesaikan penulisan modul ini. Pada akhir kata, penulis memohon maaf yang sebesar-besarnya jika dalam penulisan modul ini masih banyak kekurangan dan kelemahannya. Penulis memohon adanya sumbangan ide, kritik dan saran untuk perbaikan penulisan modul ini supaya lebih baik ke depannya.

DAFTAR ISI

BAB 1
KONSEP SISTEM BASIS DATA

1.1 Definisi Sistem Basis Data

Basis Data 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. Melalui data seseorang dapat menganalisis, menggambarkan, atau menjelaskan suatu keadaan.

Basis data (*database*) dapat dibayangkan atau digambarkan sebagai sebuah almari arsip. Jika kita memiliki sebuah lemari arsip dan bertugas untuk mengelolanya, maka kemungkinan besar kita akan melakukan hal-hal seperti: memberi map pada kumpulan arsip, memberi penomoran dengan pola tertentu yang nilainya unik pada setiap map, lalu menempatkan arsip-arsip tersebut dengan urutan tertentu didalam lemari. Walupun hal-hal tersebut tidak seluruhnya dilakukan, paling tidak semua lemari arsip menerapkan suatu aturan tertentu bagaimana keseluruhan arsip-arsip tersebut disusun.

1.2 Prinsip dan Tujuan Basis Data

Prinsip basis data adalah pengaturan suatu arsip atau data. Tujuan utamanya yakni memberikan kemudahan dalam pengambilan kembali data/arsip yang pernah disimpan. Perbedaan basis data dengan lemari arsip hanya terletak pada media penyimpanan. Basis data menggunakan media penyimpanan elektronik.

1.3 Operasi Dasar Basis Data

Menurut Fathansyah, 2012 ada beberapa operasi-operasi dasar yang dapat kita lakukan berkenaan dengan basis data yaitu:

1. Pembuatan basis data baru (*create database*), yang lebih kepada pembuatan lemari arsip yang baru.
2. Penghapusan basis data (*drop database*), yang lebih kepada perusakan lemari arsip (sekaligus berserta isinya, jika ada).
3. Pembuatan tabel baru ke suatu basis data (*create table*), yang lebih kepada penambahan map arsip ke sebuah lemari arsip yang telah ada.

4. Penghapusan tabel dari suatu basis data (*drop table*), yang lebih kepada perusakan map arsip lama yang ada di sebuah lemari arsip.
5. Penambahan atau pengisian data baru ke sebuah tabel di sebuah basis data (*insert*), yang lebih kepada penambahan lembaran arsip ke sebuah map arsip.
6. Pengambilan data dari sebuah tabel (query), yang lebih kepada pencarian lembaran arsip dari sebuah map arsip.
7. Pengubahan data dari sebuah tabel (update), yang lebih kepada perbaikan isi lembaran arsip yang ada di sebuah map arsip.
8. Pengahpusan data dari sebuah tabel (delete), yang lebih kepada penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

1.4 Fungsi Basis Data

1. *Availability*

   Fungsi yang pertama 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.

BAB II
KOMPONEN SISTEM BASIS DATA

2.1 Komponen Sistem Basis Data

Sebuah basis data, secara lengkap akan terdapat komponen-komponen utama sebagai berikut: (Fathansyah,2012)

1. Perangkat Keras (*Hardware*)

   Perangkat keras biasanya terdapat dalam sebuah sistem basis data sebagai berikut:

   a) Komputer (hanya satu untuk sistem yang stand alone atau lebih dari satu untuk sistem jaringan).
   b) Memori sekunder yang *online* (*hardisk*).
   c) Memori sekunder yang offline (tape atau removable disk) untuk membantu dalam *backup* data.
   d) Media atau perangkat komunikasi (untuk sistem jaringan).

2. Sistem Operasi (*Operating System*)

   Sistem operasi merupakan program yang mengaktifkan sistem komputer, mengendalikan seluruh sumber daya (*resource*) dalam komputer dan melakukan operasi-operasi dasar dalam komputer (operasi *input*/*output*, pengelolaan file, dan lain-lain). Sejumlah sistem operasi yang banyak digunakan seperti : MS-DOS, MS-Windows, Linux (untuk komputer stand alone atau komputer *client* dalam sistem jaringan) atau Novel-*Netware*, MS-Windows server, Unix, Linux (untuk komputer server dalam sistem jaringan komputer).

3. Basis Data (Database)

   Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data berisi sejumlah objek basis data (seperti tabel, indeks, dan lain-lain).

4. Aplikasi Pengelola Basis Data (DBMS)

   Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (sistem) yang khusus. Perangkat lunak inilah (disebut DBMS) yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Ia juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan/konsistensi data, dan sebagainya.

5. Pengguna (*User*)

   Ada beberapa jenis atau tipe pengguna pada suatu sistem basis data , dibedakan berdasarkan cara pengguna berinteraksi ke sistem

   a) Pogrammer

      Pengguna yang berinteraksi dengan basis data melalui Data Manipulation Language (DML), yang disertakan (embedded) dalam program yang ditulis dalam bahasa pemrograman induk (seperti C, C++, Pascal, PHP, Java, dll).

   b) User Mahir (*Casual User*)

      Pemakai yang berinteraksi dengan sistem tanpa menulis modul program. Mereka menyatakan query (untuk akses data) dengan bahasa query yang telah disediakan oleh DBMS.

   c) User Umum (*End User/Naïve user*)

      Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (executable program) yang telah disediakan sebelumnya.

2.2 *Data Base 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, MS-SQL Server, Oracle, dBase, FoxBase, Rbase, Borland Paradox, dll.

2.3 Komponen DBMS

Menurut (Ladjamudin, 2004), komponen DBMS diklasifikasikan menjadi 5 yaitu:

1. Perangkat Keras

   DBMS membutuhkan perangkat keras untuk bisa berfungsi dengan baik. Perangkat keras tersebut dapat berupa Personal Computer (PC) yang stand alone maupun terkoneksi dalam suatu jaringan.

2. Perangkat Lunak

   Komponen perangkat lunak terdiri dari perangkat lunak DBMS itu sendiri bersama dengan sistem operasi yang digunakan dan juga perangkat lunak jaringan yang ada di dalamnya.

3. Data

   Data merupakan komponen yang paling penting dalam suatu DBMS. Basis data terdiri dari data-data operasional dan meta data. Struktur basis data biasa dinamakan sebagai skema. Data didalam basis data mempunyai sifat terpadu (integrated) yang berarti berkas-berkas data yang ada pada basis data saling terkait satu sama lain. Sifat yang kedua yaitu berbagi (*shared*) yang berarti data dapat diakses dan digunakan oleh sejumlah pengguna.

4. Prosedur

   Prosedur menghubungkan berbagai perintah, dan aturan-aturan yang akan menentukan rancangan dan penggunaan basis data.

5. Pengguna

   Pengguna akan berinteraksi dengan mesin (software dan hardware) melalui berbagai prosedur dan aturan-aturan formal yang berlaku. Pengguna dapat dikategorikan ke dalam empat golongan, yaitu:

   A. *Data Administrator* (DA) dan *Database Administrator* (DBA)

      1. *Data Administrator* (DA) akan bertanggung jawab dalam mengelola sumber daya data berupa:
         a) Pemeliharaan dan peremajaan suatu standarisasi formal yang berlaku.
         b) Perencanaan basis data.
         c) Pemeliharaan dan peremajaan suatu standarisasi formal yang berlaku.
      2. *Database Administrator* (DBA) akan bertanggung jawab dalam mengelola sumber daya fisik dari sistem basis data berupa:
            a) Sebagai media penghubung/perantara dengan *user*.
            b) Memiliki otoritas pengecekan dan menjalankan prosedur validasi.
            c) Mengontrol performasi data dan berhak memberi tanggapan atas usulan-usulan perubahan dan peremajaan data.
            d) Bertanggung jawab terhadap seluruh informasi yang berada didalam database.
            e) Bertanggung jawab terhadap strategi backup dan peremajaan data.
            f) Bertanggung jawab terhadap startegi pengaksesan data dan mengorganisasikan file didalam media penyimpanan.

