PENDIDIKAN KOMPUTER

MODUL PEMBELAJARAN

SISTEM BASIS DATA

Dra. Suriaty, M.Pd
Andi Rustandi, S.Kom., M.TI., M.H

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 (Database), DBMS & Perancangan Basis Data, Model Data, Entity Relationship Diagram (ERD), Normalisasi, Bahasa Query Formal, Bahasa Query Terapan, Bahasa Query Terapan lanjutan, Basis Data Terdistribusi, Perancangan dan Implementasi Basisdata Menggunakan DB Designer, Lingkungan 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 I

KONSEP DASAR BASIS DATA (*DATABASE*)

1.1 Pengenalan Basis Data (*Database*)

Data merupakan sesuatu hal yang sangat penting dalam kehidupan manusia. Hal ini tidak dapat dipungkiri, karena setiap harinya kita selalu memerlukan dan menggunakan data dalam merencakan segala sesuatu, mempertimbangkan hal apapun, dan mengambil keputusan dalam kehidupan kita. Sebagai contoh, Mahasiswa pastinya memerlukan data antara lain nilai tugas, nilai ujian tengah semester, nilai ujian akhir semester, nilai indeks prestasi sementara, nilai indeks prestasi kumulatif, biaya kuliah, jadwal kuliah serta data-data yang lainnya yang berkaitan dengan kegiatan perkuliahan.

Sekarang, dapatkah kita bayangkan kebutuhan suatu perusahaan atau organisasi terhadap data yang diperlukan bagi kepentingan bisnisnya? Saat ini, baik perusahaan kecil sampai perusahaan besar, pada umumnya telah memiliki informasi dan aplikasi, yang sangat memerlukan data. Artinya, antara sistem informasi dengan data memiliki hubungan yang sangat erat dan tidak dapat dipisah-pisahkan satu sama lainnya. Hal yang penting yang perlu diperhatikan adalah jika kita berbicara mengenai data, maka akan muncul istilah database atau basis data.

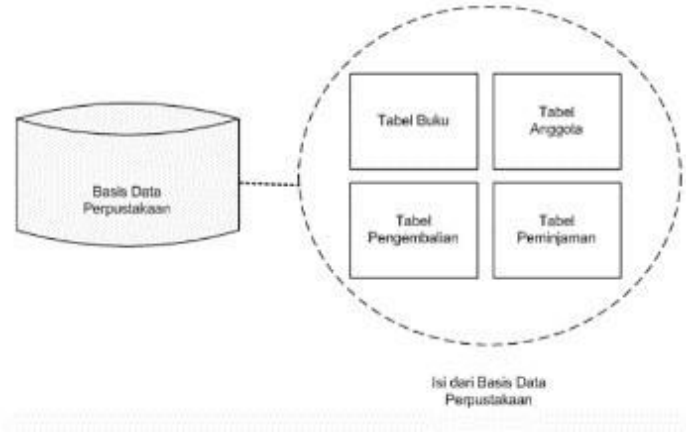
Basis data (database) dapat dibayangkan sebagai sebuah almari arsip. Jika kita memiliki sebuah lemari arsip dan bertugas untuk mengelolanya, maka kemungkinan besar kita akan melakukan hal-hal seperti : memberi map pada kumpulan arsip,

memberi penomoran dengan pola tertentu yang nilainya unik pada setiap map, lalu menempatkan arsip-arsip tersebut dengan urutan tertentu didalam lemari. Kalaupun hal-hal tersebut tidak seluruhnya dilakukan, paling tidak, semua lemari arsip menerapkan suatu aturan tertentu bagaimana keseluruhan arsip-arsip tadi disusun. Yang paling sederhana, tentu menyusun arsip-arsip tadi sesuai kronologisnya dan tanpa pengelompokkan. Hampir tidak akan pernah kita jumpai adanya lemari arsip yang tidak memiliki aturan dalam penyusunan arsip-arsip didalamnya.

Mengapa hal-hal itu kita lakukan? Jawabannya sederhana : kita berharap agar pada suatu saat nanti, sewaktu kita bermaksud untuk mencari dan mengambil arsip dari lemari maka kita akan dapat melakukannya dengan mudah dan cepat (Fathansyah, 2012). Basis Data (Database), pada saat ini sangat berdampak besar pada perkembangan ekonomi dan masyarakat. Sistem basis data berkaitan penting dalam pengembangan bidang rekayasa perangkat lunak, dan database menjadi kerangka kerja yang mendasari sistem informasi dan secara mendasar merubah cara banyak organisasi beroperasi. Contoh Penggunaan Basis Data pada aplikasi: aplikasi pengelolaan nomor telepon, aplikasi pembayaran gaji perusahaan, dll.

Berikut contoh penggunaan basis data : peminjaman pada perpustakaan : Ketika kita melakukan peminjaman di perpustakaan, kemungkinan besar basis data diakses. Petugas akan memasukkan kode buku atau menggunakan mesin pembaca, mesin ini dihubungkan dengan aplikasi database barang untuk mengetahui data buku tersebut. Aplikasi itu kemudian akan mengurangi jumlah stok buku tersebut dan menampilkan jumlah stok yang ada kepada petugas. Jika jumlah stok buku yang ada sudah di bawah ambang batas bawah stok, maka sistem database akan secara otomatis

menginformasikan kepada petugas bahwa peminjaman sudah tidak bisa dilakukan. Atau, jika pembaca menanyakan ketersediaan sebuah buku , maka petugas bisa melakukan pemeriksaan stok buku dan lokasi penyimpanan buku, dengan menjalankan aplikasi yang menentukan ketersediaan buku dari basis data.



Gambar I.1
Contoh Penggunaan Basis Data – Peminjaman Buku Pada Perpustakaan

Gambar diatas merupakan file-file yang disimpan dalam basis data pada harddisk. Basis data Perpustkaan berisi beberapa file atau tabel seperti Tabel Buku, Tabel anggota, Tabel Peminjaman dan Tabel Pengembalian.

1.2 Konsep Dasar Basis Data

1. Definisi Basis Data (*Database*)

Menurut (Fathansyah, 2012) menyampaikan bahwa :
Basis data terdiri dari dua kata , yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan dan sebagainya yang diwujudkan dalam bentuk angka, huruf, symbol, teks, gambar, bunyi atau kombinasinya.

Sedangkan menurut (Indrajani, 2009) menyampaikan bahwa, ada beberapa definisi tentang data, antara lain :

a. Data adalah fakta atau observasi mentah yang biasanya mengenai fenomena fisik atau transaksi bisnis.
b. Lebih khusus lagi, data adalah ukuran objektif dari atribut (karakteristik) dari entitas seperti orang, tempat, benda atau kejadian.
c. Representasi fakta yang mewakili suatu objek seperti pelanggan, karyawan, mahasiswa, dan lain-lain, yang disimpan dalam bentuk angka, huruf, symbol, teks, gambar, bunyi dan kombinasinya.

Menurut (Fathansyah, 2012), sebagai satu kesatuan istilah, Basis Data (database) dapat didefinisikan dalam sejumlah sudut pandang seperti :

a. Himpunan kelompok data(arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

2. Prinsip dan Tujuan Basis Data

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data/arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip.

Perbedaannya hanya terletak pada media penyimpanan yang digunakan, jika lemari dari besi atau kayu sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti cakram magnetis (magnetic disk) (Fathansyah, 2012).

Hal ini merupakan konsekuensi yang logis, karena lemari arsip langsung dikelola oleh manusia, sementara basis data dikelola melalui mesin pinter elektronis (komputer). Yang sangat ditonjolkan dalam basis data adalah pengaturan, pemilahan, pengelompokkan, pengorganisasian data yang akan disimpan sesuai dengan fungsi atau jenisnya. Pemilahan, pengelompokkan, pengorganisasian ini dapat berbentuk sejumlah tabel terpisah atau dalam bentuk pendefinisian kolom-kolom (field) dalam setiap tabel. Seperti contoh Gambar I.1, Basis data Perpustakaan berisi beberapa tabel didalamnya, yaitu tabel Anggota, tabel Buku, tabel Peminjaman dan tabel Pengembalian.

1.3 Operasi Dasar Basis Data

Di dalam sebuah disk, kita dapat menempatkan beberapa (lebih dari satu) basis data. Sementara dalam sebuah basis data, kita dapat menempatkan satu atau lebih tabel. Pada tabel ini sesungguhnya data disimpan dan ditempatkan. Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Misalnya ada basis data kepegawaian, basis data akademik, basis data inventori, dan sebagainya.

Karena itu, operasi-operasi dasar yang dapat kita lakukan berkenaan dengan basis data menurut (Fathansyah, 2012) dapat meliputi:

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru

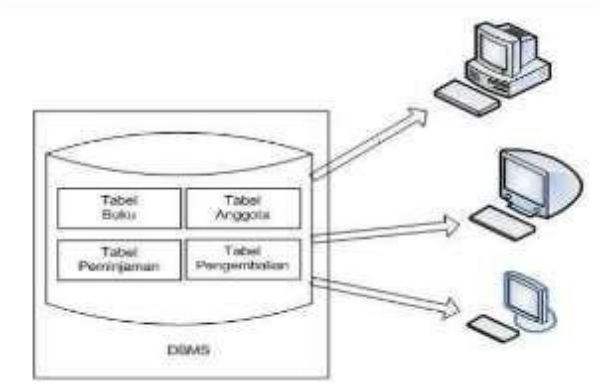
2. Penghapusan basis data (*drop database*), yang identik dengan perusakan lemari arsip (sekaligus berserta isinya, jika ada)
3. Pembuatan tabel baru ke suatu basis data (*create table*), yang identik dengan penambahan map arsip ke sebuah lemari arsip yang telah ada
4. Penghapusan tabel dari suatu basis data (*drop table*), yang identik dengan 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 identik dengan penambahan lembaran arsip ke sebuah map arsip
6. Pengambilan data dari sebuah tabel *(query*), yang identik dengan pencarian lembaran arsip dari sebuah map arsip
7. Pengubahan data dari sebuah tabel (*update*), yang identik dengan perbaikan isi lembaran arsip yang ada di sebuah map arsip
8. Pengahpusan data dari sebuah tabel (*delete*), yang identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip

1.4 Sistem Basis Data

Menurut (Fathansyah, 2012), sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi dan tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses tertentu. Sebuah kendaraan dapat mewakili sebuah sistem yang terdiri atas komponen pemantik/starter (untuk memulai pengapian), komponen pengapian (untuk pembakaran BBM yang membuat torak bekerja), komponen penggerak/torak (untuk menggerakkan roda), komponen pengereman (untuk memperlambat dan menghentikan

gerakan torak dan roda), komponen pelistrikan (untuk mengaktifkan speedometer, lampu, dan lain-lain) yang secara bersama-sama melaksanakan fungsi kendaraansecara umum, yakni sebagai sarana transportasi.

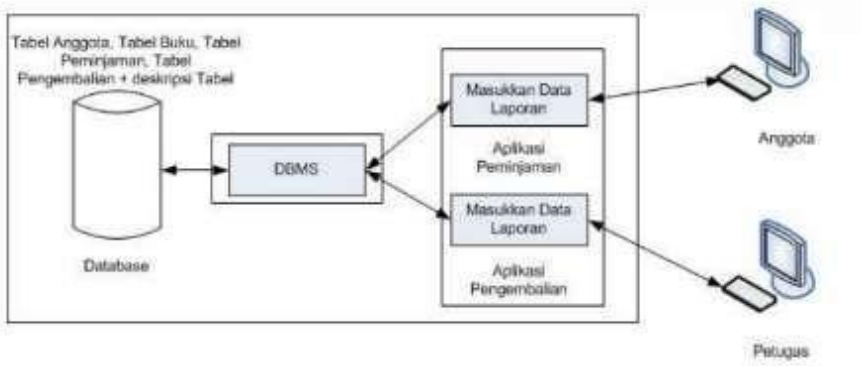
Basis data hanyalah sebuah objek yang pasif. Ia ada karena ada pembuatnya. Ia tidak akan pernah berguna jika tidak ada penegelolanya atau penggeraknya. Yang menjadi pengelola atau penggeraknya secara langsung adalah program/aplikasi (software). Gabungan keduanya (basis data dan pengelolaannya) menghasilkan sebuah sistem. Karena itu, secara umum menurut (Fathansyah, 2012), Sistem Basis Data merupakan sistem yang terdiri atas kumpulan tabel data yang saling berhubungan (dalam sebuah basis data di sebuah sistem computer) dan sekumpulan program (yang biasa disebut dnegan DBMS (*Database Management System*) yang memungkinkan beberapa pemakai dan atau program lain untuk mengakses dan memanipulasi tabel-tabel tersebut.



Gambar I.2
Sistem Basis Data

1.5 Data Base Management *System (DBMS)*

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 DBMS (RDBMS) Contoh perangkat lunak yang termasuk DBMS: dBase, FoxBase, Rbase, Microsoft-Access, Borland Paradox / Borland Interbase, MS-SQL Server, Oracle, Informix, Sybase, MySQL, dll.



Gambar I.3
Contoh Komputer Mengakses Database

Penjelasan Gambar diatas menunjukkan bagaimana sebuah komputer mengakses sebuah database : Database menampung semua data, dimulai dari anggota, data buku sampai dengan data transaksi peminjaman dan pengembalian, Sehingga anggota dapat melihat data buku yang tersedia dalam perpustakaan begitu pula anggota dapat melihat buku apa saja yang sudah dipinjam dan waktu pengembaliannya. Begitu pula dengan petugas dapat melihat fungsi yang sama pula.

1.6 Komponen Sistem Basis Data

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

1. Perangkat Keras (*Hardware*)

   Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah

   a. Komputer (satu untuk sistem stand alone atau lebih dari satu untuk sistem jaringan)
   b. Memori sekunder yang on line (Harddisk)
   c. Memori sekunder yang off line (Tape atau removable disk) untuk keperluan backup data
   d. Media/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 I/O, 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. Sistem (Aplikasi/Perangkat Lunak) 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. Pemakai (*User*)

Ada beberapa jenis atau tipe pemakai terhadap suatu sistem basis data yang dibedakan berdasarkan cara mereka berinteraksi terhadap sistem :

a. Programmer Aplikasi

Pemakai 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, dan lain-lain).

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.

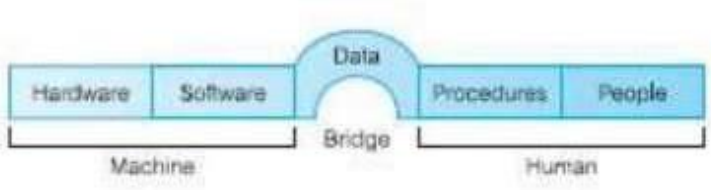
d. User Khusus (*Specialized User*)

Pemakai yang meulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khsusus, seperti aplikasi *Artificial Intelligence*, Sistem Pakar, Pengolahan citra, dan lain-lain, yang bias saja mengakses basis data dengan atau tanpa DBMS yang bersangkutan.

6. Aplikasi (Perangkat Lunak) lain

Aplikasi (perangkat lunak) ini bersifat optional. Artinya, ada atau tidaknya tergantung pada kebutuhan kita.

1.7 Komponen Sebuah DBMS



Gambar I.4
Komponen DBMS

Menurut (Ladjamudin, 2004), kelima komponen tersebut dapat diklasifikasikan menjadi :

a. *Hardware* dan *Software* yang berfungsi sebagai mesin.
b. *People* dan *Procedure* yang merupakan manusia dan tatacara menggunakan mesin.
c. Data merupakan jembatan penghubung antara manusia dan mesin agar terjadi suatu proses pengolahan data.

1. *Hardware* (Perangkat Keras)

   DBMS dan software aplikasinya membutuhkan perangkat keras (*hardware*) untuk bias bekerja dan berfungsi dengan baik. Perangkat keras tersebut dapat berupa Personal Computer (PC) yang stand alone sampai kepada suatu mainframe yang terkoneksi dalam suatu jaringan. Perankat keras umumnya akan menyesuaikan diri dengan kebutuhan suatu organisasi dan DBMS. Beberapa DBMS hanya bias dijalankan pada beberapa perangkat keras dan sistem operasi, sementara lainnya dapat digunakan secara umum oleh perangkat keras dan sistem operasi apa saja.