B. *Database Designer*

Dalam suatu proyek basis data yang besar dan kompleks kita dapat membagi database designer menjadi dua yaitu:

1. *Logical Database Designer*

   Logical database designer bertugas mengidentifikasikan data seperti entitas, atribut, kardinalitas relasi dari data tersebut dan bagaimana data tersebut disimpan didalam media penyimpanan sekunder.

2. Physical *Database Designer*

   Physical database designer bertugas melakukan *mapping* terhadap model data logical ke dalam sekumpulan tabel atau relasi yang terintegrasi, merancang dan menentukan standarisasi keamanan data.

C. Programmer Aplikasi

Setelah basis data sudah diimplementasikan maka programmer akan membuat berbagai fungsi dalam sistem komputer.

D. *End User*

End user terdiri dari dua yaitu:

1. *Naïve User* (User Umum)

   Naïve user adalah mereka yang dapat mengenali DBMS. Pengguna tersebut mengakses basis data dengan menginput atau memasukkan perintah-perintah sederhana dari sistem menu.

2. *Sophisticated User* (User Mahir)

   Sophisticated User adalah pengguna yang dapat menggunakan bahasa query tingkat tinggi seperti SQL untuk menampilkan atau membuat operasi-operasi yang diinginkan.

2.4 Keuntungan dan Kekurangan DBMS

Keuntungan DBMS, sebagai berikut:

- Pengontrolan kerangkapan data
- Lebih banyak informasi dari jumlah data yang sama
- Peningkatan integrasi data
- Penegakan standar layanan
- Konsistensi data
- *Sharing* data
- Peningatan keamanan

Kekurangan DBMS, sebagai berikut:

- Kompleksitas
- Biaya DBMS
- Biaya konversi teknologi
- Dampak kegagalan yang lebih besar
- Ukuran
- Biaya Peangkat keras tambahan
- Performa

BAB III
RELATIONAL & PERANCANGAN BASIS DATA

3.1 Basis Data Relational

Menurut (Fathansyah, 2012), model basis data relasional sering pula disebut dengan model relasional atau basis data relasional. Model basis data ini diperkenalkan pertama kali oleh E.F. Codd. Model basis data menunjukkan suatu mekanisme yang digunakan untuk mengorganisasi data secara fisik dalam disk yang akan berdampak pula pada bagaimana kita mengelompokkan dan membentuk keseluruhan data yang berterkait dalam sistem yang sedang kita tinjau.

3.2 Komponen Penyusunan Basis Data

Adapun Komponen Penyusun Basis Data ada empat, sebagai berikut:

a. Tabel

   Tabel memiliki nama dan terdiri atas baris dan kolom. Tabel pada suatu basis data tidak boleh memiliki nama sama (unik). Tabel disebut juga dengan relation atau file.

b. Kolom/Atribu

   Kolom memiliki nama. Kolom yang terdapat dalam suatu tabel juga tidak boleh memiliki nama yang sama. Nama lain kolom adalah field atau atribut.

c. Baris/Tuple

   Baris pada sebuah tabel harus unik, dapat diletakkan dalam urutan bebas dan tidak mempengaruhi makna dari tabel. Baris disebut juga dengan record atau tuple.

d. Domain

   Domain adalah sekumpulan nilai-nilai yang dapat disimpan pada satu atau lebih kolom. Sebuah domain bisa dimiliki oleh satu kolom atau lebih, tetapi sebuah kolom hanya memiliki satu domain.

3.3 Kunci Relasional

Relational Keys atau kunci relasional adalah identifikasi satu atau sekelompok kolom yang nilainya dapat membedakan secara unik tuple-tuple yang ada. Menurut Pahlevi, 2013 terdapat5 Relational Keys, sebagai berikut:

a) *Superkey*

   Superkey satu atau kelompok kolom yang nilainya secara unik membedakan tuple-tuple pada suatu tabel.

b) *Candidate Key*

Candidate key adalah suatu *superkey* dimana tidak ada satupun himpunan bagian dari *superkey* tersebut menjadi *superkey* lagi, akan tetapi tidak semua superkey menjadi *candidate key*. *Candidate key* yang terdiri dari dua kolom atau lebih disebut sebagai *composite key*.

c) *Primary Key*

Primary key adalah satu *candidate key* yang dipilih (di antara *candidate key* lain) untuk membedakan tuple-tuple scara unik dalam suatu tabel.

d) *Alternate Key*

Alternate key adalah *candidate key* yang tidak dijadikan sebagai *primary key*.

e) *Foreign Key*

Foreign key adalah satu atau kelompok kolom yang nilainya sama atau terkait dengan *candidate key* pada tabel lain atau pada tabel yang sama.

3.4 Perancangan Basis Data

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 tiga, yaitu:

1) Perancangan basis data secara konsep, merupakan proses pembuatan data model dan tidak bergantung pada seluruh aspek fisik basis data.
2) Perancangan basis data secara logis, merupakan proses pembuatan data model berdasarkan data model tertentu, tetapi tidak bergantug pada DBMS tertentu dan implementasi fisik basis data.
3) Perancangan basis data secara fisik, merupakan 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 contraints sampai dengan langkah-lagkah 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

BAB IV
MODEL DATA

4.1 Definisi Model Data

Menurut (Fathansyah, 2012) 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.

Model data merupakan suatu cara untuk menjelaskan tentang data-data yang tersimpan dalam basis data dan bagaimana hubungan antar data tersebut untuk para pemakai secara logic. Model data juga dapat diartikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantic (makna) data dan batasan data.

4.2 Jenis-jenis Model Data

Model data terbagi menjadi dua yaitu:

A. Model Data Berbasis Objek

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, yaitu nama orang, benda, lokasi, kejadian (terdapat unsur waktu didalamnya). Atribut adalah relasi fungsional dari satu objek set ke objek set yang lain. Sedangkan, Relationship adalah hubungan alamiah yang terjadi antara entitas.

Model data berbasis objek memiliki beberapa bentuk, sebagai berikut:

1) Model Keterhubungan

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).

2) Model Berorientasi Object (*Object-Oriented Model*)

Penggambaran model berbasis objek menggunakan UML. UML digambarkan dengan 2 jenis yaitu Structural Diagram dan Behaviour Diagram.

a) Structural Diagram terdiri dari:

(1) Class Diagram, menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk mebangun sistem.

(2) Object Diagram, menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem.

(3) Component Diagram, dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.

(4) Deployment Diagram, menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi.

b) Behaviour Diagram

(1) Use case Diagram, merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat.

(2) Sequence Diagram, mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

(3) Communication Diagram, penyederhanaan dari diagram kolaborasi (Collaboration Diagram).

(4) Statechart Diagram, menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

(5) Activity Diagram, menggambarkan Workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

3) Model Data Semantik

Hampir sama dengan Entity Relationship model dimana relasi antara objek dasar tidak dinyatakan dengan simbol tetapi menggunakan kata-kata (Semantic).

B. Model Data Berdasarkan Record

Model ini berdasarkan pada record untuk menjelaskan kepada user tentang hubungan logic antar data dalam basis data. Perbedaan dengan model data berbasis objek adalah

pada record based data model disamping digunakan untuk menguraikan struktur logika keseluruhan dari suatu database, juga digunakan untuk menguraikan implementasi dari sistem database. Berikut ini adalah jenis-jenis model data berbasis record:

1) Model Relational

   Dimana data serta hubungan antar data direpresentasikan oleh sejumlah tabel dan masingmasing tabel terdiri dari beberapa kolom yang namanya unik.

2) Model Hirarki

   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/grup data elemen dan memiliki hubungan kardinalitas 1:1 dan 1:Many.

3) Model Jaringan

   Model dimana data dan hubungan antar data direpresentasikan dengan record dan links. Perbedaannya terletak pada susunan record dan linknya yaitu model jaringan menyusun record-record dalam bentuk graph dan menyatakan hubungan kardinalitas 1:1, 1:M dan M:M.

BAB V
ENTITY RELATIONSHIP DIAGRAM (ERD)

5.1 Definisi *Entity Relationship Diagram* (ERD)

Menurut (Sukamto & Shalahuddin, 2018), *Entity Relationship Diagram* (ERD) adalah bentuk paling awal dalam melakukan perancangan basis data relasional. jika menggunakan OODBMS maka perancangan ERD tidak diperlukan.

Menurut (Fathansyah, 2012), ERD adalah model entity relationship yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta 'dunia nyata' yang kita tinjau.

5.2 Komponen *Entity Relationship Diagram* (ERD)

Ada beberapa komponen pada ERD diantaranya:

1. Entitas

   Entitas adalah suatu kumpulan object atau sesuatu yang dapat dibedakan atau dapat diidentifikasikan secara unik. Dan kumpulan entitas yang sejenis disebut dengan entity set. Entity Set terbagi menjadi dua, yaitu:

   a. Strong entity set, yaitu entity set yang satu atau lebih atributnya digunakan oleh entity set lain sebagai key.
   b. Weak Entity set, yaitu entity set yang bergantung terhadap strong entity set.

2. Atribut

   Atribut adalah kumpulan elemen data yang membentuk suatu entitas. Adapun jenis-jenis atribut:

   a. Atribut kunci, adalah atribut yang digunakan untuk menentukan suatu entity secara unik
   b. Atribut simple, adalah atribut yang bernilai tungga.
   c. Atribut multi value, adalah atribut yang memiliki sekelompok nilai untuk setiap instan entity.
   d. Atribut composit, adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.
   e. Atribut derivatif, adalah suatu atribut yg dihasilkan dari atribut yang lain.

3. Relationship

   Relationship merupakan hubungan yang terjadi antara satu entitas atau lebih. Participation Constraint menjelaskan apakah keberadaan suatu entity tergantung pada hubungannya dengan entity lain. Terdapat dua macam participation constrain yaitu:

   a. Total participation constrain, adalah keberadaan suatu entity tergantung pada hubungannya dengan entity lain. Didalam diagram ER digambarkan dengan dua garis penghubung antar entity dan relationship.
   b. Partial participation, adalah keberadaan suatu entity tidak tergantung pada hubungan dengan entity lain. Didalam diagram ER digambarkan dengan satu garis penghubung.

4. Indicator Type

   Ada beberapa indicator type, diantaranya:

   a. Indicator Asosiatif Object, berfungsi sebagai suatu objek dan suatu relationship.
   b. Indicator Super Tipe Indicator tipe super tipe, terdiri dari suatu object dan satu subkategori atau lebih yang dihubungkan dengan satu relationship yang tidak bernama.

5.3 Kardinalitas

Kardinalitas adalah suatu relasi yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain.

Jenis-jenis kardinalitas sebagai berikut:

1. One to One (1:1)

   Kardinalitas ini mempunyai arti setiap entitas pad himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan plaing banyak dengan satu entitas pad himpunan entitas A.

2. One to Many (1:M) atau sebaliknya Many to One (M:1)

   One to Many berarti setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya sebaliknya , dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A. Many to One berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi

tidak sebaliknya sebaliknya , dimana setiap entitas pada himpunan entitas B berhubungan dengan banyak entitas pada himpunan entitas A.

3. Many to Many (M:N)

   Berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan demikian sebaliknya, dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

5.4 Tahap-tahap Membuat ERD

Berikut ini adalah tahapan dalam pembuatan *Entity Relationship Diagram* (ERD):

1. Identifikasi dan tetapkan seluruh himpunan entitas yang akan terlibat
2. Tentukan atribut key dari masing-masing himpunan entitas
3. Identifikasi dan tetapkan seluruh himpunan relasi antar himpunan entitas yang ada beserta foreign key-nya
4. Tentukan derajat/kardinalitas relasi untuk setiap himpunan relasi
5. Lengkapi himpunan entitas dan himpunan relasi dengan atribut bukan kunci.

BAB VI
NORMALISASI

6.1 Definisi Normalisasi

Perancangan basis data seringkali diasosiasikan dengan pembuatan model Entity Relationship (model ER), dimana kelompok-kelompok data dan relasi antar kelompok data tersebut diwujudkan dalam bentuk diagram. Hal itu tidak salah, karena model memang merupakan representasi nyata dalam sebuah perancangaa. Menurut (Fathansyah, 2012), Normalisasi merupakan cara pendekatan lain dalam membangun desain lojik 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. Namun demikian, dalam pelaksanaannya desain lojik basis data relasional didasari oleh transformasi secara hati-hati dari model ER ke bentuk fisik akan menghasilkan hasil yang mirip.

Menurut (Ladjamudin, 2004), menyampaikan beberapa definisi normalisasi, sebagai berikut :

1. Normalisasi adalah suatu proses memperbaiki/membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika.
2. Normalisasi adalah proses pengelompokkan data kedalam bentuk tabel atau relasi atau file untuk menyatakan entitas dan hubungan mereka sehingga terwujud satu bentuk database yang mudah untuk dimodifikasi.
3. Normalisasi dapat berguna dalam menjawab 2 pertanyaan mendasar yaitu : “Apa yang dimaksud dengan desain database logical?” dan “Apa yang dimaksud dengan desain databse fisikal yang baik? What is a physical good logical database design?
4. Normalisasi adalah suatu proses untuk mengidentifikasi tabel kelompok atribut yang memiliki ketergantungan yang sangat tinggi antara satu atribut dengan atribut yang lainnya.
5. Normalisasi bisa disebut juga sebagai proses pengelompokkan atribut-atribut dari suatu relasi sehingga membnetuk WELL STRUCTURED RELATION. WELL STRUCTURED RELATION adalah sebuah relasi yang jumlah kerangkapan datanya sedikit (minimum Amount Of Redundancy), serta memberikan kemungkinan bagi user untuk melakukan INSERT, DELETE, dan MODIFY terhadap baris-baris data.

6.2 Bentuk Normalisasi

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

1) Bentuk normal pertama (1NF)
2) Bentuk normal kedua (2NF)
3) Bentuk normal ketiga (3NF)
4) Bentuk normal Boyce-Codd (BCNF)
5) Bentuk normal keepat (4NF)
6) Bentuk Normal kelima (5NF)

6.3 Langkah-Langkah Pembuatan Normalisasi

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).

2) Bentuk Normal Kedua (2NF)

   Bentuk normal kedua didasari konsep full functional dependency (ketergantungan fungsional sepenuhnya) yang dapat didefinisikan sebagai berikut: Jika A dan B adalah atribut-atribut dari suatu relasi. B dikatakan full functional dependency terhadap A, jika B adalah tergantung fungsional terhadap A, tetapi tidak secara tepat memiliki ketergantungan fungsional dari subset atau himpunan bagian dari A.

3) Bentuk Normal Ketiga (3NF)

   Bentuk normal ketiga adalah suatu relasi dikatakan dalam bentuk normal ketiga (3NF) jika memiliki syarat: berada dalam bentuk normal kedua, setiap atribut bukan kunci haruslah tidak emiliki dependensi transitif (ketergantungan transitif) terhadap kunci primer, dengan kata lain suatu atribut bukan kunci tidak boleh memiliki ketergantungan fungsional terhadap atribut bukan kunci lainnya, selruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap primary key di relasi itu saja.

4) Bentuk Normal Boyce Codd (BCNF)

   Bentuk BCNF merupakan suatu relasi jika dan hanya jika semua penentu (determinan) adalah kunci kandidat (atribut yang bersifat unik) BCNF merupakan bentuk normal sebagai perbaikan terhadap 3NF. Suatu relasi yang memenuhi BCNF selalu memenuhi 3NF, tetapi tidak untuk sebaliknya.

5) Bentuk Normal Keempat (4NF)

Bentuk normal keempat berkaitan dengan sifat ketergantungan banyak nilai (Multivalued Depedency) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional.