2. *Software* (Perangkat Lunak)

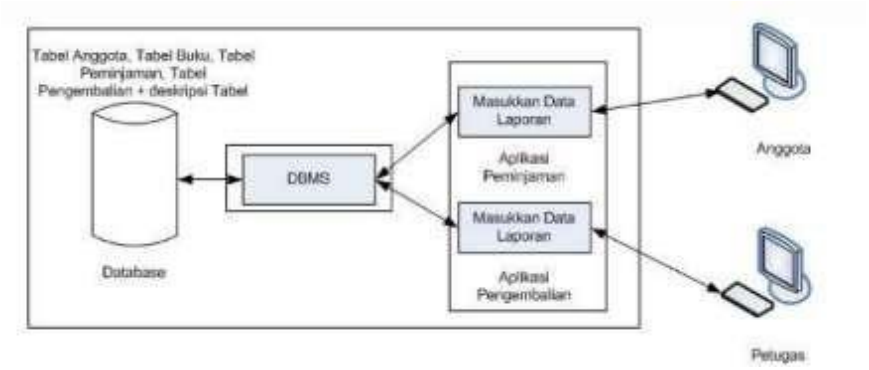
   Komponen perangkat lunak terdiri dari perangkat lunak DBMS itu sendiri bersama dengan sistem operasi, juga termasuk didalamnya adalah perangkat lunak jaringan. Jika DBMS tersebut digunakan pada suatu jaringan dan program-program aplikasi. Bahasa pemrograman/program aplikasi dapat dikategorikan sebagai berikut :

   a. Bahasa pemrograman generasi ketiga, seperti C++, Fortran, PASCAL, dan lain-lain.
   b. Bahasa pemrograman generasi keempat, seperti SQL, Oracle, Sybase,LISP, Informaix, dan lain-lain.

3. Data

   Data merupakan komponen yang sangat penting dalam sebuah DBMS. Basis Data terdiri dari data-data operasional dan meta data (data mengenai data). Struktur basis data dinamakan skema. Data didalam basis data mempunyai sifat

terpadu (*integrated*) dan berbagi (*shared*). Terpadu berarti bahwa berkas-berkas data yang ada pada basis data daling terkait, tetapi kemubaziran data tidak akan terjadi atau hanya terjadi sedikit sekali. Sedangkan berbagi data berarti bahwa data dapat dipakai oleh sejumlah pengguna. Artinya sesuatu data dapat diakses oleh sejumlah pengguna dalam waktu bersamaan.



Gambar I.5
Contoh Komputer Mengakses Basis Data

Skema :

a. Tabel Buku (kode buku, judul buku, dst)
b. Tabel Anggota (kode anggota, nama anggota, dst)
c. Tabel Peminjaman (Kode pinjam, tgl pinjam, dst)
d. Tabel Pengembalian (kode kembali, tgl kembali, dst)

4. *Procedure* (Prosedur)

Prosedur menghubungkan berbagai perintah, dan aturan-aturan yang akan menentukan rancangan dan penggunaan basis data. User (pengguna/pemakai) dari sistem dan staff yang akan mengatur dan merancang basis data berdasarkan prosedur-prosedur yang didokumentasikan. Dokumen tersebut berisi tentang

bagaimana cara menggunakan dan menjalankan suatu sistem. Instruksi-instruksi ini dapat terjadi dari :

a. Log on ke DBMS.
b. Gunakan fasilitas-fasilitas DBMS yang standar, atau program aplikasi.
c. Memulai dan mengakhiri DBMS.
d. Buatkan backup dari basis data tersebut.
e. Tangani masalah-masalah *hardware* dan *software*.
f. Ubahlah struktur tabel, dan lain-lain.

5. *People* (Pengguna/*User*)

People atau pengguna akan berinteraksi dengan mesin *(software* dan *hardware*) melalui berbagai prosedur dan aturan-aturan formal yang berlaku. *People* adalah seseorang yang menginginkan agar persoalannya dikerjakan oleh komputer. Pengguna dapat dikategorikan ke dalam empat golongan, yaitu :

a. Data Administrator (DA) dan Database Administrator (DBA)
    1) Data Administrator (DA) akan bertanggungjawab dalam mengelola sumber daya data berupa:
        a) Perencanaan basis data
        b) Pemeliharaan dan peremajaan suatu standarisasi formal yang berlaku
        c) Menentukan kebijakan-kebijakan, prosedur formal, merancang basis data logic.

Data administrator akan berkonsultasi dengan Manajer Senior, untuk meyakinkan bahwa alur pengembangan basis data akan sangat mendukung dan sejalan dengan target dari enterprise/perusahaan/organisasi tersebut.

2) Database Administratot (DBA) akan bertanggungjawab dalam mengelola sumber daya fisik dari sistem basis data berupa :

a) Bertanggungjawab terhadap seluruh informasi yang berada didalam database.

b) Bertanggungjawab terhadap startegi pengaksesan data dan mengorganisasikan file didlaam media penyimpanan.

c) Sebagai media penghubung/perantara dengan user.

d) Memiliki otoritas pengecekan dan menjalankan prosedur validasi.

e) Bertanggungjawab terhadap strategi backup dan peremajaan data.

f) Mengontrol performasi data dan berhak memberi tanggapan atas usulan-usulan perubahan dan peremajaan data.

b. Database Designer

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

1) Logical database designer (perancang basis data logic)

Logical database designer bertugas mengidentifikasikan data 9seperti entitas dan atribut), kardinalitas relasi dari data tersebut dan bagaimana data tersebut disimpan didalam media penyimpanan sekunder. *Logical database designer* harus memiliki pemahaman yang menyeluruh tentang organisasi data dan aturan-aturan bisnis.

2) *Physical database designer* (perancang basis data fisik)

Physical database designer mengambil model data logical dan memutuskan/menentukan bagaimana model data logic tersebut diimplementasikan ke dalam model data fisik. Pekerjaannya antara lain meliputi:

a) Melakukan mapping terhadap model data logical ke dalam sekumpulan tabel atau relasi yang terintegrasi.

b) Melakukan pemilihan atau menentukan struktur media penyimpanan, dan metode pengaksesan data agar dapat menghasilkan tampilan yang menarik dalam kegiatan-kegiatan basis data dan pengolahan data.

c) Merancang dan menentukan standarisasi keaamanan data.

c. Programmer Aplikasi (*Application Programmer*)

Ketika basis data telah diimplementasikan, programmer aplikasi yang akan membuat berbagai fungsi dan prosedur dalam sistem komputer yang akan digunakan oleh end user. Biasanya programmer aplikasi akan mengerjakan spesifikasi modul yang dirancang oleh sistem analis.

d. End User

1) *Naïve User* (User Umum)

Naïve user adalah mereka yang dapat mengenali DBMS. Mereka mengakses basis data melalui program aplikasi yang telah tertulis secara khusus yang memungkinkan untuk dioperasikan sesederhana mungkin. Mereka mengakses basis data dengan menginput atau memasukkan

perintah-perintah sederhana dari sistem menu. Artinya, mereka tidak perlu mengetahui lebih banyak tentang basis data atau DBMS.

2) *Sophisticated User* (User Mahir)

Sophisticated User adalah mereka yang sangat familiar dengan struktur basis data dan fasilitas-fasilitas yang ditawarkan oleh DBMS. *Sophisticated User* dapat menggunakan bahasa query tingkat tinggi seperti SQL untuk menampilkan atau membuat operasi-operasi yang diinginkan. Beberapa *Sophisticated User* akan membuat sendiri program aplikasi yang mereka butuhkan.

1.8 Keuntungan dan Kekurangan DBMS

Keuntungan DBMS, sebagai berikut :

1. Pengontrolan kerangkapan data
2. Konsistensi data
3. Lebih banyak informasi dari jumlah data yang sama
4. Sharing data
5. Peningkatan integrasi data
6. Peningatan keamanan
7. Penegakan standar layanan

Kekurangan DBMS, sebagai berikut :

1. Kompleksitas
2. Ukuran
3. Biaya DBMS

4. Biaya Peangkat keras tambahan
5. Biaya konversi teknologi
6. Performa
7. Dampak kegagalan yang lebih besar

BAB II

BASIS DATA *RELATIONAL* & PERANCANGAN BASIS DATA

2.1 Basis Data *Relational*

1. Pengertian Basis Data *Relational*

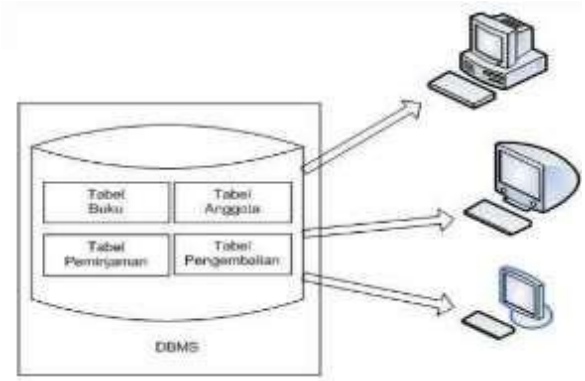
Basis data relasional adalah basis data yang mempresentasikan data dalam bentuk tabel-tabel, dimana tabeltabel tersebut dihubungkan oleh nilai-nilai yang sama/umum pada kolom-kolom terkait. Menurut (Fathansyah, 2012), meyampaikan bahwa model basis data relasional sering pula disebut dengan model relasional 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.

Sedangkan menurut (Hariyanto, 2004), mendefinisikan Basis data relasional merupakan kumpulan sejumlah berhingga relasi. Basis data relasional juga merupakan himpunan skema (metadata) dan humpunan instan skema (data-data).

2. Komponen Penyusun Basis Data

Untuk mengilustrasikan Basis data relasional secara lebih nyata, berikut ini kita ambil sebuah contoh basis data sederhana yang berkaitan dengan perpustakaan. Kita sebut saja basis data tersebut sebagai Basis Data Perpustakaan. Basis Data Perpustakaan terdiri dari empat tabel, yaitu tabel Anggota, tabel Buku, tabel

Peminjaman dan tabel Pengembalian. Basis data perpustakaan memiliki skema sebagai berikut :



Gambar 2.1
Skema Tabel

Tabel Anggota

Kode Anggota	Nama
A01	Surya
A02	Fitri
A03	Syahrur

Tabel Buku

Kode Buku	Judul	Stok Buku
B01	Pemograman C++	10
B02	Membuat Aplikasi 30 Menit	15
B03	Cooking is Easy	15

Skema :

a. Tabel Buku : kode buku (3) , judul buku (20)

b. Tabel Anggota : kode anggota (3), nama anggota (25)

c. Tabel Peminjaman : Kode pinjam (5), tgl pinjam (date), kode anggota (3)

d. Tabel Pengembalian : Kode kembali (5), tgl kembali (date), kode anggota (3)

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 memilki nama yang sama (unik). Tabel disebut juga dengan Relation atau File. Pada gambar diatas terdiri dari 4 tabel yaitu, tabel anggota, tabel buku, tabel peminjaman, tabel pengembalian. Menurut (Ladjamudin, 2004), tabel atau relasi memiliki karakteristik, sebagai berikut:

1) Nama relasi yang digunakan dalam suatu basis data haruslah berbeda satu dengan yang lainnya.
2) Masing-maisng atribut suatu relasi terdiri dari simple attribute dan bernilai tunggal.
3) Masing-masing atribut dalam suatu relasi memiliki yang nama yang unik atau berbeda dengan lainnya.
4) Semua nilai dari suatu atribut haruslah berasal dari domain yang sama.
5) Tidak ada tuple yang ganda.
6) Tuple-tuple boleh tidak berurutan.
7) Atribut-atributnya tidak perlu berurutan.
8) Semua elemen data pada suatu kolom tertentu dalam relasi yang sama harus mempunyai jenis yang sama.

b. Kolom/Atribut

Kolom memiliki nama. Kolom yang terdapat dalam suatu tabel tidak boleh memiliki nama yang sama. Urutan nama boleh sembarang dan tidak

mempengaruhi makna dari tabel. Nama lain kolom adalah Field atau Atribut. Pada gambar diatas, contoh kolom pada tabel Buku yaitu kode buku dan judul buku.

c. Baris/Tuple

Berisikan data dari sebuah objek. Baris pada sebuah tabel harus unik, dapat diletakkan dalam urutan bebas dan tidak mempengaruhi makna dari tabel. Baris disebut juga dengan Record atau tuple. Pada slide diatas tabel anggota dapat menyimpan tiga obyek (yaitu tiga data anggota).

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. Karena domain membatasi dan mengatur nilai yang dapat disimpan maka disebut domain constraint. Pada gambar diatas, kolom yaitu kode anggota hanya berisi 3 nilai saja, yaitu "A01".

3. *Relational Keys*

Relational Keys adalah identifikasi satu atau sekelompok kolom yang nilainya dapat membedakan secara unik tuple-tuple tersebut. Menurut (Ladjamudin, 2005), menyampaikan bahwa *Key* adalah elemen record yang dipakai untuk menemukan record tersebut pada waktu akses, atau bias juga digunakan untuk mengidentifikasi suatu entity atau record atau baris.

Tabel Anggota

Kode Anggota	Nama
A01	Surya
A02	Fitri
A03	Syahrur

Tabel Pengembalian

Kode kembali	Kode pinjam
KM01	PJ01
KM02	PJ02

Tabel Buku

Kode Buku	Judul	Stok Buku
B01	Pemograman C++	10
B02	Membuat Aplikasi 30 Menit	15
B03	Cooking is Easy	15

Tabel Peminjaman

Kode pinjam	Tgl pinjam	Kode buku	Kode anggota	juml	Tgl kembali
PJ01	10-01-2019	B01	A01	1	13-01-2019
PJ01	10-01-2019	B02	A01	1	13-01-2019
PJ01	10-01-2019	B03	A01	1	13-01-2019
PJ02	12-01-2019	B02	A02	1	14-01-2019
PJ02	12-01-2019	B03	A02	1	14-01-2019

Menurut (Pahlevi, 2013) terdapat 5 Relational Keys, sebagai berikut :

a. Superkey

Adalah satu atau kelompok kolom yang nilainya secara unik membedakan tuple-tuple pada suatu tabel. Pada gambar diatas di masing-masing tabel terdapat lebih dari satu *superkey*, yaitu :

1) Tabel anggota : kode anggota, nama anggota
2) Tabel buku : kode buku, judul, stok buku
3) Tabel peminjaman : kode pinjam, tgl pinjam, kode buku, kode anggota, juml, tgl kembali
4) Tabel pengembalian : kode kembali, kode pinjam

Pada gambar diatas di masing-masing tabel terdapat lebih dri satu superkey, yaitu :

1) Tabel anggota :
    a) Kolom kode anggota,
    b) Kolom no faktur dan kolom nama anggota

2) Tabel buku :

a) Kolom kode buku

b) Kolom kombinasi kode buku, judul, stok buku Dst.

b. Candidate Key

Adalah *superkey* di mana tidak ada satupun himpunan bagian dari *superkey* tersebut menjadi superkey lagi. Tidak semua *superkey* menjadi *candidate key*. *Candidate key* yang terdiri dari dua kolom atau lebih disebut sebagai *composite key*. Pada gambar diatas masing-masing tabel terdapat lebih *candidate key* atau bukan *candidate key*, yaitu :

1) Tabel anggota :

a) Kolom kode anggota : *candidate key*

Kolom kode anggota dan kolom nama anggota = bukan *candidate key*

b) Tabel buku :

Kolom kode buku = *candidate key*

Kolom kombinasi kode buku, judul, stok buku : bukan *candidate key*

2) Tabel Peminjaman :

a) Kolom kode pinjam, kode buku, kode anggota : *candidate key*

b) Kolom kombinasi kode pinjam, kode buku, kode anggota, jumlah : bukan *candidate key*

3) Tabel Buku :

a) Kolom kode buku merupakan *candidate key*

b) Kolom kombinasi kode buku, judul buku, stok buku bukan *candidate key*

4) Tabel pengembalian

a) Kolom kode kembali, kode pinjam merupakan *candidate key*

b) Kolom kombinasi kode kembali, kode pinjam bukan *candidate key*

c. Primary Key

Adalah (satu) *candidate key* yang dipilih (di antara *candidate key* lain) untuk membedakan tuple-tuple scara unik dalam tabel. Jika dalam satu tabel hanya terdapat satu *candidate key* (misal tabel anggota dan tabel buku), maka key tersebut menjadi *primary key*. Tetapi jika terdapat lebih dari satu *candidate key* (misal tabel penjualan dan tabel pengembalian), maka salah satu *candidate key* tersebut dpat dijadikan *primary key*. *Primary key* masing-masing tabel pada gambar diatas adalah

1) Tabel anggota : kode anggota
2) Tabel buku : kode buku
3) Tabel peminjaman : kode pinjam
4) Tabel pegembalian : kode kembali

d. Alternate Key

Adalah *candidate key* yang tidak dijadikan sebagai *primary key*. Misal pada tabel pengembalian jika kita memilih kode kembali sebagai *primary key*, maka kode pinjam dapat dijadikan *alternate key*.

e. Foreign Key

Adalah satu atau kelompok kolom yang nilainya sama atau terkait dengan *candidate key* pada tabel lain atau pada tabel yang sama. Misal pada tabel peminjaman ada kolom kode anggota yang terhubung dengan tabel anggota, maka kode anggota adalah *foreign key*. Kolom-kolom yang saling terkait ini sangat penting dalam operasi join. Pada tabel pengembalian ada kolom kode pinjam yang terhubung dengan tabel peminjaman, maka kode pinjam disini juga sebagai *foreign key.*

4. Skema tabel

Adalah informasi dasar yang mendeskripsikan tabel yang terdiri atas nama tabel dan sekumpulan pasangan kolom domain.