6) Bentuk Normal Kelima (PNJF)

Bentuk normal kelima merupakan nama lain dari Project Join Normal Form (PNJF) yaitu berhubungan dengan ketergantungan relasi antar tabel (Join Dependency)

6.4 Keuntungan & Syarat Normalisasi

Adapun keuntungan dari normalisasi, yaitu :

1. Meminimalkan ukuran penyimpanan yang diperlukan untuk menyimpan data.
2. Meminimalkan resiko inkonsistensi data pada basis data
3. Meminimalkan kemungkinan anomali pembaruan
4. 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 berajlan 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 modification anomaly

  Desain database yang baik menyajikan suatu keyakinan bahwa ketika user melakukan perubahan dalam database, maka tidak terjadi hal yang tidak diinginkan.

BAB VII
NORMALISASI LANJUTAN

7.1 Definisi Anomaly

Menurut (Ladjamudin, 2004) anomaly merupakan penyimpangan-penyimpangan atau error atau inkonsistensi data yang terjadi pada saat dilakukan proses insert, delete maupun update dalam suatu basis data.

7.2 Jenis Anomaly

Terdapat 3 jenis Anomaly (penyimpangan), sebagai berikut:

1) Insertion Anomaly

   Merupakan error atau kesalahan yang terjadi sebagai akibat operasi insert record/tuple pada sebuah relation.

   Contoh ada matakuliah baru (CS-600) yang akan diajarkan, maka matakuliah tersebut tidak bisa di insert/disisipkan ke dalam relasi matakuliah sampai ada mahasiswa yang mengambil matakuliah tersebut.

2) Deletion Anomaly

   Merupakan error atau kesalahan yang terjadi sebagai akibat operasi delete record/tuple pada sebuah relation.

   Contoh mahasiswa dengan NIM: 0000123 memutuskan untuk batal ikut matakuliah dengan kode CS-400, karena ia merupakan satu-satunya peserta matakuliah tersebut, maka bila record tersebut dihapus akan berakibat hilangnya informasi matakuliah CS-400.

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.

   Contoh bila biaya kuliah untuk matakuliah CS-200 akan dinaikkan dari 75 menjadi 100, maka harus dilakukan beberapa kali modifikasi terhadap record-record mahasiswa yang mengambil matakuliah tersebut, agar data tetap konsisten.

7.3 Atribut dan Ketergantungan Fungsi

Beberapa konsep yang harus diketahui dalam Normalisasi:

1. Field/ Atribut Kunci

   Key Field / atribut kunci dalam database yaitu Super key, Candidate key, Primary key, Alternate key dan Foreign key.

2. Kebergantungan Fungsi.

   a) Ketergantungan Fungsional (Fungsional Dependent)

      Keterkaitan antar hubungan antara 2 atribut pada sebuah relasi. Dituliskan dengan cara: A -> B, yang berarti: atribut B fungsionality Dependent terhadap atribut A atau Isi (value) atribut A menentukan isi atribut B.

   b) Fully Functionaly 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: atribut Y pada relasi R adalah FFD pada atribut X pada relasi R jika Y FD pada X tida FD pada himpunan bagian dari X.

      Contoh

      PersonID, Project, Project_budget -> time_spent_byperson_ onProject (bukan FFD)

      PersonID, Project -> time_spent_byperson_onProject (FDD).

   c) Ketergantungan Partial

      Sebagian dari kunci dapat digunakan sebagai kunci utama.

   d) Ketergantungan Transitif

      Menjadi atribut biasa pada suatu relasi tetapi menjadi kunci pada relasi lain.

   e) Determinan Suatu atribut (field) atau gabungan atribut dimana beberapa atribut lain bergantung sepenuhnya pada atribut tersebut.

BAB VIII
BAHASA QUERY FORMAL

8.1 Pengertian Bahasa Query Formal

Menurut (Fathansyah, 2012), 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.

8.2 Kelompok Bahasa Query

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 diingunkan.
2. Bahasa non procedural

   User menunjukkan data atau informasi yang diinginkan tanpa menyatakan suatu cara atau prosedur tertentu untuk memperoleh data atau informasi tersebut.

Dua Dasar terbentuknya bahasa query menurut Ladjamudin, 2004:

1) Aljabar Relasional

   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.
2) Kalkulus Relasional

   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 subsitusi nilai argument dari fungsi tersebut. Jadi, bila semua argument dari sebuah fungsi disubsitusi dengan suatu nilai, maka fungsi tersebut menjadi suatu ekspresi yang disebut preposisi, yaitu suatu ekspresi yang hanya bernilai.

8.3 Operator Aljabar Relational

Operator pada aljabar relational dibagi menjadi 2 kelompok:

1. Operator dasar untuk fundamental operational Operator dasar terdiri dari : selection, projection, cartesian product, set difference dan union.
2. Operator tambahan untuk additional operasional Operator tambahan terdiri dari setvintersection, theta join, natural join dan division.

8.4 Studi Kasus

Kasus untuk mengerjakan perintah aljabar relasional

Database: Praktikum

Tabel Matakuliah

Kd_mk	Nama_mk	SKS	NIP
207	Logika & Algo	4	199910486
310	Struktur data	3	200109655
360	Sistem basis data	3	200209817
545	IMK	2	200209818
305	Pemrograman pascal	4	200703073
544	Desain grafis	2	200010490

Tabel Mahasiswa

NIM	Nama_mhs	Alamat	Jenis_kelamin
110509022	Hafidz	Depok	Laki-laki
110509102	Raffa	Depok	Laki-laki
110509000	Naia	Depok	Perempuan
110403088	Arif	Pondok Labu	Laki-laki
120609050	Leni	Kampung Melayu	Perempuan
120609028	Wahyuni	Tangerang	Perempuan
120609002	Aris	Depok	Laki-laki

Tabel Registrasi

Kd_mk	NIM
360	110509102
545	110403088
547	120609028

Tabel Dosen

NIP	Nama_dosen	Gaji
199910486	Billy	3000000
200109655	Mardiana	3000000
200209817	Indriyani	2000000
200209818	Suryani	2000000
200703073	Malau	3000000
200010490	Irfiani	2500000

- Operator Dasar

1) Selection (σ)

   Operasi selection menyeleksi tupel-tupel pada sebuah relation yang memenuhi predicate/syarat yang sudah ditentukan.

   Sintaks: σ predicate/syarat (tabel);

   Contoh

   Mencari tuple-tuple dari Mahasiswa yang memiliki jenis kelamin laki-laki

   Query

   σ Jenis_kelamin =“LAKI-LAKI” (Mahasiswa);

   Hasil query diatas

NIM	Nama_mhs	Alamat	Jenis_kelamin
110509022	Hafidz	Depok	Laki-laki
110509102	Raffa	Depok	Laki-laki
110403088	Arif	Pondok Labu	Laki-laki
120609002	Aris	Depok	Laki-laki

2) Projection (π)

Operator projection beroperasi pada sebuah relation, yaitu membentuk relation baru dengan meng-copy atribut-atribut dan domain-domain dari relation tersebut berdasarkan argumen-argumen pada operator tersebut.

Sintask: π A1,A2,A3…..An (nama tabel); dimana a adalah Atribut.

Contoh: Tampilkan nama beserta gaji dari dosen

Query π Nama_dosen, Gaji (Dosen);

Hasil dari query diatas

Nama_dosen	**Gaji**
Billy	3000000
Mardiana	3000000
Indriyani	2000000
Suryani	2000000
Malau	3000000
Irfiani	2500000

3) Cartersian Product (X)

Operator dengan dua relasi untuk menghasilkan tabel hasil perkalian kartesian. Operator ini merupaka binary operation yaitu operator yang beroperasi pada dua relasi. Operator Cartesian product menggunakan simbol X.

Sintaks: R1 X R2, dimana R adalah relasi.

Contoh

A = {1,2,3}

B = {5,7}

A X B = {(1,5),(1,7),(2,5),(2,7),(3,5),(3,7)}

4) Union (∪)

Operasi untuk menghasilkan gabungan tabel dengan syarat kedua tabel memiliki atribut yang sama yaitu domain atribut ke-i masing-masing tabel harus sama.

Operasi ini dapat dilaksanakan apabila kedua tabel mempunyai atribut yang sama sehingga jumlah komponen nya sama. Jadi yang akan kita gabungkan dari tabel-tabel tersebut, harus memiliki atribut yang sama dalam tabel tersebut.

Contoh

Tabel Dosen

Kd_dosen	Nama_dosen	Alamat_dosen	Kota
SY	Syamsudin, S.Si	Jl. Mawar II no.49	Tangerang
FS	Farida Syarif, M.Kom	Jl. Melati	Jakarta

Tabel Mahasiswa

NIM	Nama_mhs	Alamat_mhs	Kota	Tgl_lahir
1811001	Budiarto	Jl. Krakatau	Bogor	02-10-1998
1911002	Muhammad Ali	Jl. Perintis	Tangerang	20-06-1997

Jika dilakukan query sebagai berikut:

πKota (Mahasiswa) $\cup$ πKota(Dosen)

Hasil dari Query diatas

Kota
Tangerang
Jakarta
Bogor
Tangerang

5) Set Difference (-)

Operasi untuk mendapatkan tabel disuatu relasi tapi tidak ada di relasi lainnya. Dengan kata lain operator ini berfungsi untuk mengeliminasi entity atau record dari suatu tabel yang ada pada tabel yang lainnya.

Operasi ini dapat dilaksanakan apabila kedua tabel mempunyai atribut yang tidak sama yang akan ditampilkan.

Contoh

Tabel Kuliah_S1

Kd_kul	Nama_kul	SKS	Semester
IF110	Pemrograman I	3	1
IF221	Bahasa inggris	2	1
IF310	Struktur data	3	2
IF320	Basis data	4	3

IF411	Pemrograman II	3	3
IF423	Sistem pakar	2	4

Tabel Kuliah_D3

Kd_kul	Nama_kul	SKS	Semester
IF110	Pemrograman I	3	1
IF120	Aplikasi akuntansi	2	1
IF310	Struktur data	3	2
IF320	Basis data	4	3

Jika diberikan query sebagai berikut :

π nama_kul (Kuliah_S1) - π nama_kul (Kuliah_D3), maka hasilnya

Nama_kul
Bahasa inggris
Pemrograman II
Sistem pakar

- **Operator Tambahan**

Berikut ini adalah tabel-tabel yang akan dijadikan studi kasus dalam menjalankan operator tambahan.

Tabel R1

No_id	No_boat	Tanggal
22	101	01/06/20
58	103	10/10/20

Tabel S1

No_id	Nama	Rating	Umur
22	Sarah	7	25
31	Andri	8	22
58	Rafa	10	20

Tabel S2

No_id	Nama	Rating	Umur
28	Yuni	9	35
31	Andri	8	22
44	Shela	5	35
58	Rafa	10	20

1) Set Intersection (∩)

   Operasi untuk menghasilkan irisan dua tabel dengan syarat kedua tabel memiliki atribut yang sama, domain atribut ke-i kedua tabel tersebut sama.

   Contoh: S1 ∩ S2

   Hasilnya

No_id	Nama	Rating	Umur
58	Rafa	10	20

2) Theta Join

   Operasi yang menggabungkan operasi cartesian product dengan operasi selection dengan suatu kriteria.

   Contoh: S1 S1.No_id< R1.No_id R1

   Hasilnya

No_id	Nama	Rating	Umur	No_id	No_boat	Tanggal
22	Sarah	7	25	58	103	10/10/20
31	Andri	8	22	58	103	10/10/20

3) Natural Join

   Operasi menggabungkan operasi selection dan cartesian product dengan suatu kriteria pada kolom yang sama.

   Contoh: S1 θ No_id R1

   Hasilnya

No_id	Nama	Rating	Umur	No_id	No_boat	Tanggal
22	Sarah	7	25	58	103	10/10/20
58	Rafa	10	20	58	103	10/10/20

4) Division

Merupakan operasi pembagian atas tuple-tuple dari 2 relation.

Contoh

Tabel A

Sno	Pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2

Tabel B

Pno
P2

Hasil -> A/B

Sno
S1
S2

BAB IX
BAHASA QUERY TERAPAN (Structured Query Language (SQL))

9.1 Definisi SQL

Menurut Fathansyah, 2012 SQL merupakan bahasa query yang paling banyak digunakan oleh DBMS dan diterapkan dalam berbagai development tools dan program aplikasi ketika berinteraksi dengan Basis Data. Bahasa ini dibangun dengan dasar Aljabar Relational dan sedikit Kalkulus Relational.

Menurut Indrajani, 2009 SQL merupakan bahasa yang mudah dipelajari karena termasuk kedalam bahasa non procedural, cukup menspesifikasikan informasi apa yang dibutuhkan daripada bagaimana mendapatkannya.

9.2 SQL Sebagai Sub-bahasa

Menurut Ladjamudin, 2004 penyebuatan SQL sebagai bahasa query sebenarnya tidak tepat sebab kemampuan SQL tidak terbatas hanya untuk query (memperoleh data), tetapi juga mencakup kemampuan seperti:

a) Pendefinisian Struktur Data,
b) Pengubahan Data, dan
c) Pengaturan Sekuritas

Structured Query Language (SQL) dikatakan sebagai sub bahasa data karena SQL tidak mendukung persyaratan bahasa yang lengkap, sekalipun SQL dipakai untuk mengakses basis data. SQL tidak menyediakan hal-hal seperti pernyataan pengujian kondisi dan pernyataan pengulangan atau iterasi. Berikut subdivisi SQL:

a. *Data Definition Language* (DDL)

   Query-query ini digunakan untuk mendefinisikan struktur atau skema basis data.

b. *Data Manipulation Language* (DML)

   Query-query ini digunakan untuk manajemen data dalam basis data.

c. Data Access (Data Control Language / DCL)

   Query-query ini digunakan uuntuk mengendalikan pengaksesan data.

9.3 Pengelompokan SQL

9.3.1 *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. Yang termasuk dalam kelompok DDL ini adalah CREATE, ALTER dan DROP.(Ladjamudin, 2004).

1) CREATE
   a) Pembuatan Database

      Sintaks : CREATE DATABASE nama_database

      Contoh CREATE DATABASE praktikum

   b) Pembuatan Tabel

      Sintaks : CREATE TABLE nama_table (nama_kolom1 tipe_data_kolom1, nama_kolom2 tipe_data_kolom2,….)

      Keterangan :

      a. Nama tabel merupakan nama tabel yang baru, panjangnya tidak dapat lebih dari 8 karakter, tidak memakai spasi, berisi huruf maupun angka.
      b. Nama Kolom adalah nama untuk kolom yang baru, panjangnya tidak dapat lebih dari 10 karakter, tidak memiliki spasi, berisi huruf dan angka.
      c. Tipe Data adalah jenis data yang nilainya akan dimasukkan dalam kolom yang telah ditentukan.
      d. Setiap kolom pada pendefinisian tabel dapat dilengkapi dengan UNIQUE, NULL, NOT NULL dan NOT UNIQUE.

   c) Pembuatan Index

      Indeks biasa diciptakan dengan tujuan sebagai berikut :

      1. Indeks dapat meningkatkan kinerja
      2. Indeks menjamin bahwa suatu kolom bersifat unik.

         Sintaks : CREATE [UNIQUE] INDEX nama_index ON nama_table (nama_kolom);

         Contoh CREATE UNIQUE INDEX MHSIDX ON Mahasiswa(NIM)

   d) Pembuatan View

      Pembuatan View lebih bersifat memanipulasi data daripada pernyataan definisi data, harus menggunakan SELECT untuk mengerjakan perintah ini. Sintaks : CREATE VIEW nama_view [(nama_kolom1,….)] AS SELECT statement [WITH CHECK OPTION]; Contoh : Buat view dengan nama MATKULVIEW

yang berisi semua data matakuliah. Query : CREATE VIEW MATKULVIEW AS SELECT * FROM matkul;

2) DROP
   a) DROP DATABASE (menghapus database)
      Sintaks: DROP DATABASE nama_database;
      Contoh Menghapus Database KAMPUS
      Query : DROP DATABASE KAMPUS;
   b) DROP TABLE (menghapus tabel)
      Sintaks: DROP TABLE nama_table;
      Contoh Mengahapus Tabel MHS
      Query : DROP TABLE MHS
   c) DROP INDEX (menghapus index)
      Sintaks DROP INDEX nama_index;
      Contoh Menghapus Index MHSIDX
      Query : DROP INDEX MHSIDX;
   d) DROP View(Menghapus View)
      Sintaks Drop View nama_view;
      Contoh Menghapus View MHSVIEW
      Query : DROP VIEW MHSVIEW

3) ALTER TABLE

   Digunakan untuk merubah struktur dari tabel yang telah dibuat dalam database. Perubahan struktur yang dapat dilakukan adalah menambah kolom baru, merubah nama kolom, merubah tipe data, menambah kunci, mengahapus kolom yang ada.