Contoh :

Skema Tabel Anggota (kode anggota, nama)

Skema Tabel Buku (kode buku, judul)

Skema Tabel Peminjaman (kode pinjam, tgl pinjam, tgl kembali, juml, kode anggota, kode buku)

Skema Tabel Pengembalian (kode kembali, kode pinjam)

5. Skema Basis Data

Adalah sekumpulan skema tabel dengan masing-masing tabel memiliki nama yang berbeda.

Contoh : Skema Basis Data Perpustakaan : Tabel anggota (kode anggota, nama) , Tabel buku (kode buku, judul, stok buku), Tabel peminjaman (kode pinjam,

tgl pinjam, kode buku, tgl kembali, kode anggota, juml) dan Tabel Pengembalian (kode kembali, kode pinjam).

6. *Integrity Constraint*

Pada penjelasan diatas telah dibahas mengenai domain constraints. Terdapat empat contraints/batasan lain yang menjaga integritas data yang disimpan pada basis data :

a. *Null*

Adalah nilai pada suatu kolom (*tuple)* masih belum diketahui (*unknown).* Ini bisa berarti nilai tersebut tidak dapat diterapkan pada kolom tersebut. Namun, null tidak sama dengan nilai numerik nol atau string "-"; nol dan spasi adalah nilai, tetapi null menunjukkan tidak adanya nilai. Misal dalam sebuah tabel ada sebuah data yang belum diketahui boleh dituliskan *null*, akan tetapi hal tersebut tidak berlaku untuk *primary key*. Karena kolom primary key bersifat unik, jika *primary key* menyimpan *null* maka sifat unik dari kolom tersebut akan hilang karena bisa saja beberapa tuple memiliki nilai *null.*

b. *Entity integrity*

Adalah batasan atau aturan yang menyatakan bahwa kolom-kolom *primary key* tidak boleh menyimpan *null.* Seperti di jelaskan sebelumnya *primary key* digunakan untuk mendefinisikan secara unik sebuah tuple.

c. *Referential integrity*

Adalah batasan yang menyatakan jika suatu tabel memiliki kolom *foreign key* maka nilai pada *foreign key* tersebut harus sesuai dengan nilai kolom

candidate key dan jika tidak demikian maka *foreign key* dapat dituliskan *null.* Dua keadaan penulisan *null* tidak perlu dilakukan :

1) Pada saat kolom tersebut diberikan batasan tidak boleh diberikan *null*.
2) Pada saat kolom tersebut juga merupakan bagian dari *primary key.*

d. General constraints

Adalah batasan /aturan tambahan yang ditetapkan oleh pemakai atau administrator basis data sesuai aturan/batasan yang ada pada suatu organisasi. Contoh :

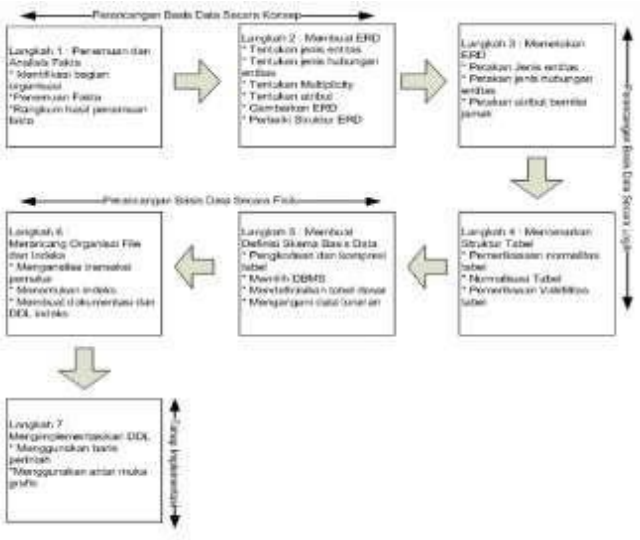
1) Peminjaman buku tidak diijinkan jika stok buku hanya satu
2) Jika anggota masih memiliki buku yang belum dikembalikan maka tidak di perbolehkan untuk meminjam kembali

2.2 Perancangan Basis Data

Proses pembangunan basis data terdiri dari dua tahapan utama :

1. Tahap analisis dan perancangan

   Adalah tahapan pemetaan atau pembuatan model dari dunia nyata menggunakan notasi perancangan basis data tertentu serta pembuatan deskripsi implementasi basis data.



Gambar 2.2
Tahap Analisis dan Perancangan

Tahapan analisis dan perancangan dibagi menjadi tiga, yaitu :

a. Perancangan basis data secara konsep

   Merupakan proses pembuatan data model dan tidak bergantung pada seluruh aspek fisik basis data.

b. 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.

c. Perancangan Basis Data Secara Fisik

   Merupakan proses pembuatan deskripsi implementasi basis data pada media penyimpanan sekunder (disk). Deskripsi ini menjelaskan tabeltabel dasar, organisasi file, indeks untuk mendapatkan ases data secara efisien, dan semua integrity contraints, dan langkah-lagkah keamanan.

2. Tahap Implementasi

Tahapan ini mengimplementasikan rancangan basis data yang telah dibuat. Implementasi menggunakan aplikasi klien yang disediakan oleh DBMS terpilih.

2.3 Studi Kasus

Perpustakaan Smart adalah perpustakaan umum yang anggotanya pelajar, mahasiswa dan masyarakat yang didirikan oleh Walikota Jakarta Barat. Keberadaan perpustakaan berlokasi di Walikota yang aplikasi pelayanan masih bersifat tradisional.

Prosesnya :

1. Setiap calon anggota yang akan menjadi anggota harus mengisi formulirdengan biaya administrasi Rp.10.000,-
2. Anggota dapat meminjam buku maksimal 3 buku
3. Untuk masa peminjaman selama 1 minggu (7 hari)
4. Keterlambatan pengembalian dikenakan denda sesuai dengan kondisi denda, diantaranya. Diantaranya :
    a. Denda keterlambatan pengembalian dikenakan biaya administrasi Rp.500 perharinya (bukti surat denda terlampir)
    b. Denda Buku perpustakaan rusak maka dikenakan biaya revisi buku perpustakaan(biaya ini dikenakan setelah buku diperbaiki).(bukti surat denda terlampir)
    c. Denda Buku Hilang, maka dikenakan biaya penggantian seharga buku tersebut.(bukti surat denda terlampir)

d. Perpustakaan smart dapat menerima sumbangan dari donatur statusnya (anggota atau masyrakat luas)

Buat Database dan Tabel-tabelnya?

BAB III

MODEL DATA

1. Pengertian Model Data

Sebagaimana telah disebutkan di bab sebelumnya, upaya perancangan basis data dapat juga kita tempuh dengan secara langsung membuat sebuah model dari awal sekali. Langkah ini biasanya ditempuh dari kelangkaan data/fakta yang kita miliki. Di sisi lai, sebuah model dinyatakan dalam bentuk diagram awal akan lebih mudah untuk dievaluasi/dianalisis untuk kemudian dilakukan perbaikan-perbaikan untuk mendapatkan sebuah model data yang lebih permanen dan lebih mendekati kenyataan sesungguhnya.

Menurut (Fathansyah, 2012), model data dapat didefinisikan sebagaikumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantic(makna) data dan batasan data. Oleh karena yang ingin ditunjukkan adalah makna daridata dan keterhubungannya dengan data lain, maka model data ini lebih tepat jika disebut Model Data Lojik.

2. Jenis-jenis Model Data

a. Model Data Berdasarkan Object

Menurut (Ladjamudin, 2005), 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), sedangkan atribut merupakan relasi fungsional dari satu objek set ke objek set yang lain. Relationship adalah hubungan alamiah yang terjadi antara entitas. Model data berbasi objek memiliki beberapa bentuk, sebagai berikut :

1) Model Keterhubungan Entitas (Entity-Relationship Model)

a) Pengertian Model Keterhubungan Entitas (Entity Relationship Model / ER Model)

Model Keterhubungan Entitas (Entity-Relationship Model) merupakan model yang paling populer digunakan dalam perancangan basis data. Entity-Relationship Model merupakan Model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa *real world* terdiri dari objek-object dasar yang mempunyai hubungan atau relasi antara objek-objek tersebut. Komponen utama pembentuk Model *Entity-Relationship*, yaitu: Entitas (*Entity*) dan Relasi (*Relation*). Kedua komponen ini dideskripsikan lebih lanjut melalui sejumlah Atribut/Properti.

Tabel Buku

Kode Buku	Judul	Stok Buku
B01	Pemograman C++	10
B02	Membuat Aplikasi 30 Menit	15
B03	Cooking is Easy	15

Tabel Anggota

Kode Anggota	Nama
A01	Surya
A02	Fitri
A03	Syahrur

Tabel Peminjaman

Kode pinjam	Tgl pinjam	Kode buku	Kode anggota	juml	Tgl kembali
PJ01	10-01-2019	B01	A01	1	13-01-2019
PJ01	10-01-2019	B02	A01	1	13-01-2019
PJ01	10-01-2019	B03	A01	1	13-01-2019
PJ02	12-01-2019	B02	A02	1	14-01-2019
PJ02	12-01-2019	B03	A02	1	14-01-2019

Dari tabel diatas kita dapat menentukan :

Entitas : Buku, Anggota, Peminjaman.

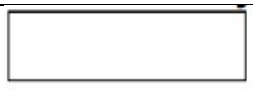
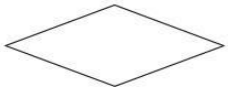
Atribut : Tabel Buku (kode buku, judul, stok buku), Tabel Anggota (kode anggota, nama) dan Tabel Peminjaman (kode pinjam, tgl pinjam, kode buku, kode anggota, juml, tgl kembali).

Relasi : hubungan antara kode buku di tabel buku dengan kode buku di tabel peminjaman. Begitu pula dengan kode anggota.

Model Entity Relationship yang berisi komponen himpunan entitas, relasi, yang dilengkapi atribut-atribut, dapat digambarkan menggunakan Diagram Entity Relationship (Diagram E-R).

b) Simbol dasar yang digunakan :

Tabel III.1
Simbol Dasar Entity Relationship Diagram

	Menyatakan Himpunan Entitas
	Menunjukan Himpunan Relasi
	Menyatakan Atribut (Atribut key digaris bawahi)

———————	Penghubung/Link

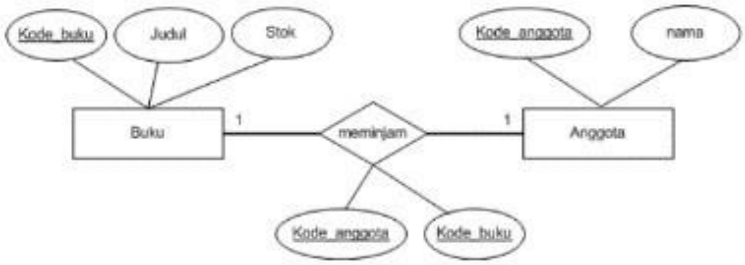
c) Mapping Kardinalitas

Dalam Diagram E-R aturan terpenting adalah Kardinalitas relasi/ *Mapping Cardinalities* yang menentukan jumlah entity yang dapat dikaitkan dengan entity lainnya melalui relationship-set.

Jenis *Mapping Cardinalities*:

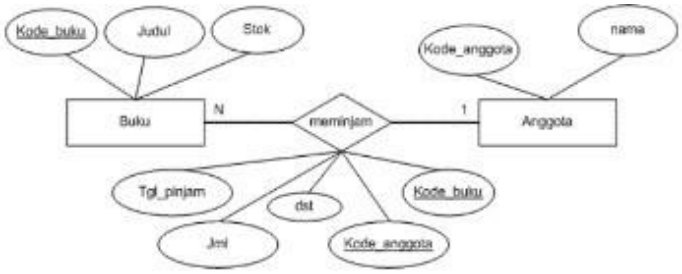
(1) Relasi satu ke satu *(one-to-one*)

Contoh :



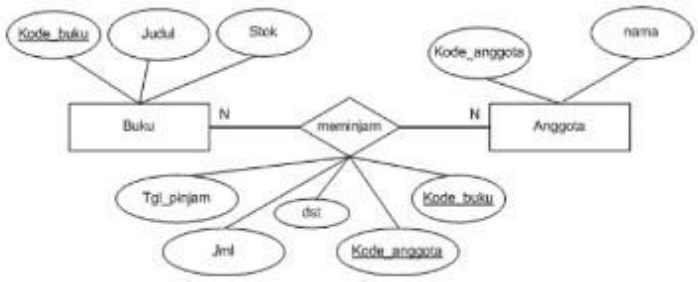
Gambar III.1
Relasi *one to one*

(2) Relasi satu ke banyak (*one-to-Many*)



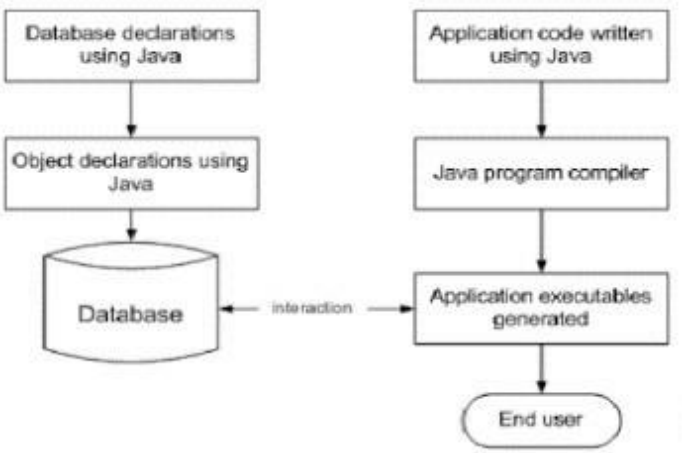
Gambar III.2
Relasi *one to many*

(3) Relasi banyak ke banyak (*many-to-many*)



Gambar III.3
Relasi *many to many*

2) Model Berorientasi Object (Object-Oriented Model)



Gambar III.4
Model Berorientasi Object

Penggambaran model berbasis objek menggunakan UML. UML Digambarkan dengan 2 Jenis, menurut (Sukamto & Shalahuddin, 2018):

a) Structural Diagram

(1) Class Diagram

Menggambarkan struktur sistem dari segi pendefinisiankelas-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*). *Collaboration diagram* sudah tidak muncul pada UML versi 2.x.

(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.