   Sintaks :

   ALTER TABLE nama_tabel
   a) ADD nama_kolom jenis_kolom [FIRST | AFTER nama_kolom]
      Ket: untuk menambah kolom baru.
   b) CHANGE [COLUMN] oldnama newnama
      Ket: untuk merubah nama kolom.
   c) MODIFY nama_kolom jenis kolom, …
      Ket: untuk merubah tipe data kolom.
   d) DROP nama_kolom

Ket: untuk menghapus nama kolom

e) RENAME newnama_tabel.

Ket: untuk mengganti nama tabel.

9.3.2 *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:

1) INSERT, mempunyai fungsi untuk penambahan record baru kedalam sebuah tabel. Sintaknya INSERT INTO Nama_tabel [(nama_kolom1,…)] values (nilai_atribut1, …);
2) DELETE, mempunyai fungsi untuk menghapus record dari sebuah tabel. Sintaknya DELETE FROM nama_table WHERE kondisi;
3) UPDATE, mempunyai fungsi untuk mengubah nilai atribut pada suatu record dari sebuah tabel. Sintaknya UPDATE nama_tabel SET nama_kolom = value_1 WHERE kondisi;
4) SELECT, mempunyai fungsi untuk menampilkan isi tabel. Sintaknya SELECT [DISTINCT | ALL] nama_kolom FROM nama_tabel [WHERE condition] [GROUP BY column_list] [HAVING condition] [ORDER BY column_list [ASC | DESC]];

9.3.3 *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 dalam kelompok DCL atau Data Access yaitu:

1) GRANT, mempunyai fungsi untuk memberikan hak akses. Sintaksnya GRANT hak_akses ON nama_db TO nama_pemakai [IDENTIFIED BY] [PASSWORD] ‘Password’ [WITH GRANT OPTION]; atau GRANT hak_akses ON [nama_db]nama_tabel TO nama_pemakai [IDENTIFIED BY] [PASSWORD] ‘Password’ [WITH GRANT OPTION];

2) REVOKE, mempunyai fungsi untuk mencabut kembali hak akses yang sudah diberikan. Sintaksnya REVOKE hak_akses ON nama_db FROM nama_pemakai ; atau REVOKE hak_akses ON nama_tabel FROM nama_pemakai;

9.3.4 *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. Sintaknya RECOVER TABLE nama_tabel;

9.3.5 *Data Auxiliary*

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

1) SELECT … INTO OUTFILE ‘filename’, berfungsi untuk mengekspor data dari tabel ke file lain. Sintaknya SELECT … INTO OUTFILE ‘Nama File’ [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']];
2) LOAD, berfungsi untuk mengimpor data dari file lain ke tabel. Sintaknya LOAD DATA INFILE “ nama_path” INTO TABLE nama_tabel [nama_kolom] ; [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']];
3) RENAME TABLE, berfungsi untuk mengganti nama tabel. Sintaknya RENAME TABLE OldnamaTabel TO NewNamaTabel;
   Contoh RENAME TABLE matakuliah TO matkul;

BAB X
BAHASA QUERY TERAPAN LANJUTAN (Structured Query Language (SQL))

10.1 Pengertian Join

Menurut (Ladjamudin, 2004) menyampaikan bahwa Join merupakan operasi yang digunakan untuk menggabungkan dua tabel atau lebih dengan hasil berupa gabungan dari kolom-kolom yang berasal dari tabel-tabel tersebut

10.2 Jenis-jenis Join

Ada beberapa tipe Join, yaitu :

a. INNER JOIN

   Inner join adalah menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian.

b. LEFT JOIN atau LEFT OUTER JOIN

   Left Join adalah menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kiri.

c. RIGHT JOIN atau RIGHT OUTER JOIN

   Right join adalah menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kanan.

10.3 Fungsi Agregasi (Agregat)

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

a. COUNT

   Berfungsi untuk menghitung jumlah atau untuk mendapatkan nilai banyaknya record hasil query.

   Contoh : Menghitung jumlah record mahasiswa dari tabel.

   Query : SELECT COUNT(*) FROM mahasiswa;

b. SUM

   Berfungsi untuk menghitung total dari kolom yang mempunyai tipe data numerik.

   Contoh : Menghitung total sks dari tabel matkul;

   Query : SELECT SUM(SKS) AS ‘TOTAL SKS’ FROM matkul;

c. AVG

Berfungsi untuk menghitung rata-rata dari datadata dalam sebuah kolom.

Contoh : Menghitung Nilai Rata-rata Nilai Final dari Tabel Nilai

Query : SELECT AVG(FINAL) AS ‘FINAL’ FROM Nilai;

d. MIN

Berfungsi untuk menghitung nilai minimal dalam sebuah kolom.

Contoh : Menghitung Minimal Nilai Final dari Tabel Nilai

Query : SELECT MIN(FINAL) FROM Nilai;

e. MAX

Berfungsi untuk menghitung nilai maksimum dalam sebuah kolom
Contoh : Menghitung maksimal nilai MID dari tabel Nilai
Query : SELECT MAX(MID) FROM Nilai;

10.4 Sub Query

10.4.1 Definisi Sub Query

Subquery mempunyai arti suatu query berada di dalam query. Dengan menggunakan subquery, maka hasil dari query akan menjadi bagian dari query diatasnya. Subquery terletak didalam klausa WHERE atau HAVING. Klausa WHERE,digunakan untuk memilih baris-baris tertentu ang kemudian digunakan untuk query. Sedangkan pada klausa HAVING, subquery digunakan untuk memilih kelompok baris yang kemudian digunakan oleh query.

Contoh: Menampilkan daftar kode pengarang dan nama pengarang (berdasarkan tabel Pengarang) yang kode pengarangnya tercantum pada tabel Buku.

Query: SELECT KD_PENG, NAMA FROM Pengarang WHERE KD_PENG IN (SELECT KD_PENG FROM BUKU);

Pada query tersebut, SELECT KD_PENG FROM BUKU disebut dengan subquery, sedangkan SELECT KD_PENG, NAMA berkedudukan sebagai query

10.4.2 Aturan Membuat Sub Query

Ada beberapa aturan-aturan dalam membuat subquery:

1) Klausa Order By tidak boleh digunakan di subquery, Order By hanya dapat digunakan di pernyataan Select luar.
2) Klausa subquery Select harus berisi satu nama kolom tunggal atau ekspresi kecuali untuk subquery-subquery menggunakan kata kunci EXIST
3) Secara default nama kolom di subquery mengacu ke nama tabel di klausa FROM dari subquery tersebut.
4) Saat subquery adalah salah satu dua operan dilibatkan di pembandingan, subquery harus muncul di sisi kanan pembandingan.

10.4.3 Penggunaan Any & All

Penggunaan Any dan All Digunakan berkaitan dengan subquery. Jika subquery diawali kata kunci Any, syaratnya akan bernilai *True* jika dipenuhi sedikitnya satu nilai yang dihasilkan subquery tersebut atau dapat pula dikatakan menghasilkan *True* kalau paling tidak salah satu perbandingan dengan hasil subquery menghasilkan nilai *True*.