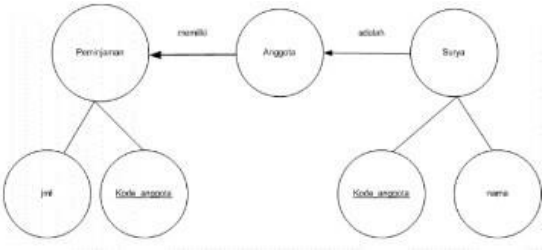
3) Model Data Semantik (*Semantic Data Model*)

Hampir sama dengan *Entity Relationship model* dimana relasi antara objek dasar tidak dinyatakan dengan simbol tetapi menggunakan kata-kata (*Semantic*). Sebagai contoh, dengan masih menggunakan relasi pada Bank X sebagaimana contoh sebelumnya, dalam semantic model adalah seperti terlihat pada gambar di atas. Tanda-tanda yang menggunakan dalam *semantic model* adalah:

Tabel III.2
Simbol Model Data Semantik

→	Menujukkan adanya relasi
———	Menunjukkan atribut

Contoh :



Gambar III.5
Contoh Model Data Semantik

4) Model Data Fungsional (*Functional Data Model*)

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 (*higher level description of implementation*)

Jenis-Jenis Model Data Berbasis Record :

1) Model Relational

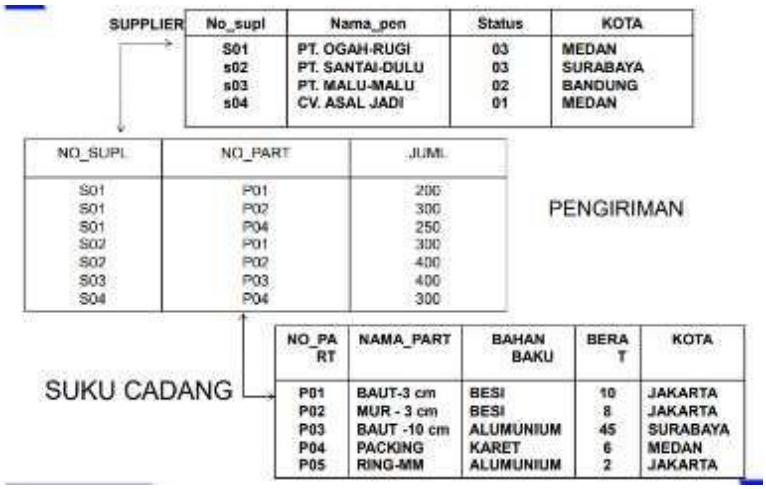
Dimana data serta hubungan antar data direpresentasikan oleh sejumlah tabel dan masingmasing tabel terdiri dari beberapa kolom yang namanya *unique*. Model ini berdasarkan notasi teori himpunan (*set theory*), yaitu relation.

Contoh : database penjualan terdiri dari 3 tabel, sebagai berikut :

(a) Tabel Supllier

(b) Tabel Suku Cadang

(c) Tabel Pengiriman

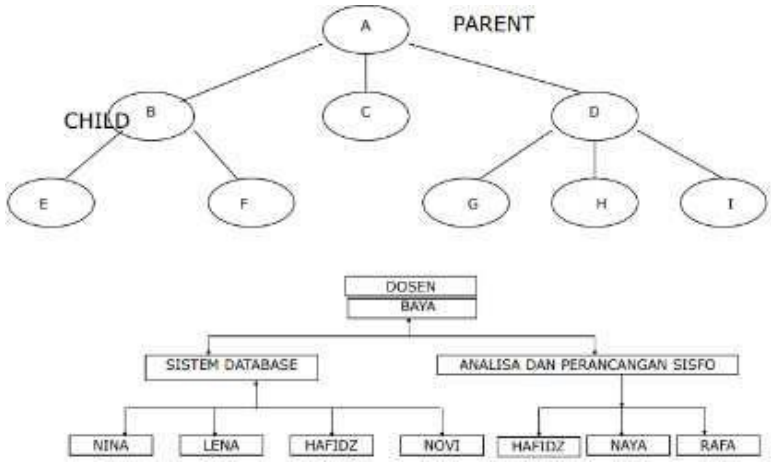


Gambar III.6
Contoh Model Data Relational

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 cardinalitas 1:1 dan 1:M.

Contoh :

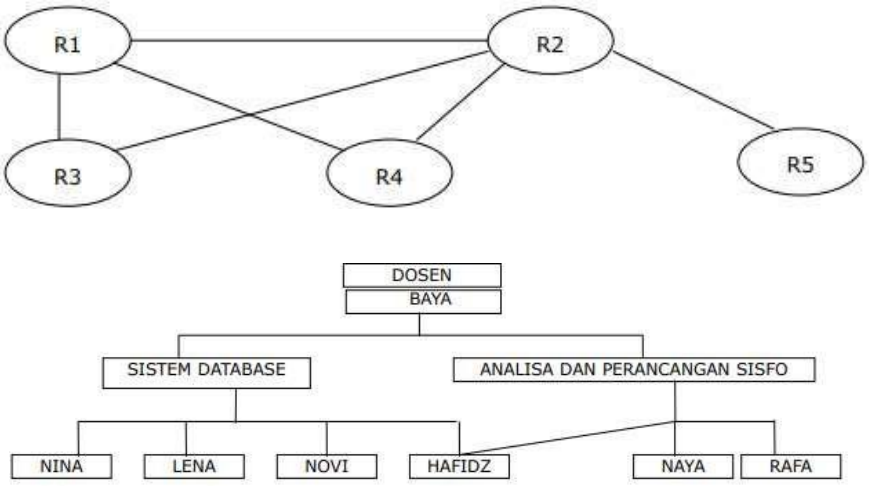


Gambar III.7
Contoh Model Data Hirarki

3) Model Jaringan

Distandarisasi tahun 1971 oleh Database Task Group (DBTG) atau disebut juga model CODASYL (Conference on Data System Language), mirip dengan hirarkical model dimana data dan hubungan antar data direpresentasikan dengan record dan links. Perbedaannya terletak pada susunan record dan linknya yaitu network model menyusun record-record dalam bentuk graph dan menyatakan hubungan cardinalitas 1:1, 1:M dan N:M.

Contoh :



Gambar III.8
Contoh Model Data Jaringan

BAB IV

ENTITY RELATIONSHIP DIAGRAM (ERD)

4.1 Pengertian *Entity* Relationship Diagram (ERD)

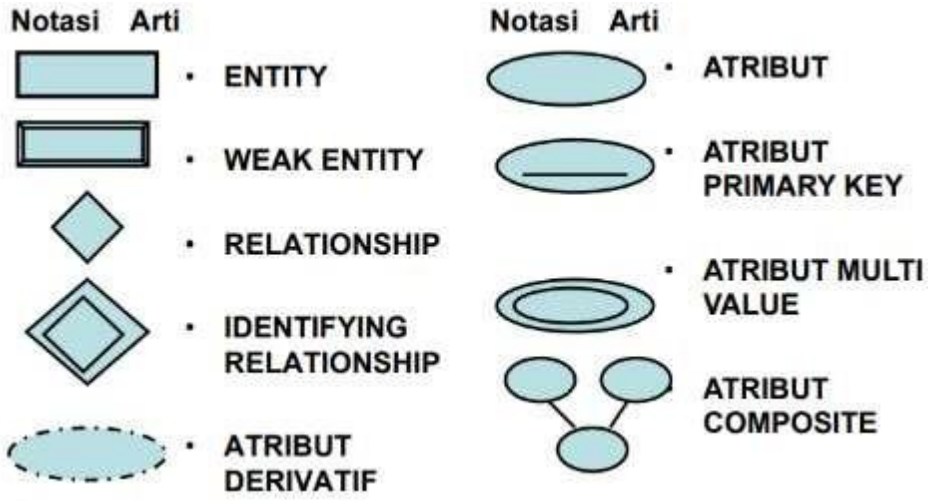
Beberapa pengertian mengenai ERD oleh beberapa ahli:

1. Menurut (Ladjamudin, 2004), ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak.
2. 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.
3. 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.

Biasanya ERD ini digunakan oleh professional sistem untuk berkomunikasi dengan pemakai eksekutif tingkat tinggi dalam suatu organisasi. Pemakai ini lebih tertarik dengan hal-hal sebagai berikut :

a. Data apa saja yang dibutuhkan untuk bisnis mereka?
b. Bagaimana data tersebut berelasi dengan data lainnya?
c. Siapa saja yang diperkenankan untuk mengakses data tersebut?

4.2 Komponen ERD



Gambar IV.1
Komponen ERD

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

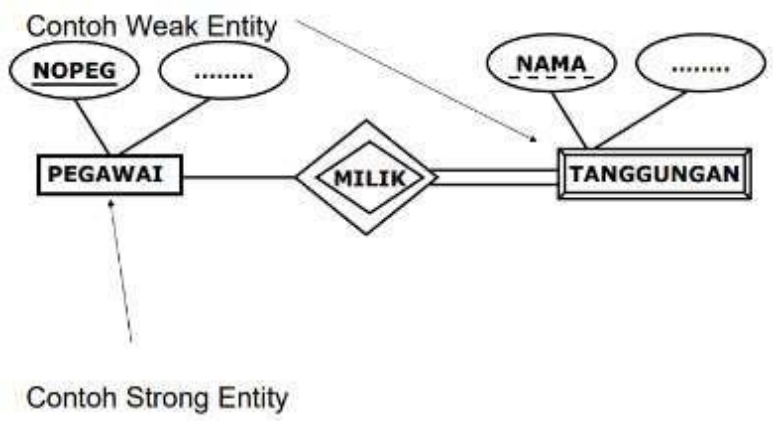
Yaitu entity set yang satu atau lebih atributnya digunakan oleh entity set lain sebagai key. Digambarkan dengan empat persegi panjang. Misal : E adalah sebuah entity set dengan atribute-atribute a1, a2,..,an, maka entity set tersebut direpresentasikan dalam bentuk tabel E yang terdiri dari n kolom, dimana setiap kolom berkaitan dengan atribute-atributenya.

b. Weak Entity set

Yaitu Entity set yang bergantung terhadap *strong entity set*. Digambarkan dengan empat persegi panjang bertumpuk. Misal : A adalah *weak entity set*

dari atribute-atribute a1, a2, .., ar dan B adalah strong entity set dengan atribute-atribute b1, b2,..,bs, dimana b1 adalah atribute primary key, maka weak entity set direpresentasikan berupa table A, dengan atribute-atribute {b1} u {a1,a2,.., ar}

Contoh :



Gambar IV.2
Contoh Strong dan Weak Entity Set

2. Atribut

Atribut adalah kumpulan elemen data yang membentuk suatu entitas.

Jenis-jenis atribut :

a. Atribut Kunci

Merupakan atribut yang digunakan untuk menentukan suatu entity secara unik

b. Atribut simple

Merupakan atribut yang bernilai tunggal

c. Atribut Multi Value

Merupakan atribut yang memiliki sekelompok nilai untuk setiap instan entity.

Contoh : Pada gambar dibawah ini, yang menjadi atribut key adalah NIP. Tgl Lahir dan Nama adalah atribut simple. Sedangkan Gelar merupakan contoh atribut multivalue.

Gambar IV.3
Contorh Atribut Key, Simple dan Multivale

d. Atribut Composit

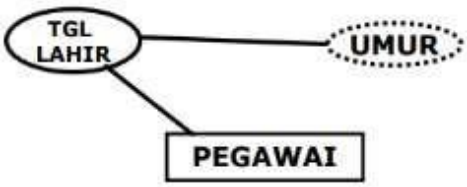
Merupakan suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu. Contohnya adalah atribut nama pegawai yang terdiri dari nama depan, nama tengah dan nama belakang.

Gambar IV.4
Contoh Atribut Composit

e. Atribut Derivatif

Merupakan suatu atribut yg dihasilkan dari atribut yang lain. Sehingga umur yang merupakan hasil kalkulasi antara Tgl Lahir dan tanggal hari ini.

Sehingga keberadaan atribut umur bergantung pada keberadaan atribut Tgl Lahir.



Gambar IV.5
Contoh Atribut Derivatif

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:

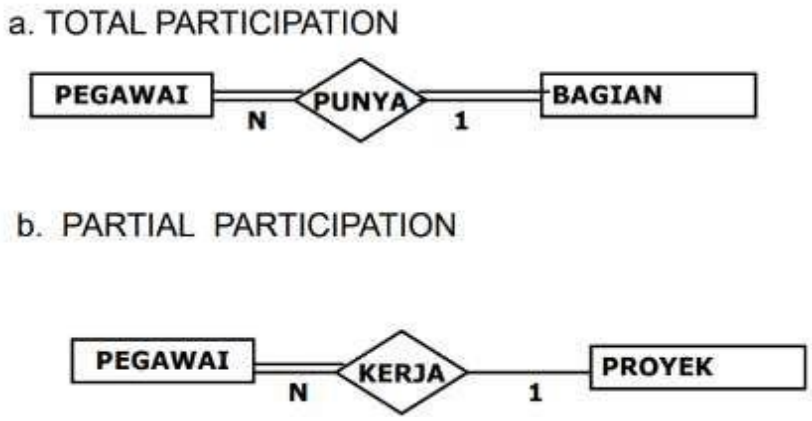
a. Total participation constrain

Yaitu Keberadaan suatu entity tergantung pada hubungannya denganentity lain. Didalam diagram ER digambarkan dengan dua garis penghubung antar entity dan relationship.

b. Partial participation

Yaitu Keberadaan suatu entity tidak tergantung pada hubungan dengan entity lain. Didalam diagram ER digambarkan dengan satu garis penghubung.

Contoh :



Gambar IV.6
Contoh Jenis Participation Constrain

4. Indicator Type

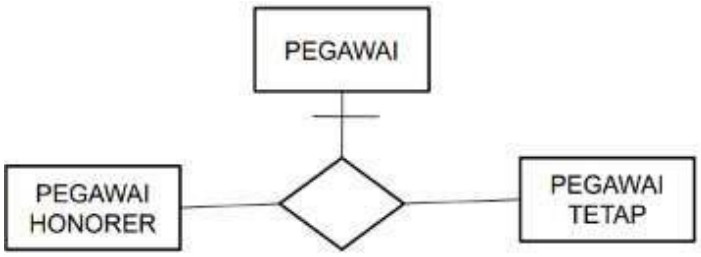
a. Indicator Asosiatif Object

Indicator tipe asosiatif object berfungsi sebagai suatu objek dan suatu relationship. Contoh :

Gambar IV.7
Contoh *Indicator Asosiatif Object*

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. Contoh :



Gambar IV.7
Contoh *Indicator Supert type*

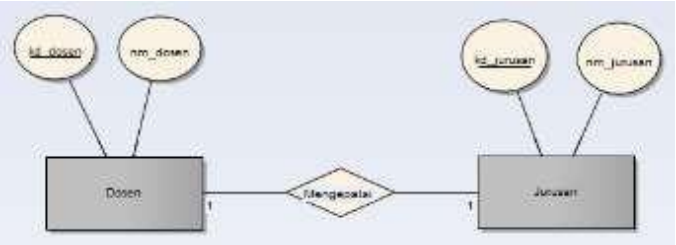
4.3 Kardinalitas / Derajat Relasi

Menurut (Fathansyah, 2012), Kardinalitas relasi 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)

   Yang berarti 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.

   Contoh :



Gambar IV.8
Contoh *One to One*

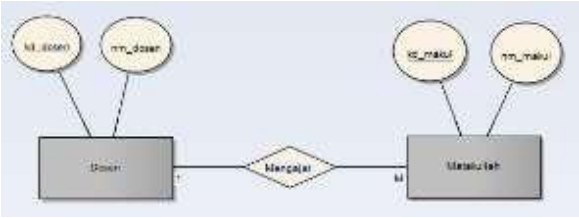
Pada relasi ini, setiap dosen paling banyak mengepali satu jurusan (walaupun memang tidak semua dosen yang menjadi ketua jurusan). Dan setiap jurusan pasti dikepalai oleh paling banyak satu orang dosen.

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.

   Contoh :



Gambar IV.9
Contoh *One to Many*

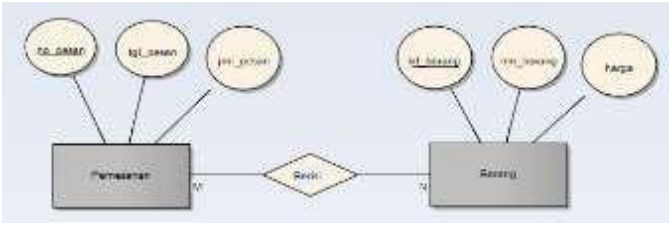
Pada relasi ini, setiap dosen dapat mengajar lebih dari satu matakuliah,sedang setiap matakuliah diajar hanya oleh paling banyak satu orang dosen.