10.4.4 Penggunaan Exist & Not Exist

Penggunaan Exist dan Not Exist akan mengirim nilai *True* jika dan hanya jika terdapat sedikitnya satu baris di tabel hasil yang dikirim oleh subquery dan Exist mengirim nilai False jika subquery mengirim tabel kosong. Untuk Not Exist kebalikan dari Exist.

BAB XI
BASIS DATA TERDISTRIBUSI

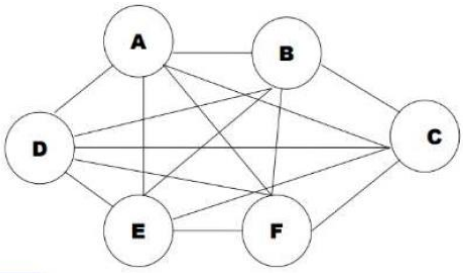
11.1 Pengertian Basis Data Terdistribusi

11.2 Topologi Distribusi Data

11.3 Bentuk-Bentuk Topologi Terdistribusi

11.3.1 *Fully Connected Network*

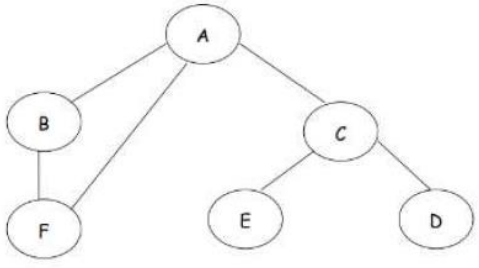
Fully connected yaitu jenis topologi di mana setiap perangkat komputer dalam jaringan ini saling berhubungan. Apabila ada 5 komputer di jaringan ini, maka setiap komputer terhubung ke 4 komputer lainnya.



Gambar 11.1 *Fully Connected Network*

11.3.2 *Partialy Conncted Network*

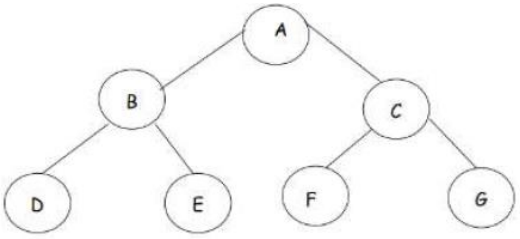
Partially Connected Network terdiri dari: *Tree-structured network*. Biaya instalasi dan komunikasi pada topologi jenis ini biasanya rendah. Namun, jika terjadi failure link atau failure site maka pengaksesan data menjadi terhambat dan mengakibatkan availibilitas/ketersediaan menjadi rendah.



Gambar 11.2 *Partialy Conncted Network*

11.3.3 *Tree Structured Network*

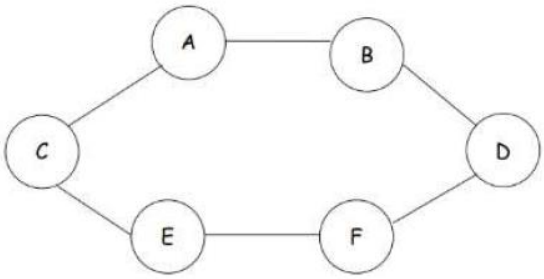
Topologi tree merupakan penggabungan antara topologi bus dengan topologi star. Topologi tree memiliki satu kabel utama (*backbone*) yang menghubungkan beberapa hub dalam jaringan. Hub menghubungkan beberapa client, dan merupakan pusat kendali jaringan. Komunikasi data dari atau untuk client juga harus melalui hub.



Gambar 11.3 *Tree Structured Network*

11.3.4 *Ring Network*

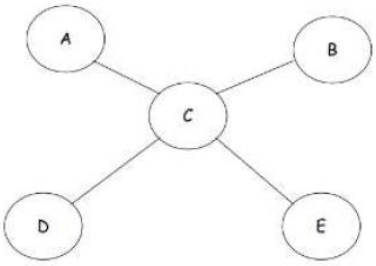
Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk jalur melingkar membentuk cincin.



Gambar 11.4 *Ring Network*

11.3.5 *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.



Gambar 11.5 *Star Network*

BAB XII
BASIS DATA TERDISTRIBUSI LANJUTAN

12.1 Keuntungan dan Kerugian Basis Data Terdistribusi

Perkembangan sistem basis data terdistribusi akan merefleksikan dan mencerminkan struktur organisasional data akan dapat diakses dengan baik oleh seluruh unit dan menyimpan data-data penting dan sering digunakan ke tempat-tempat yang paling sering digunakan dan mudah ditemukan, serta meningkatkan kemampuan data untuk digunakan secara bersama-sama berikut efisiensi pengaksesannya. Basis data terdistribusi yaitu kumpulan data logic yang saling berhubungan, secara fisik trdistribusi dalam jaringan komputer, yang tidak tergantung dari program aplikasi, dan dapat digunakan oleh banyak aplikasi sekarang maupun pada masa yang akan datang. DBMS teristribusi adalah sistem *software* yang memungkinkan ditatanya suatu basis data terdistribusi bagi setiap pemakai (*user*).

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. (Fathansyah, 2012).

12.2 Pengertian Fragmentasi Data

Fragmentasi adalah 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.

12.3 Alasan Dibutuhkan Fragmentasi

a) Penggunaan

   Umumnya, aplikasi-aplikasi beroperasi dengan terhadap suatu view tertentu bukan seluruh relasi. Dengan demikian untuk melakukan distribusi data maka beralasan untuk bekerja dengan suatu subset relasi (fragmen).

b) Efisiensi

Data yang disimpan dekat dengan aplikasi yang sering menggunakannya. Data yang tidak diperlukan oleh aplikasi local tidak disimpan di situs itu.

c) Paralleslisme

Dengan fragmen-fragmen sebagai unit distribusi, transaksi dapat dibagi menjadi beberapa subquery yang beroperasi pada fragmen-fragmen itu. Fragmentasi harus meningkatkan derajat konkurensi atau paralelisme sistem.

d) Keamanan

Data yang tidak diperlukan oleh aplikasi local tidak disimpan di situs itu. Dengan cara ini, data tidak tersedia untuk pemakai-pemakai yang tidak diotorisasi.

12.4 Aturan Fragmentasi

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. Ketika ada relasi, pembagian datanya harus menjadi satu kesatuan dengan relasinya.

b. Rekontruksi (*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 vertical.

BAB XIII
LINGKUNGAN BASIS DATA

13.1 Jenis-Jenis Fragmentasi

a) Fragmentasi horizontal

Terdiri dari tuple dari fragment global yang kemudian dipecah-pecah atau disekat menjadi beberapa sub-sets. Fragmentasi horizontal berisikan tuple2 yang dipartisikan dari sebuah relasi global ke dalam sejumlah subset r1,r2…rn, tiap2 subset berisi tuple dari r, setiap tuple dari r harus memiliki satu fragment sehingga relasi yang asli dapat disusun kembali.

b) Fragmentasi vertikal

Membagi atribut-atribut dari fragment global yang tersedi a menjadi beberapa grup. Penambahan tuple-id didalam fragmentasi vertikal. Fragmentasi vertikal disempurnakan dengan menambahkan sebuah atribut yang disebut tuple identifier(tuple-id) ke dalam skema r. Sebuah tuple-id adalah sebuah alamat logik dari sebuah tuple. Setiap tuple didalam r harus memiliki sebuah alamat yang unuk, yaitu atribute tuple-id sebagai kunci penambahan skema.

c) Fragmentasi campuran

Relasi r (global) dibagi2 kedalam sejumlah relasi fragment r1, r2..rn. Tiap2 fragmentasi diperoleh sebagai hasil baik dari skema fragmentasi horizontal ataupun fragmentasi vertikal di relasi r atau dari sebuah fragmentasi r yang diperoleh sebelumnya.

13.2 Konkurensi

Ada 3 masalah yang disebabkan oleh *Concurrency* :

1. 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.
2. Masalah Modifikasi Sementara (*Uncommited Update Problem*) Masalah ini timbul jika transaksi membaca suatu record yang sudah dimodifikasi oleh transaksi lain tetapi belum terselesaikan (uncommited), terdapat kemungkinan kalau transaksi tersebut dibatalkan (rollback).

3. 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.

13.3 *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 :

1. Share (S)

   Kunci ini memungkinkan pengguna dan para pengguna konkuren yang lain dapat membaca record tetapi tidak mengubahnya.
2. Exclusive (X)

   Kunci ini memungkinkan pengguna untuk membaca dan mengubah record. Sedangkan pengguna konkuren lain tidak diperbolehkan membaca ataupun mengubah record tersebut.

13.4 *Timestamping*

Timestamping adalah salah satu alternatif mekanisme kontrol konkurensi yang dapat menghilangkan masalah dead lock.

Dua masalah yang timbul pada Timestamping :

1. Suatu transaksi memerintahkan untuk membaca sebuah item yang sudah di update oleh transaksi yang belakangan.
2. Suatu transaksi memerintahkan untuk menulis sebuah item yan nilainya sudah dibaca atau ditulis oleh transaksi yang belakangan

BAB XIV
LINGKUNGAN BASIS DATA LANJUTAN

14.1 *Crash* & *Recovery*

Crash adalah suatu failure atau kegagalan dari suatu sistem. Penyebab kegagalan yaitu:

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) contoh: RAM
5. *Non Volatile Storage*, biasanya informasi yang terdapat pada non volatile strorage tidak akan hilang jika terjadi kerusakan sistem contoh: ROM
6. *Stable Storage*, informasi yang terdapat dalam stable storage tidak pernah hilang. contoh: Harddisk RAID Jenis-jenis kegagalan
7. *Logical Error*, program tidak dapat lagi dilaksanakan disebabkan oleh kesalahan input, data tidak ditemukan, over flow
8. *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.
9. *System Crash*, kegagalan fungsi perangkat keras, menyebabkan hilangnya data pada volatile storage, tetapi data pada non volatile storage masih tetap ada.
10. **Disk Failure**, hilangnya data dari sebuah blok disk disebabkan oleh kerusakan head atau kesalahan pada waktu pengoperasian transfer data.

14.2 *Security*

Security adalah suatu proteksi data terhadap perusakan data dan pemakaian oleh pemakai yang tidak mempunyai ijin. Ada beberapa masalah security secara umum.

1. Di dalam suatu perusahaan siapa yang diijinkan untuk mengakses suatu sistem.
2. Bila sistem tersebut menggunakan password, bagaimana kerahasian dari password tersebut dan berapa lama password tersebut harus diganti.

3. Di dalam pengontrolan hardware, apakah ada proteksi untuk penyimpanan data(data storage).

Kategori penyalahgunaan database:

1. Katagori yang tidak disengaja Contoh: Anomali yang disebabkan oleh pendistribusian data pada beberapa komputer.
2. Katagori yang disengaja Contoh: Insert, Delete & Update oleh pihak yang tidak berwenang.

Tingkatan masalah security:

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

14.3 Pemberian Wewenang & *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

14.4 Integrity

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. Informatika.

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

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

Sukamto, R. A., & Shalahuddin, M. (2018). Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek. Informatika.

LAMPIRAN

UNIVERSITAS MUHAMMADIYAH PROF. DR. HAMKA
FAKULTAS TEKNIK
Jl. Tanah Merdeka No. 6, Kp. Rambutan, Ps. Rebo, Jakarta Timur. Telp. (021) 8400941; Fax. (021) 87782739
Website : www.ft.uhamka.ac.id; Email : ft@uhamka.ac.id

SURAT TUGAS
MELAKUKAN KEGIATAN PENDIDIKAN DAN PENGAJARAN
Nomor : 289 /F.03.03/2020

Bismillahirrahmanirrahim
Yang bertanda tangan dibawah ini

Nama	D.11.0709/0323056403
NPD/NIDN	Penata /III.C
Pangkat/Jabatan Akademik	Lektor
Jabatan	Dekan Fakultas Teknik UHAMKA
Unit Kerja	0323056403

Memberikan tugas mengajar pada **Semester Gasal Tahun Akademik 2020 /2021** kepada :

Nama	**Ade Davi Wiranata, S.Kom., M.Kom**
NPD/NIDN	0325119302
Pangkat/Jabatan Akademik	Penata Muda Tingkat 1 /III.B
Jabatan Fungsional	-
Unit Kerja	Fakultas Teknik UHAMKA

Untuk mata kuliah-matakuliah terjadwal sebagai berikut :

Hari	**Waktu**	**Mata Kuliah**	**Prodi**	**KLS**	**SKS**
Selasa	14.40-16.20	Peng.Teknologi Informasi	Teknik Informatika	1E	2
Selasa	18.15-19.55	Peng.Teknologi Informasi	Teknik Informatika	1H	2
Kamis	09.30-12.00	Sistem Basis Data	Teknik Informatika	3E	3
Sabtu	13.00-15.30	Sistem Basis Data	Teknik Informatika	3F	3
Kamis	13.00-14.40	Praktik Sistem Basis Data	Teknik Informatika	3E	1
Jumat	07.50-09.30	Praktik Sistem Basis Data	Teknik Informatika	3F	1
Rabu	07.50-10.20	Logika Informatika	Teknik Informatika	1C	3
Sabtu	07.50-10.20	Logika Informatika	Teknik Informatika	1F	3
		Jumlah SKS			**18**

Demikian surat tugas ini diberikan kepada yang bersangkutan untuk dilaksanakan dengan penuh amanh dan tanggung jawab.

Jakarta, 27 September 2020

Dekan,

Dr. Sugema, ST., M.Kom.

Tembusan :
1. Rektor UHAMKA
2. Wakil Rektor 1 dan II
3. Arsip

UNIVERSITAS MUHAMMADIYAH PROF. DR. HAMKA
FAKULTAS TEKNIK
Jl. Tanah Merdeka No. 6, Kp. Rambutan, Ps. Rebo, Jakarta Timur. Telp. (021) 8400941; Fax. (021) 87782739
Website : www.ft.uhamka.ac.id; Email : ft@uhamka.ac.id

SURAT TUGAS
MELAKUKAN KEGIATAN PENDIDIKAN DAN PENGAJARAN
Nomor : 320 /F.03.03/2021

Bismillahirrahmanirrahim
Yang bertanda tangan dibawah ini

Nama	D.11.0709/0323056403
NPD/NIDN	Penata /III.C
Pangkat/Jabatan Akademik	Lektor
Jabatan	Fakultas Teknik UHAMKA
Unit Kerja	0323056403

Memberikan tugas mengajar pada **Semester Genap Tahun Akademik 2020 /2021** kepada :

Nama	**Ade Davi Wiranata, S.Kom., M.Kom**
NPD/NIDN	0325119302
Pangkat/Jabatan Akademik	Penata Muda Tingkat 1 /III.B
Jabatan Fungsional	-
Unit Kerja	Fakultas Teknik UHAMKA

Untuk mata kuliah-matakuliah terjadwal sebagai berikut :

Hari	**Waktu**	**Mata Kuliah**	**Prodi**	**KLS**	**SKS**
Senin	18.15-19.55	Pemrograman Berbasis Objek	Teknik Informatika	4C	2
Selasa	16.20-18.00	Praktikum Berbasis Objek	Teknik Informatika	4C	1
Selasa	14.40-16.20	Struktur Data	Teknik Informatika	2D	2
Sabtu	13.00-14.40	Struktur Data	Teknik Informatika	2G	2
Kamis	13.00-14.40	Praktikum Strukur Data	Teknik Informatika	2D	1
Sabtu	14.40-16.20	Praktikum Strukur Data	Teknik Informatika	2G	1
Selasa	07.50.10.20	Teknik Perancangan Sistem	Teknik Informatika	4F	3
		Jumlah SKS			**12**

Demikian surat tugas ini diberikan kepada yang bersangkutan untuk dilaksanakan dengan penuh amanh dan tanggung jawab.

Jakarta, 5 Maret 2021

Dekan,

Dr. Sugema, ST., M.Kom.

Tembusan :
1. Rektor UHAMKA
2. Wakil Rektor 1 dan II
3. Arsip