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

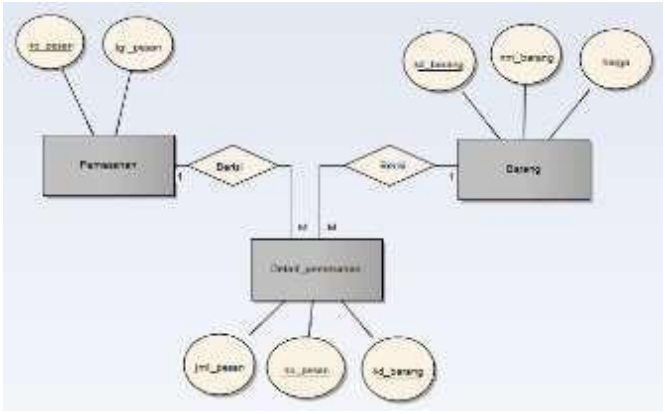
setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

Contoh :



Gambar IV.10
Contoh *Many to Many*

Pada relasi diatas setiap pemesanan bias berisi banyak barang dan setiap barang mengisi banyak pemesanan. Jika kardinalitasnya Many to Many ini dapat disederhanakan kembali, karena munculnya entitas yang baru. Penggambarannya sebagai berikut :



Gambar IV.11
Contoh *Many to Many*

Pada ERD yang baru ini, muncul entitas yang baru yaitu Detail_pemesanan. Dan dalam entitas Detail_pemesanan, terdapat no_pesan dan kd_barang sebagai *Foreign Key*.

4.4 Tahapan Pembuatan ERD

Menurut (Hidayatullah & Kawistara, 2015), Tahpan pembuatan ERD sebagai beirkut :

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.

4.5 *Logical* Record *Structured* (LRS)

Merupakan representasi dari struktur record-record pada tabel-tabel yang terbentuk dari hasil relasi antar himpunan entitas.

Menentukan Kardinalitas, Jumlah Tabel dan Foreign Key (FK)

1. *One To One*

Contoh :

Gambar IV.12
Contoh *One To One*

Gambar di atas menunujukan relasi dengan kardinalitas 1-1, karena: 1 supir hanya bisa mengemudikan 1taksi, dan 1 taksi hanya bisa dikemudikan oleh 1 supir. Relasi 1-1 akan membentuk 2 tabel:

a. Tabel Supir (nosupir, nama, alamat)

b. Tabel Taksi (notaksi, nopol, merk, tipe)

Gambar IV.13
Contoh LRS

2. *One To Many*

Contoh :

Gambar IV.14
Contoh *One to Many*

Gambar di atas menunujukan relasi dengan kardinalitas 1-M, karena: 1 Dosen bisa membimbing banyak Kelas, dan 1 Kelas hanya dibimbing oleh 1 Dosen. Relasi 1-M akan membentuk 2 tabel:

a. Tabel Dosen (nip, nama, alamat)

b. Tabel Kelas (kelas, jurusan, semester, jmlmhs)

Gambar IV.15
Contoh *LRS*

3. Many To Many

Contoh :

Gambar IV.16
Contoh *Many to Many*

Gambar di atas menunjukan relasi dengan kardinalitas M-M, karena: 1 Mahasiswa bisa belajar banyak Mata Kuliah, dan 1 Mata Kuliah bisa dipelajari oleh banyak Mahasiswa. Relasi M-M akan membentuk 3 tabel:

a. Tabel Mahasiswa (nim, nama, alamat)

b. Tabel Mtkuliah (kdmk, nmmk, sks)

c. Tabel Nilai (nim, kdmk, nilai) → menggunakan super key/composite key

Gambar IV.17
Contoh *LRS*

4.6 Membuat ERD

Langkah-langkah pembuatan ERD dan LRS:

1. Tentukan entity-entity yang diperlukan
2. Tentukan relationship antar entity-entity
3. Menggambar ERD Sementara
4. Mengisi kardinalitas
5. Menentukan kunci utama
6. Menggambar ERD Berdasarkan Kunci
7. Tentukan attribute-attribute
8. Transformasi ERD ke LRS Menggambar LRS

Contoh :

Sebuah perusahaan mempunyai beberapa bagian. Masing-masing bagian mempunyai pengawas dan setidaknya satu pegawai. Pegawai harus ditugaskan pada paling tidak satu bagian, tetapi dapat pula beberapa bagian. Paling tidak satu pegawai mendapat tugas sebuah proyek. Namun, seorang pegawai dapat libur dan tidak mendapat tugas proyek.

Deskripsi Permasalahan :

a. Masing-masing bagian hanya mempunyai satu pengawas
b. Seorang pengawas hanya bertugas pada satu bagian
c. Masing-masing bagian memiliki paling tidak satu pegawai
d. Masing-masing pegawai bekerja paling tidak pada satu bagian
e. Masing-masing proyek dikerjakan oleh paling tidak satu pegawai
f. Seorang Pengawas bisa mendapat tugas 0 atau beberapa proyek

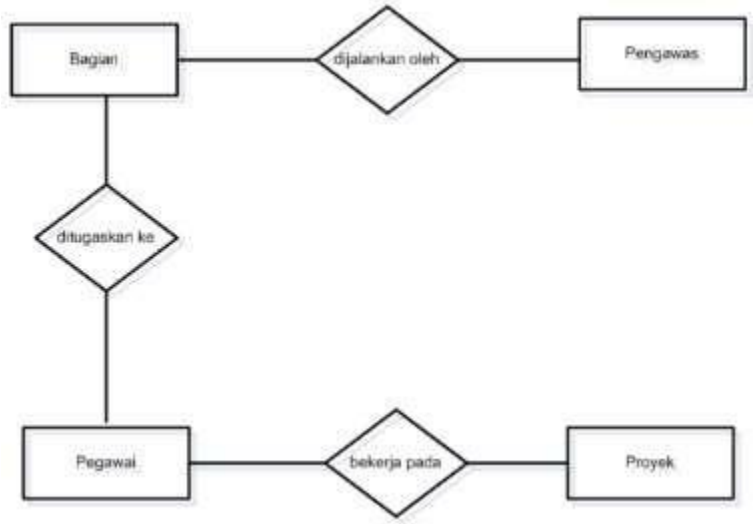
Langkah 1 : Menentukan Entitas Entitas yang dibutuhkan adalah : Bagian, Pegawai, Pengawas, dan Proyek

Langkah 2 : Menentukan Relasi dengan matriks relasi

	Bagian	Pegawai	Pengawas	Proyek
Bagian		ditugaskan ke	dijalankan oleh	
Pegawai	milik			bekerja pada
Pengawas	menjalankan			
Proyek		menggunakan		

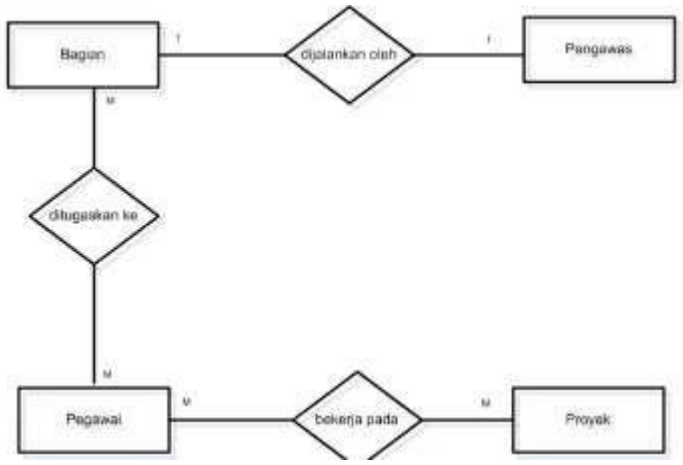
Gambar IV.18
Hasil Langkah kedua

Langkah 3 : Menggambar ERD Sementara



Gambar IV.19
Hasil Langkah ketiga

Langkah 4 : Mengisi Kardinalitas

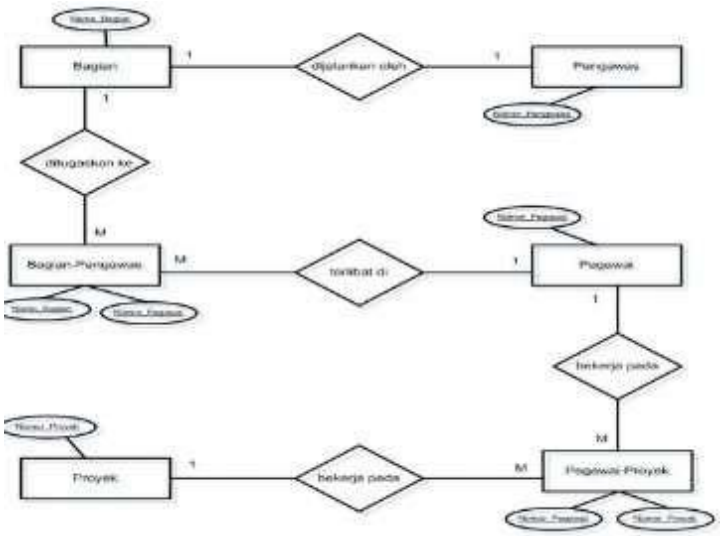


Gambar IV.20
Hasil Langkah keempat

Langkah 5 : Menentukan Kunci Utama Kunci Utama : Nama Bagian, Nomor Pengawas, Nomor Pegawai, Nomor Proyek.

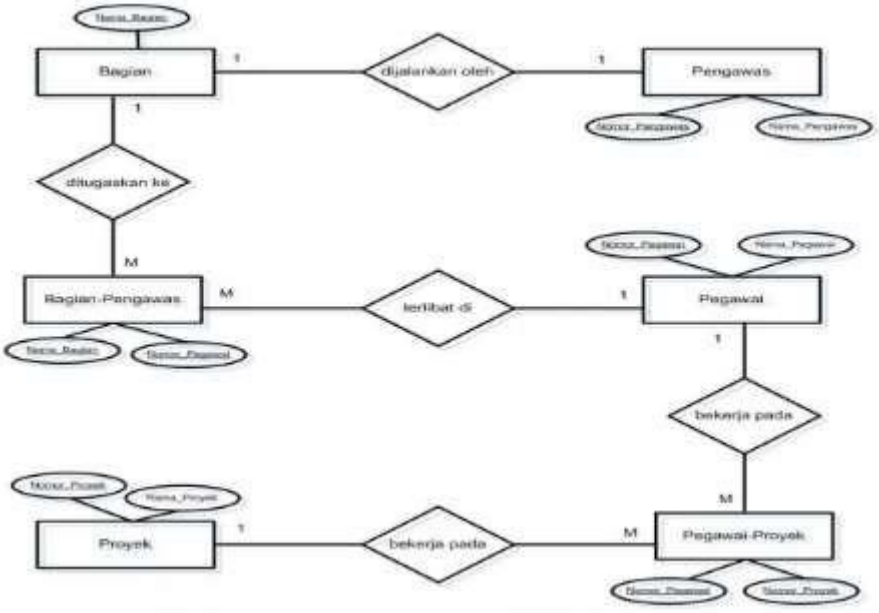
Langkah 6 : Menggambarkan ERD berdasarkan kunci

Karena ada dua relasi many-to-many pada ERD sementara, yaitu antara Bagian dan Pegawai, serta Pegawai dan Proyek. Oleh karena itu dibuatkan entitas baru yaitu Bagian-Pegawai dan Pegawai-Proyek. Kunci utama Bagian-Pegawai adalah gabungan Nama Bagian dan Nomor Pegawai. Kunci utama Pegawai-Proyek adalah gabungan Nomor Pegawai dan Nomor Proyek



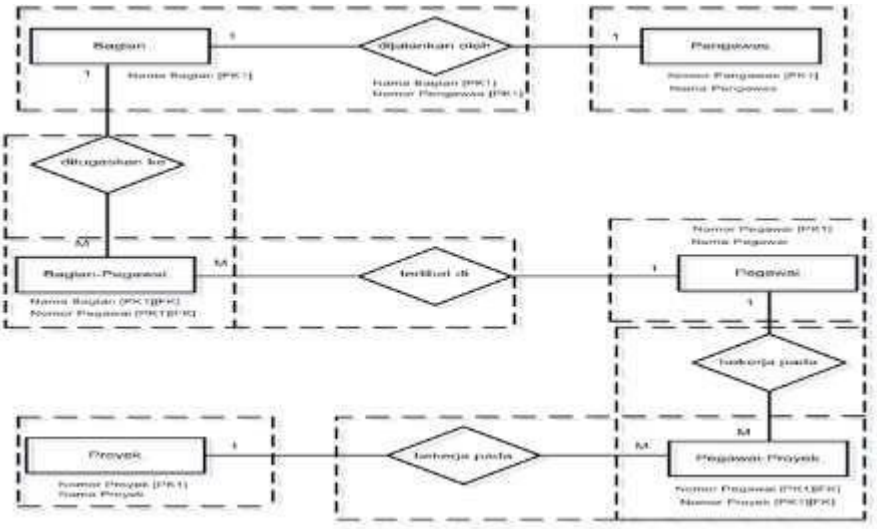
Gambar IV.21
ERD berdasarkan Kunci

Langkah 7: Menentukan Atribut yang diperlukan



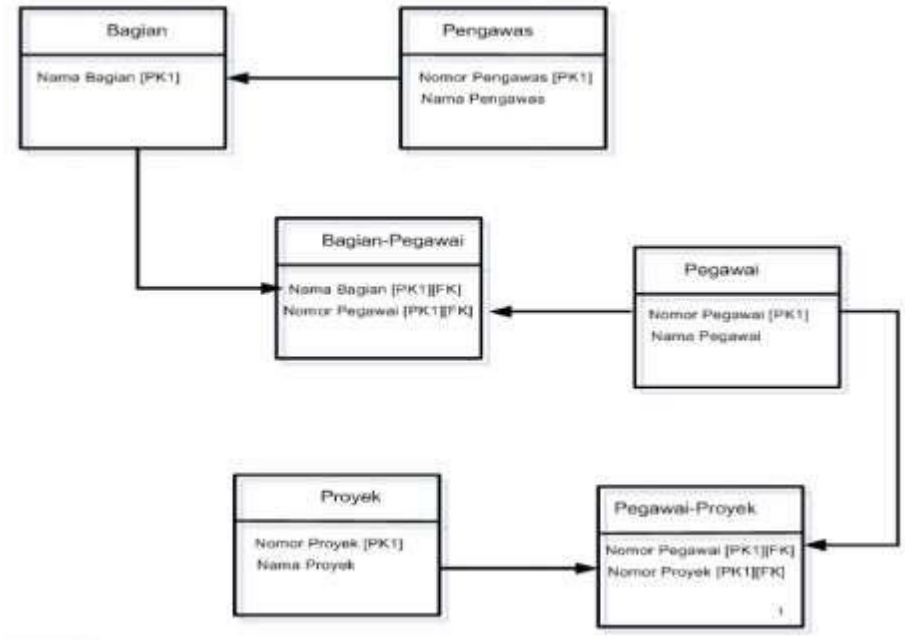
Gambar IV.22
Hasil Langkah ketujuh

Langkah 8 : Transformasi ERD ke LRS



Gambar IV.23
Transformasi ERD ke LRS

LRS Yang Terbemtuk :



Gambar IV.18
LRS yang Terbentuk

BAB V

TEKNIK NORMALISASI

5.1 Pengertian 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 bias 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 pada relation tersebut, yang tidak berakibat terjadinya ERROR atau INKONSESTENSI DATA, yang disebabkan oleh operasi-operasi tersebut.

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

5.2 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. 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 bias diinsert / 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 : 12100001 memutuskan untuk batal ikut matakulia 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 keslaahan 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 menjadi 75 menjadi 100, maka harus dilakukan beberapa kali modifikasi terhadap record-record mahasiswa yang mengambil matakuliah tersebut, agar data tetap konsisten.

Menurut (Ladjamudin, 2004) terdapat problem-problem pada relation yang sudah dinormalisasi, yaitu :

1. Performance problem

   Merupakan masalah terhadap performa database.

2. Referential Integrity Problem

   Masalah yang timbul terhadap referensi antar data-data diantara dua tabel atau lebih

5.3 Atribut dan Ketergantungan Fungsi

Beberapa konsep yang harus diketahui dalam Normalisasi, adalah :

1. Field/ Atribut Kunci

   Key Field / atribute kunci dalam database:

   a. Super key

   b. Candidate key

c. *Primary key*

d. *Alternate key*

e. *Foreign key.*

2. Kebergantungan Fungsi.

a. Ketergantungan Fungsional (Fungsional Dependent)

Keterkaitan antar hubungan antara 2 atribute pada sebuah relasi. Dituliskan dengan cara : A → B, yang berarti : Atribute B fungsionality Dependent terhadap atribute A atau Isi (value) atribute A menentukan isi atribute B Definisi dari functional dependent : Diketahui sebuah relasi R, atribute Y dari R adalah FD pada atribute 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.

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: Atribute Y pada relasi R adalah FFD pada atribute 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 atribute biasa pada suatu relasi tetapi menjadi kunci pada relasi lain

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

5.4 Bentuk Normalisasi

Aturan-aturan normalisasi dinyatakan dengan istilah bentuk normal. Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi dalam basis data dan harus dipenuhi oleh relasi-relasi tersebut pada level-level normalisasi.

Beberapa level yang biasa digunakan pada normalisasi adalah:

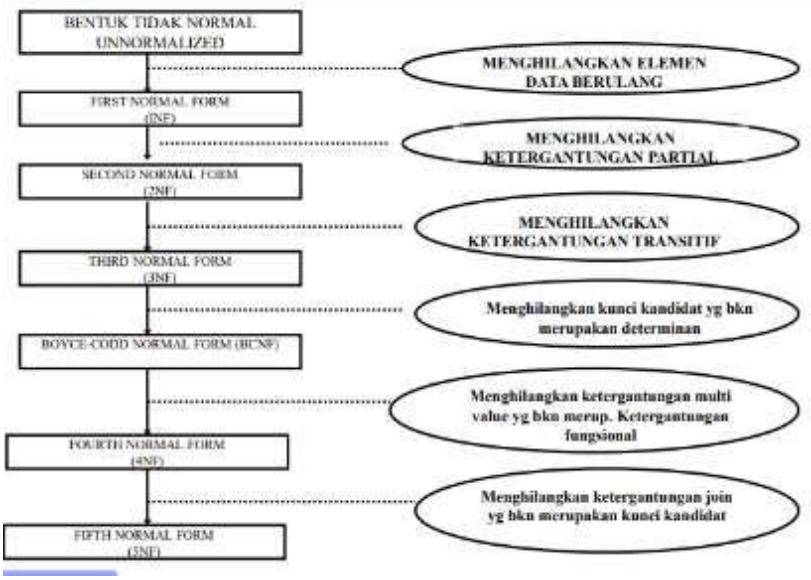
1. Bentuk normal pertama (1NF)
2. Bentuk normal kedua (2NF)
3. Bentuk normal ketiga (3NF)
4. Bentuk normal Boyce-Codd (BCNF)
5. Bentuk normal keepat (4NF)
6. Bentuk Normal kelima (5NF)

Akan dijelaskan di BAB selanjutnya.

BAB VI

TEKNIK NORMALISASI LANJUTAN

6.1 Langkah-Langkah Pembuatan Normalisasi



Gambar 6.1
Langkah-langkah pembuatan normalisasi

1. Bentuk Tidak Normal (*Unnormalized Form*)

   Bentuk tidak normal adalah bentuk tabel yang belum ternormalisasi. Tabel yang belum ternormalisasi adalah tabel yang memiliki atribut yang berulang. Menurut (Ladjamudin, 2004), bentuk tidak normal meupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti form tertentu, dapat saja data tidakmlengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan saat menginput.

2. Bentuk Normal Pertama (*First Normal Form* / 1NF)

Menurut (Ladjamudin, 2004), 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). Atom adalah zat terkecil yang masih memiliki sifat induknya, bila dipecah lagi maka ia tidak memiliki sifat induknya.

Syarat bentuk normal kesatu, antar lain:

a. Setiap data dibentuk dlaam flat file, data dibentuk dalam satu record demi satu record nilai dari field berupa "atomic value"
b. Tidak ada set atribut yang berulang atau bernilai ganda
c. Telah ditentukannya candidate key
d. Tiap atribut hanya memiliki satu pengertian.

3. Bentuk Normal Kedua

Menurut (Ladjamudin, 2004), 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.

Syarat bentuk normal kedua:

a. Bentuk data telah memnuhi kriteria bentuk normal kesatu
b. Atribut bukan kunci haruslah memiliki ketergantungan fungsional sepenuhnya pada kunci utama atau primary key.

4. Bentuk Normal Ketiga

Definisi bentuk normal ketiga: Suatu relasi dikatakan dalam bentuk normal ketiga (3NF) jika memiliki syarat, sebagai berikut :

a. Berada dalam bentuk normal kedua
b. 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, selruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap primary key di relasi itu saja..

5. Bentuk Normal Boyce Codd (BCNF)

Definisi bentuk normal Boyce-Codd: Suatu relasi disebut memenuhi bentuk normal Boyce-Codd 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.

6. Bentuk Normal Keempat

Bentuk normal keempat berkaitan dengan sifat Ketergantungan Banyak-Nilai (*Multivalued Depedency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional.

7. Bentuk Normal Kelima

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

6.2 Studi Kasus

1. Sistem Perpustakaan

Daftar Anggota Perpustakaan

Kode Anggota	Nama
A01	Surya
A02	Fitri
A03	Syahrur

Daftar Buku Perpustakaan

Kode Buku	Judul	Stok Buku
B01	Pemograman C++	10
B02	Membuat Aplikasi 30 Menit	15
B03	Cooking is Easy	15

Bukti Peminjaman Buku

Tanggal Pinjam : 10 Januari 2019 No. Anggota : A01
No. Pinjam : PJ01

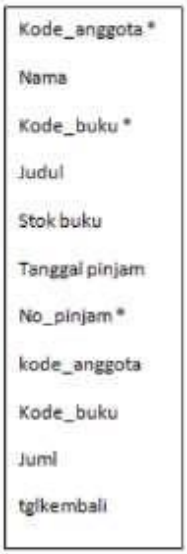
Kode Buku	Judul Buku	Jumlah Buku
B01	Pemograman C++	1
B02	Membuat Aplikasi 30 Menit	1
B03	Cooking is Easy	1

Tanggal Kembali : 13 Januari 2019

a. Bentuk TIdak Normal

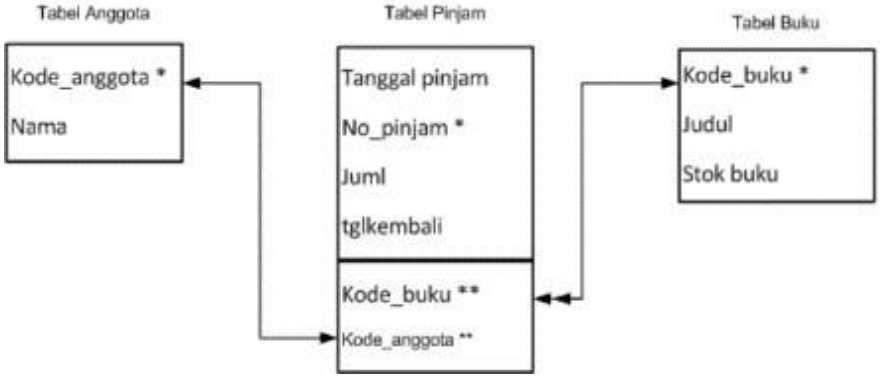
Gambar 6.2
Bentuk Tidak Normal

b. Bentuk Normal Kesatu



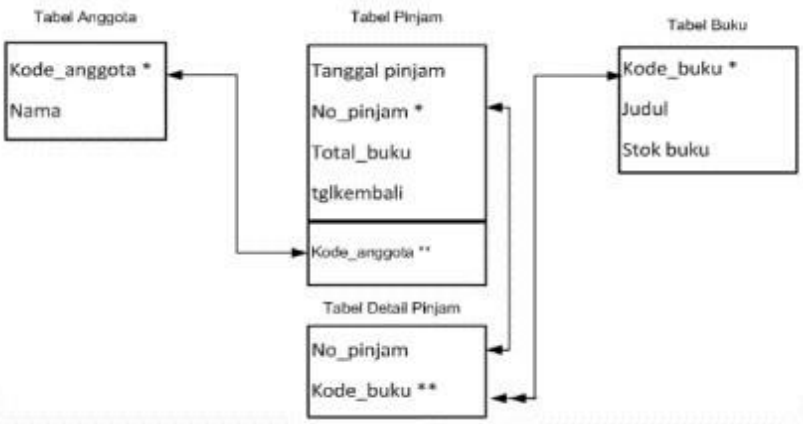
Gambar 6.3
Bentuk Normal Kesatu

c. Bentuk Normal Kedua



Gambar 6.4
Bentuk Normal Kedua

d. Bentuk Normal Ketiga



Gambar 6.5
Bentuk Normal Ketiga

2. Sistem Pembelian

PT. SANTA PURI
Jalan senopati 11
yogyakarta

FAKTUR PEMBELIAN BARANG

Kode Suplier : G01
Nama Suplier : Gobel Nustra

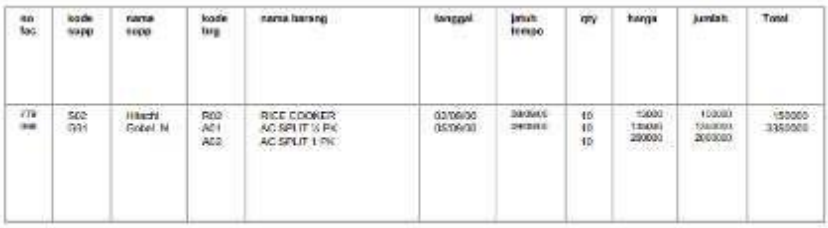
Tanggal : 05/09/2000
Nomor : 998

Kode	Nama Barang	Qty	Harga	Jumlah
A01	AC SPLIT ½ PK	10.0	135,000	1,350,000
A02	AC SPLIT 1 PK	10.0	200,000	2,000,000
			Total Faktur	3,350,000

Jatuh tempo faktur : 09/09/2000

Gambar 6.6
Faktur Pembelian

a. Bentuk Tidak Normal



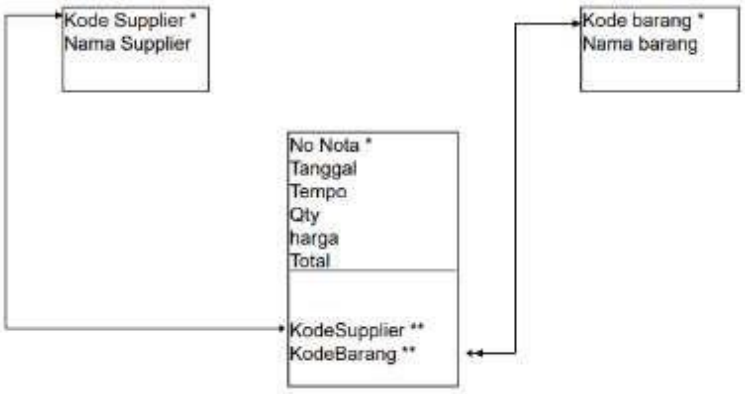
Gambar 6.7
Bentuk Tidak Normal

b. Bentuk Normal Kesatu

nofac	kode supp	nama supp	Kode brg	nama barang	tanggal	jatuh tempo	qty	harga	jumlah	Total
779	S02	Hitachi	R02	RICE COOKER	02/09/00	08/09/00	10	15000	150000	150000
998	G01	Gobel N	A01	AC SPLIT ½ PK	05/09/00	09/09/00	10	135000	1350000	3350000
998	G01	Gobel N	A02	AC SPLIT 1 PK	05/09/00	09/09/00	10	200000	2000000	3350000

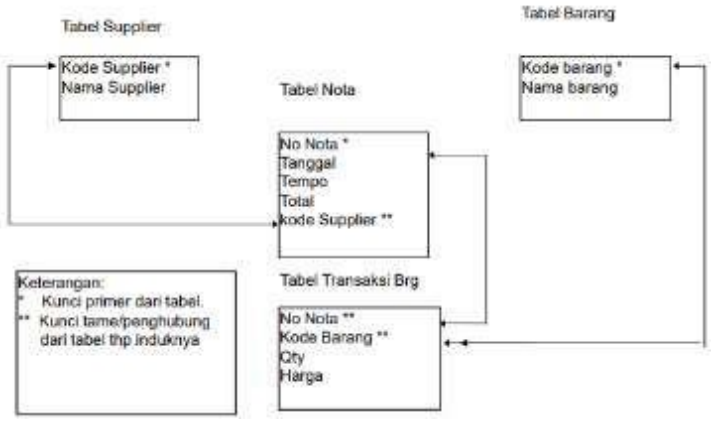
Gambar 6.8
Bentuk Normal Kesatu

c. Bentuk Normal Kedua



Gambar 6.9
Bentuk Normal Kedua

d. Bentuk Normal Ketiga



Gambar 6.10
Bentuk Normal Ketiga

BAB VII

BAHASA QUERY FORMAL

7.1 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. 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.

Menurut (Ladjamudin, 2004) dalam bahasa Query Formal, ada dua dasar pembentukan bahasa Query, yaitu:

1. 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

Merupakan bahasa manipulasi teoritis yang non procedural. Artinya bahasa ini mengekspresikan “what to do or what characteristics are required”dan tidak mengekspresikan “how to do it” seperti pada bahasa query 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 berniali benar atau salah.

7.2 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.

Contoh :

Tabel dibawah ini adalah contoh untuk mengerjakan perintah-perintah Aljabar relasional:

Relasi 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
547	APSI	4	200109601
305	PEMR. PASCAL	4	200703073
544	DISAIN GRAFIS	2	200010490

Relasi MAHASISWA

NIM	NAMA_MHS	ALAMAT	J_KEL
1105090222	HAFIDZ	DEPOK	LAKI-LAKI
1105091002	RAFFA	DEPOK	LAKI-LAKI
1105095000	NAIA	DEPOK	PEREMPUAN
1104030885	ARIF	P.LABU	LAKI-LAKI
1206090501	LENI	KMP. MELAYU	PEREMPUAN
1206090582	WAHYUNI	TANGERANG	PEREMPUAN
1205097589	ARIS	DEPOK	LAKI-LAKI
1106094586	YANI	CILEDUG	PEREMPUAN
110709	BAMBANG	SALEMBA	LAKI-LAKI

Relasi REGISTRASI

KD_MK	NIM
360	1105090222
545	1206090501
547	1105095000

Relasi DOSEN

NIP	NAMA_DOS	GAJI
199910486	BILLY	3500000
200109655	MARDIANA	4000000
200209817	INDRIYANI	4500000
200209818	SURYANI	4250000
200109601	DWINITA	3500000
200703073	MALAU	2750000
200010490	IRFIANI	3500000

a. Operator Dasar

1) Selection (σ) Lower Case Omega

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

Sintaks : σ predicate/syarat $^{(tabel)}$;

Contoh :

a) Mencari tuple-tuple dari MAHASISWA yang memiliki jenis kelamin laki-laki

Query : σ J_KEL="LAKI-LAKI" $^{(MAHASISWA)}$;

Hasil query diatas sebagai berikut :

NIM	NAMA_MHS	ALAMAT	J_KEL
1105090222	HAFIDZ	DEPOK	LAKI-LAKI
1105091002	RAFFA	DEPOK	LAKI-LAKI
1104030885	ARIF	P.LABU	LAKI-LAKI
1205097589	ARIS	DEPOK	LAKI-LAKI
110709	BAMBANG	SALEMBA	LAKI-LAKI

b) Tampilkan data mata kuliah yang memiliki kode 360 atau yang memilki sks 4.

Query : σ KD_MK="360" V SKS=4 $^{(MATAKULIAH)}$;

Hasil query diatas sebagai berikut :

KD_MK	NAMA_MK	SKS	NIP
207	LOGIKA&ALGO	4	199910486

360	SISTEM BASIS DATA	3	200209817
547	APSI	4	200109601
305	PEMR. PASCAL	2	200703073

2) Projection (π)

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

Sintask : π A1,A2,A3…..An $^{(\text{nama tabel})}$; dimana a adalah Atribut.

Contoh : Tampilkan nama beserta gaji dari dosen

Query : π nama_dos,gaji $^{(\text{DOSEN})}$;Hasil

dari query diatas adalah :

NAMA_DOS	GAJI
BILLY	3500000
MARDIANA	4000000
INDIYANI	4500000
SURYANI	4250000
DWINITA	3500000
MALAU	2750000
IRGIANI	3500000

3) Cartesian 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 symbol X.

Sintaks : R1 X R2, dimana R adalah relasi.

Operator Cartesian product akan merangkaikan setiap tuple dari R1 dan setiap tuple dari R2, sehingga jika R1 terdiri dari n tuple dan R2 terdiri dari m tuple, maka hasil relasi R = R1 X R2 akan terdiri dari mn tuple.

Sebagai ilustrasi :

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

Misal diatas, ibu ada dua himpunan yaitu A dan B, hasil dari Cartesian product dari AXB adalah seperti diatas. Jadi seperti di relasikan satu-satu yaa. Misal 1 dengan 5 kemudian 7, hasilnya (1,5) dan (1,7), begitupun untuk 2 dan 3. Jika ini diterapkan dengan table, maka operasi perhitungannya akan sama.

Contoh :

Hasil Cartesian Product antara Tabel Mahasiswa dengan Tabel Registrasi (Mahasiswa X Registrasi);

RELASI : MAHASISWA

NIM	NAMA_MHS	ALAMAT	J_KEL
1105090222	HAFIDZ	DEPOK	LAKI-LAKI
1105091002	RAFFA	DEPOK	LAKI-LAKI
1105095000	NAIA	DEPOK	PEREMPUAN
1104030885	ARIF	P.LABU	LAKI-LAKI
1206090501	LENI	KMP. MELAYU	PEREMPUAN
1206090582	WAHYUNI	TANGERANG	PEREMPUAN
1205097589	ARIS	DEPOK	LAKI-LAKI
1106094586	YANI	CILEDUG	PEREMPUAN
110709	BAMBANG	SALEMBA	LAKI-LAKI

RELASI : REGISTRASI

KD_MK	NIM
360	1105090222
545	1206090501
547	1105095000

(MAHASISWA X REGISTRASI)

NIM	NAMA_MHS	ALAMAT	J_KEL	KD_MK	NIM
1105090222	HAFIDZ	DEPOK	LAKI-LAKI	360	1105090222
1105090222	HAFIDZ	DEPOK	LAKI-LAKI	545	1206090501
1105090222	HAFIDZ	DEPOK	LAKI-LAKI	547	1105095000
1105091002	RAFFA	DEPOK	LAKI-LAKI	360	1105090222
1105091002	RAFFA	DEPOK	LAKI-LAKI	545	1206090501
1105091002	RAFFA	DEPOK	LAKI-LAKI	547	1105095000
1105095000	NAIA	DEPOK	PEREMPUAN	360	1105090222
1105095000	NAIA	DEPOK	PEREMPUAN	545	1206090501
1105095000	NAIA	DEPOK	PEREMPUAN	547	1105095000
1104030885	ARIF	P.LABU	LAKI-LAKI	360	1105090222
1104030885	ARIF	P.LABU	LAKI-LAKI	545	1206090501
1104030885	ARIF	P.LABU	LAKI-LAKI	547	1105095000
1206090501	LENI	KMP. MELAYU	PEREMPUAN	360	1105090222
1206090501	LENI	KMP. MELAYU	PEREMPUAN	545	1206090501
1206090501	LENI	KMP. MELAYU	PEREMPUAN	547	1105095000
1206090582	WAHYUNI	TANGERANG	PEREMPUAN	360	1105090222
1206090582	WAHYUNI	TANGERANG	PEREMPUAN	545	1206090501
1206090582	WAHYUNI	TANGERANG	PEREMPUAN	547	1105095000
1205097589	ARIS	DEPOK	LAKI-LAKI	360	1105090222
1205097589	ARIS	DEPOK	LAKI-LAKI	545	1206090501
1205097589	ARIS	DEPOK	LAKI-LAKI	547	1105095000
1106094586	YANI	CILEDUG	PEREMPUAN	360	1105090222
1106094586	YANI	CILEDUG	PEREMPUAN	545	1206090501
1106094586	YANI	CILEDUG	PEREMPUAN	547	1105095000
110709	BAMBANG	SALEMBA	LAKI-LAKI	360	1105090222

110709	BAMBANG	SALEMBA	LAKI-LAKI	545	1206090501
110709	BAMBANG	SALEMBA	LAKI-LAKI	547	1105095000

Tampilkan NAMA_MHS (dari Tabel Mahasiswa), KD_MK(dari Tabel Registrasi) dimana Alamat = “DEPOK” atau J_KEL=”LAKI-LAKI”

Maka query nya adalah :

Π NAMA_MHS, KD_MK (σ ALAMAT = “DEPOK” ˅ J_KEL = “LAKI-LAKI” ˄ MAHASISWA.NIM = REGISTRASI.NIM)

(MAHASISWA X REGISTRASI)

Hasil Query diatas adalah :

Kita cari terlebih dahulu, NIM yang sama antara table Mahasiswa dengan table registrasi, hasilnya sebagai berikut :

NIM	NAMA_MHS	ALAMAT	J_KEL	KD_MK	NIM
1105090222	HAFIDZ	DEPOK	LAKI-LAKI	360	1105090222
1105095000	NAIA	DEPOK	PEREMPUAN	547	1105095000
1206090501	LENI	KMP. MELAYU	PEREMPUAN	545	1206090501

Kemudian, baru kita tampilkan NAMA_MHS beserta KD_MK nya, yang memiliki alamat di depok atau jenis kelaminnya Laki-laki, hasilnya sebagai berikut :

NAMA_MHS	KD_MK
HAFIDZ	360
NAIA	547

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 RUS={ X I X E R atau X E S}.

Contoh :

Sintaks yang digunakan dalam operasi union ini adalah :

$$\mathbf{R \cup S = \{x \mid x \in R \text{ atau } X \in S\}}$$

Operasi ini dapat dilaksanakan apabila R dan S mempunyai atribut yang sama sehingga jumlah komponennya sama. Jadi yang akan kita gabungkan dari tabel-tabel tersebut, harus memiliki atribut yang sama dalam table tersebut.

Tabel Dosen

KODE_DOS	NAMA_DOS	ALAMAT_DOS	KOTA
SY	Syamsudin, S.Si	Jl. Suci	Bekasi
FS	Farida Syarif, Ir	Jl. Tenteram	Jakarta

Tabel Mahasiswa

NIM	NAMA_MHS	ALAMAT_MHS	KOTA	TGL_LHR
980001	Ali Akbar	Jl. Merdeka	Bogor	02-01-1979
980002	Budi Haryanto	Jl. Gajah Mada	Jakarta	06-10-1978

Jika dilakukan query sebagai berikut :

$\pi KOTA^{(Mahasiswa)} \cup \pi KOTA^{(Dosen)}$

Hasil dari Query diatas adalah:

KOTA
Bogor
Jakarta
Bekasi
Jakarta

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. R – S = { X I X E R dan X E S }.

Operasi ini dapat dilaksanakan apabila R dan S mempunyai atribut yang tidak sama yang akan ditampilkan, artinya adalah atribut R yang tidak ada di S akan ditampilkan, sedangkan atribut yang sama tidak ditampilkan.

Contoh :

Tabel Kuliah_S1

Kode_kul	Nama_kul	Sks	semester
IF-110	Pemrograman I	3	1
IF-221	Struktur Data	3	2
IF-310	Basis Data	4	3
IF-320	Pemrograman II	3	3
IF-411	Sistem Basis Sata	3	4
IF-423	Sistem Pakar	2	4

Tabel Kuliah_D3

Kode_kul	Nama_kul	Sks	semester
IF-110	Pemrograman I	3	1
IF-120	Aplikasi Akuntansi	2	1
IF-221	Struktur Data	3	2
IF-310	Basis Data	4	3

Jika diberikan query sebagai berikut :

πnama_kul (Kuliah_S1) - πnama_kul (Kuliah_D3), maka hasilnya adalah :

Nama_kul
Pemrograman II
Sistem Basis Data
Sistem Pakar

Contoh lain :

Tampilkan nama dari mahasiswa yang tinggal di depok tetapi bukan berjenis kelamin perempuan.

Query I : tampilkan nama yang tinggal di depok πnama_mhs(σalamat="DEPOK" (MAHASISWA)) ;

Query II : tampilkan nama yang berjenis kelamin perempuan πnama_mhs(σj_kel ="PEREMPUAN" (MAHASISWA)) ;

Tampilkan query I minus query II : πnama_mhs(σalamat="DEPOK"(MAHASISWA)) - πnama_mhs(σj_kel="PEREMPUAN" (MAHASISWA));

b. Operator Tambahan

Relasi R1

No_Id	No_Boat	Tanggal
22	101	10/10/96
580	103	11/12/96

Relasi S1

No_Id	Nama	Rating	Umur
22	Dustin	7	45
31	Andi	8	55.5
58	Rara	10	35

Relasi S2

No_Id	Nama	Rating	Umur
28	Ana	9	35
31	Andi	8	55.5
44	Rika	5	35
58	Rara	10	35

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.

Sintaks : R = R1∩ R2; dan akan menghasilkan relasi R dengan elemen yang terdapat di R1 dan juga terdapat di R2.

Contoh : S1 ∩ S2

Hasilnya :

No_Id	Nama	Rating	Umur
58	Rara	10	35

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	Dustin	7	45	58	103	11/12/96
31	Andi	8	55.5	58	103	11/12/96

3) NATURAL JOIN

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

Contoh : $S1 \theta_{No\_id} R1$

Hasilnya :

No_Id	Nama	Rating	Umur	No_Boat	Tanggal
22	Dustin	7	45	103	11/12/96
58	Rara	10	35	103	11/12/96

4) DIVISION

Merupakan operasi pembagian atas tuple-tuple dari 2 relation.

Contoh:

A

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

B

Pno
P2

A/B

Sno
S1
S2

BAB VIII

BAHASA QUERY TERAPAN

8.1 *Structured Query Language* (SQL)

1. Pengenalan SQL

Bahasa Query Formal yang sudah kita bahas sebelumnya, menjadi dasar dalam pembentukan bahasa query terapan. Dengan bahasa query formal itulah, algoritma dan sintaks dari ekspresi-ekspresi dalam bahasa query terapan disusun. Penguasaan terhadap bahasa query formal akan sanagat memudahkan kita dalam mempelajari dan menguasai pemakaian bahasa query terapan.

Bahasa query yang paling popular tentu saja adalah SQL (*Sturctured Query Language*), karena bahasa ini diakomodasi oleh hampir semua DBMS. Menurut (Fathansyah, 2012) menyampaikan bahwa 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 mudah dipelajari karena merupakan bahasa non procedural, cukup menspesifikasikan informasi apa yang dibutuhkan daripada bagaimana mendapatkannya.

2. SQL Sebagai Subbahasa

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 : pendefinisian struktur

data, pebgubahan data, pengaturan sekuritasn, dan lain-lain. Terkadang SQL dikatakan sebagai subbahasa data. Adapun alas an SQL dikatanakn sebagai subbahasa data adalah 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.

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.

3. Pengelompokkan SQL

a. *Data Definition Language* (DDL)

Menurut (Ladjamudin, 2004), DDL merupakan 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.

1) CREATE

a) Pembuatan Database

Sintaks : CREATE DATABASE nama_database

Nama Database adalah yang dapat mewakili suatu kejadian dapat berupa nama organisasi atau perusahaan.

Contoh : Buat database dengan nama KAMPUS

Query : CREATE DATABASE KAMPUS

b) Pembuatan Tabel

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

Keterangan :

(1) Nama tabel merupakan nama tabel yang baru, panjangnya tidak dapat lebih dari 8 karakter, tidak memakai spasi, berisi huruf maupun angka.

(2) Nama Kolom adalah nama untuk kolom yang baru, panjangnya tidak dapat lebih dari 10 karakter, tidak memiliki spasi, berisi huruf dan angka.

(3) Tipe Data adalah jenis data yang nilainya akan dimasukkan dalam kolom yang telah ditentukan.

Contoh :

(a) Char (n) : karakter string sebanyak N karakter, max 254 char, dan isinya ditulis dengan tanda (‘) atau (“)

(b) Integer, digunakan untuk bilangan bulat sebanyak 11 digit

(c) Small int, untuk bilangan angka sebanyak 6 digit

(d) Decimal(p,q), untuk bilnagan angka sebanyak p digit dengan tempat desimal q

(e) Float(x,y), untuk bilangan angka (floating point sebanyak x digit dengan y digit dari tidak desimal)

(4) Sebagai tambahan, setiap kolom pada pendefinisian tabel dapat dilengkapi dengan UNIQUE, NULL, NOT NULL dan NOT UNIQUE.

(a) NULL, menyatakan bahwa nilai kolom bisa tidak diisi (default)

(b) NOT NULL, menyatakan bahwa nilai kolom harus diisi

(c) UNIQUE, menyatakan bahwa nilai pada kolom tidak boleh ada yang sama

(d) NOT UNIQUE, menyatakan bahwa nilai pada kolom boleh kembar

Contoh : Buat struktur tabel dengan nama tabel Mahasiswa dengan data NIM char(8), NAMA char(25), ALAMAT char(30)

Query : CREATE TABLE Mahasiswa (NIM char(8) not null, NAMA char(25) notnull, ALAMAT char(30) notnull)

Perintah ini maksudnya adalah membuat tabel Mahasiswa dimana NIM wajib diisi, NAMA wajib diisi, ALAMAT wajib diisi.

c) Pembuatan Index

Indeks biasa diciptakan dengan tujuan sebagai berikut :

(1) Indeks dapat meningkatkan kinerja

(2) Indeks menjamin bahwa suatu kolom bersifat unik.

Dengan adanya indeks, maka pencarian suatu data yang berdasarkan kolom yang diindeks akan dapat dilakukan dengan cepat. Namun kelebihan ini tentu saja juga dibayar dengan suatu kelemahan. Pengindeksan memperlambat proses penambahan dan penghapusan baris pada tabel, karena saat terjadi penambahan atau penghapusan baris, indeks perlu diperbaharui.

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

Keterangan :

(1) Unique adalah pilihan perincian yang dapat digunakan untuk menguatkan nilai data didalam kolom nama index menjadi unik.

(2) Nama_index adalah nama index yang akan diciptakan

(3) Nama_Tabel adalah nama tabel yang berisi kolom index akan dibuat (nama tabel yang akan mengindeks)

(4) Nama_kolom (asc atau dec) adalah nama dari kolom tempat index akan dibuat.(nama kolom untuk mengindeks)

Contoh : Buat index data Mahasiswa berdasarkan NIM dengan nama MHSIDX Dimana NIM tidak boleh sama

Query : 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] ;

Keterangan :

(1) nama_view harus dimulai dengan huruf, bilangan, dan garis bawah, panjangnya harus kurang dari 9 huruf

(2) Nama_kolom merupakan sebuah nama kolom optimal yang harus diberikan kepada satu kolom view.

(3) Pernyataan SELECT, berupa pernyataan select apa saja kecuali bahwa :

   (a) Pernyataan tersebut tidak boleh berisi sebuah klause UNION

   (b) Tidak boleh berisi klausa ORDER BY

   (c) Tidak boleh berisi klausa SAVE TO TEMP

(4) [With Check Option] adalah klausa optimal yang menyebabkan semua update dan penyisipan ke view akan diperiksa untuk mengetahui apakah semua itu memenuhi definisi view.

Contoh : Buat view dengan nama MHSVIEW yang berisi semua data mahasiswa

Query : CREATE VIEW MHSVIEW AS SELECT * FROM Mahasiswa

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

Hasil dari bentuk drop tabel adalah

a) Semua record dalam tabel akan dihapus

b) Index dan view di tabel akan hilang

c) Deskripsi tabel akan hilang.

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, menambahkunci, mengahapus kolom yang ada.

Sintaks :

ALTER TABLE nama_tabel ADD nama_kolom jenis_kolom

[FIRST | AFTER nama_kolom]

CHANGE [COLUMN] oldnama newnama

MODIFY nama_kolom jenis kolom, …

DROP nama_kolom

RENAME newnama_tabel

Keterangan :

a) ADD digunakan untuk menambah kolom baru

Sintaks :

ALTER TABLE nama_tabel ADD nama_kolom jenis_kolom

[FIRST | AFTER nama_kolom];

FIRST : Penambahan kolom baru diletakkan pada urutan kolom pertama

AFTER : Penambahan kolom baru diletakkan setelah kolom yang ditunjuk

Jika ingin menambah kolom Primary Key maka Sintaksnya :

ALTER TABLE nama_tabel ADD PRIMARY KEY nama_kolom;

b) CHANGE digunakan untuk merubah nama kolom

Sintaks :

ALTER TABLE nama_tabel CHANGE [COLUMN] oldnama newnama ;

c) MODIFY digunakan untuk merubah tipe data kolom

Sintaks :

ALTER TABLE nama_tabel MODIFY nama_kolom jenis kolom ;

d) DROP digunakan untuk menghapus nama kolom

Sintaks :

ALTER TABLE DROP nama_kolom ;

e) RENAME digunakan untuk mengganti nama tabel.

Sintaks :

ALTER TABLE newnama_tabel;

Contoh :

(1) Tambahkan kolom JKEL dengan panjang 1 char pada tabel Mahasiswa

Query : ALTER TABLE Mahasiswa ADD JKEL char(1);

(2) Ubah panjang kolom JKEL menjadi 15 char

Query : ALTER TABLE Mahasiswa MODIFY COLUMN JKEL char(15);

(3) Hapus kolom JKEL dari data table MHS

Query : ALTER TABLE Mahasiswa DROP JKEL;

b. Data Manipulation Language (DML)

Menurut (Ladjamudin, 2004), DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan. Yang termasuk dalam kelompok DML adalah SELECT (memilih data), INSERT (menambah data), DELETE (menghapus data) dan UPDATE (menguah data)

1) INSERT

Digunakan untuk penambahan record baru kedalam sebuah tabel.

Sintaks: INSERT INTO Nama_tabel [(nama_kolom1,…)] values (nilai atribut1, …)

Contoh: Masukan data Mahasiswa dengan Nim 10296832, Nama Nurhayati beralamat di Jakarta

Query : INSERT INTO Mahasiswa (Nim, Nama, Alamat) values (“10296832”,”Nurhayati”,“Jakarta”);

2) DELETE

Digunakan untuk menghapus record dari sebuah tabel.

Sintaks: DELETE FROM nama_table WHERE kondisi

Keterangan :

a) nama_tabel : nama tabel yang baris2 nya ingin dihapus

b) WHERE, klausa yang menentukan baris2 yang akan dihapus

Contoh: Hapus data Mahasiswa yang mempunyai NIM “21198002”

Query : DELETE FROM Mahasiswa WHERE NIM=” 21198002”

3) UPDATE

Digunakan untuk mengubah nilai atribut pada suatu record dari sebuah tabel.

Sintaks:

UPDATE nama_tabel SET nama_kolom = value_1 WHERE kondisi ;

Keterangan :

a) nama_tabel adalah nama tabel yang akan di update

b) SET untuk menentukan kolom yang akan diubah dan nilai penggantinya

c) WHERE kondisi adalah klausa yang menetapkan baris2 yang akan di update

Contoh: Ubah alamat menjadi “Depok” untuk mahasiswa yang memiliki NIM “10296832”

Query : UPDATE Mahasiswa SET ALAMAT=”Depok” WHERE NIM=” 10296832”;

4) SELECT

Digunakan untuk menampilkan isi tabel.

Sintaks: SELECT [DISTINCT | ALL] nama_kolom FROM nama_tabel [WHERE condition] [GROUP BY column_list] [HAVING condition] [ORDER BY column_list [ASC | DESC]]

Keterangan :

a) SELECT, memilih data yang akan ditampilkan berdasarkan atribut

b) DISTINCT, menghilangkan duplikasi

c) FROM, mendefinisikan tabel yang akan digunakan dalam query

d) WHERE, menentukan syarat yang akan dipilih

e) GROUP BY, mengelompokkan data yang mempunyai nilai sama

f) HAVING, syarat data yang dikelompokkan digunakan bersama GROUP BY

g) ORDER BY, mengurutkan data

Contoh :

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Tabel Nilai

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Tabel MataKuliah

KD_MK	NAMA_MK	SKS
KK021	Sistem Basis Data	2
KD132	Sistem Informasi Manajemen	3
KU122	Pancasila	2

Gambar 8.1 Contoh Tabel Mahasiswa, Tabel Nilai. Tabel Matakuliah

(1) Tampilkan semua data Mahasiswa

Query : SELECT NIM,NAMA,ALAMAT FROM Mahasiswa;

Atau SELECT * FROM Mahasiswa;

Hasilnya :

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor

(2) Tampilkan Mata Kuliah yang SKS nya 2

Query : SELECT NAMA_MK FROM MataKuliah WHERE SKS=2

Hasilnya :

NAMA_MK
Sistem Basis Data
Pancasila

(3) Tampilkan semua data nilai dimana nilai MID lebih besar sama dengan 60 atau nilai finalnya lebih besar 75.

Query : SELECT * FROM Nilai WHERE MID >= 60 OR FINAL > 75

Hasilnya :

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
41296525	KU122	90	80
21196353	KU122	75	75

(4) Menampilkan NIM beserta NAMA yang diuruttkan berdasarkan NAMA

Query : SELECT NIM, NAMA FROM Mahasiswa ORDER BY NAMA;

Hasilnya:

NIM	NAMA
10296126	Astuti
31296500	Budi
10296001	Fintri
21198002	Julizar
10296832	Nurhayati
50096487	Pipit
41296525	Prananigrum
21196353	Quraish

(5) Mengelompokkan Alamat Mahasiswa

Query : SELECT ALAMAT FROM Mahasiswa GROUP BY ALAMAT;

Hasilnya :

ALAMAT
Bekasi
Bogor
Depok
Jakarta

(6) Query : SELECT NAMA, ALAMAT FROM Mahasiswa GROUP BY ALAMAT HAVING COUNT (ALAMAT) > 1;

Klausa HAVING disediakan untuk mendukung klausa GROUP BY. Kegunaannya adalah untuk menentukan kondisi bagi GROUP BY. Kelompok yang memenuhi kondisi HAVING saja yang akan dihasilkan. Perintah dengan HAVING diatas hanya akan menghasilkan baris untuk ALAMAT yang NAMA mahasiswa lebih dari satu.

Hasilnya :

ALAMAT
Bogor
Depok
Jakarta

c. Data Access (Data Control Language / DCL)

Menurut (Ladjamudin, 2004), DCL berisi perintah-perintah untuk mengendalikan pengaksesan data. Pengendalian dapat dilakukan per tabel, per kolom maupun per operasi. Yang termasuk dalam kelompok DCL atau Data Access adalah GRANT (memberi kendali pengaksesan data) dan REVOKE (mencabut kemampuan pengaksesan data)

1) GRANT

Digunakan untuk memberikan hak akses.

Sintaks :

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];

Contoh: Berikan hak akses kepada Adi untuk menampikan nilai final test pada tabel Nilai.

Query : GRANT SELECT (FINAL) ON NILAI TO ADI;

2) REVOKE

Digunakan untuk mencabut kembali hak akses yang sudah diberikan.

Sintaks :

REVOKE hak_akses ON nama_db FROM nama_pemakai ; atau

REVOKE hak_akses ON nama_tabel FROM nama_pemakai ;

Contoh: Tarik kembali dari Adi hak akses untuk menampilkan nilai final test

Query : REVOKE SELECT (FINAL) ON NILAI FROM ADI;

d. Data Integrity

Data Integrity merupakan perintah yang digunakan untuk mengembalikan data sebelum terjadi kerusakan.Yang termasuk di dalam kelompok Data Integrity adalah RECOVER TABLE.

Sintaks :

RECOVER TABLE nama_tabel;

Contoh : Kembalikan keadaan data mahasiswa seperti pada saat sebelum terjadi kerusakan

Query : RECOVER TABLE MHS ;

e. Data Auxiliary

Data Auxiliary merupakan perintah yang digunakan untuk mengubah data maupun kolom pada tabel. Yang termasuk dalam kelompok Data auxiliary adalah SELECT INTO OUTFILE, LOAD, RENAME TABLE

1) SELECT … INTO OUTFILE ‘filename’

   Digunakan untuk mengekspor data dari tabel ke file lain.

   Sintaks :

   SELECT … INTO OUTFILE ‘Nama File’ [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']] ;

   Contoh: Ubah semua data mahasiswa ke bentuk ASCII dan disimpan ke file teks di directory/home/adi dengan pemisah antar kolom ‘|’

   Query : SELECT * FROM MHS INTO OUTFILE “/home/adi/teks” FIELDS TERMINATED BY “|”;

2) LOAD

   Digunakan untuk mengimpor data dari file lain ke tabel.

   Sintaks :

   LOAD DATA INFILE “ nama_path” INTO TABLE nama_tabel [nama_kolom] ; [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']];

Contoh: Memasukkan data-data dari file teks yang berada pada direktori "/home/adi" ke dalam tabel MHS_2. Dimana pemisah antara kolom dalam file teks adalah tab (\t) :

Query : LOAD FROM "/home/adi/teks" INTO MHS_2 FILELDS TERMINATED BY '\t';

3) RENAME TABLE

Digunakan untuk mengganti nama tabel.

Sintaks :

RENAME TABLE OldnamaTabel TO NewNamaTabel

Contoh : RENAME TABLE MHS TO MAHASISWA

BAB IX

BAHASA QUERY TERAPAN LANJUTAN

1. 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. Contoh, diketahui ada dua tabel yaitu Tabel Mahasiswa dan Tabel Nilai, sebagai berikut :

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Tabel Nilai

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Ada beberapa tipe Join, yaitu :

a. INNER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian.

Jika ada Query sebagai berikut :

Nilai INNER JOIN Mahasiswa ON Nilai.NIM = Mahasiswa.NIM;

Maka Hasil gabungannya adalah :

NIM	KD_MK	MID	FINAL	NIM	NAMA	ALAMAT
10296832	KK021	60	75	10296832	Nurhayati	Jakarta
10296126	KD132	70	90	10296126	Astuti	Jakarta
31296500	KK021	55	40	31296500	Budi	Depok
41296525	KU122	90	80	41296525	Prananigrum	Bogor
21196353	KU122	75	75	21196353	Quraish	Bogor
50096487	KD132	80	0	50096487	Pipit	Bekasi

Kemudian jika diberikan query sebagai berikut:

SELECT Nilai.NIM, Mahasiswa.NAMA, Nilai.KD_MK, Nilai.MID FROM Nilai INNER JOIN Mahasiswa ON Nilai.NIM = Mahasiswa.NIM;

Maka Hasilnya adalah :

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
21196353	Quraish	KU122	75
50095487	Pipit	KD132	80

b. LEFT JOIN atau LEFT OUTER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kiri.

Contoh : Jika ada query sebagai berikut :

Mahasiswa LEFT OUTER JOIN Nilai ON Nilai.NIM = Mahasiswa.NIM

Maka Hasil gabungannya adalah sebagai berikut :

NIM	NAMA	ALAMAT	NIM	KD_MK	MID	FINAL
10296832	Nurhayati	Jakarta	10296832	KK021	60	75
10296126	Astuti	Jakarta	10296126	KD132	70	90
31296500	Budi	Depok	31296500	KK021	55	40
41296525	Prananigrum	Bogor	41296525	KU122	90	80
50096487	Pipit	Bekasi	50096487	KD132	80	0
21196353	Quraish	Bogor	21196353	KU122	75	75
10296001	Fintri	Depok	NULL	NULL	NULL	NULL
21198002	Julizar	Jakarta	NULL	NULL	NULL	NULL

Kemudian jika diberikan query sebagai berikut :

SELECT Mahasiswa.NIM, Mahasiswa.NAMA, Nilai.KD_MK, Nilai.MID FROM Mahasiswa LEFT OUTER JOIN Nilai ON Nilai.NIM = Mahasiswa.NIM; Maka Hasilnya :

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
50096487	Pipit	KD132	80
21196353	Quraish	KU122	75
10296001	Fintri	NULL	NULL
21198002	Julizar	NULL	NULL

c. RIGHT JOIN atau RIGHT OUTER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kanan.

Contoh, jika ada query seperti dibawah ini :

Nilai RIGHT OUTER JOIN Mahasiswa ON Nilai.NIM = Mahasiswa.NIM;

Maka Hasilnya :

NIM	KD_MK	MID	FINAL	NIM	NAMA	ALAMAT
10296832	KK021	60	75	10296832	Nurhayati	Jakarta
10296126	KD132	70	90	10296126	Astuti	Jakarta
31296500	KK021	55	40	31296500	Budi	Depok
41296525	KU122	90	80	41296525	Prananigrum	Bogor
50096487	KD132	80	0	50096487	Pipit	Bekasi
21196353	KU122	75	75	21196353	Quraish	Bogor
NULL	NULL	NULL	NULL	10296001	Fintri	Depok
NULL	NULL	NULL	NULL	21198002	Julizar	Jakarta

Kenudian, jika diberikan query sebagai berikut :

SELECT Mahasiswa.NIM, Mahasiswa.NAMA, Nilai.KD_MK, Nilai.MID FROM Nilai RIGHT OUTER JOIN Mahasiswa ON Nilai.NIM = Mahasiswa.NIM;

Maka Hasilnya :

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
50096487	Pipit	KD132	80
21196353	Quraish	KU122	75
10296001	Fintri	NULL	NULL
21198002	Julizar	NULL	NULL

2. Fungsi Agregat

Disamping 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

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

Contoh : Menghitung jumlah record mahasiswa dari tabel.

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Query : SELECT COUNT(*) FROM MAHASISWA;

Hasilnya: 8

b. SUM

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

Contoh : Menghitung total sks dari tabel MATAKULIAH.

KD_MK	NAMA_MK	SKS
KK021	Sistem Basis Data	2
KD132	Sistem Informasi Manajemen	3
KU122	Pancasila	2

Query : SELECT SUM(SKS) AS 'TOTAL SKS' FROM MATAKULIAH;

Hasilnya ;

Total SKS
7

c. AVG

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

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

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Query : SELECT AVG(FINAL) AS 'FINAL' FROM Nilai;

Hasilnya :

FINAL
60

d. MIN

Digunakan untuk menghitung nilai minimal dalam sebuah kolom.

Contoh : Menghitung Minimal Nilai Final dari Tabel Nilai

Query : SELECT MIN(FINAL) FROM Nilai;

Hasilnya :

FINAL
0

e. MAX

Digunakan untuk menghitung nilai maksimum dalam sebuah kolom

Contoh : Menghitung maksimal nilai MID dari tabel Nilai

Query : SELECT MAX(MID) FROM Nilai;

Hasilnya :

MID
90

3. SubQuery

a. Pengenalan Subquery

Menurut (Ladjamudin, 2004), subquery berarti query didalam query. Dengan menggunakan subquery, maka hasil dari query akan menjadi bagian dari query diatasnya. Subquery terletak didalam klausa WHERE atau HAVING. Pada klausa WHERE, subquery digunakan untuk memilih baris-baris tertentu, yang kemudian digunakan untuk query. Sedangkan pada klausa HAVING, subquery digunakan untuk memilih kelompok baris, yang kemudian digunakan oleh query.

Tabel Pengarang

KD_PENG	NAMA	KOTA
P01	Ashadi	Yogyakarta
P02	Rian	Yogyakarta
P03	Suadi Marwan	Bandung
P04	Siti Halimah	Solo
P05	Amir Hamzah	Kudus
P06	Suparman	Jakarta
P07	Jaja	Bandung
P08	Sama	Bogor

Tabel Buku

KD_BUKU	JUDUL	KD_PENG
B01	Pengantar Basis Data	P02
B02	Pemrograman C++	P01
B03	Sistem Pakar	P03
B04	Visual C++	P05
B05	Pemrograman Pascal	P01
B06	Pemrograman Delphi	P07

Sebagai contoh, diinginkan untuk menampilkan daftar kode pengarang dan nama pengarang (berdasarkan tabel Pengarang) yang kode pengarangnya tercantum pada tabel Buku. Maka query nya :

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

Pada query diatas, SELECT KD_PENG FROM BUKU disebut dengan subquery, sedangkan SELECT KD_PENG, NAMA berkedudukan sebagai query.Hasil dari query diatas :

KD_PENG	NAMA
P01	Ashadi
P02	Rian
P03	Suadi Marwan
P05	Amir Hamzah
P07	Jaja

b. Aturan Membuat Subquery

Aturan-aturan untuk membuat subquery, yaitu

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.

c. 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.

Contoh :

Menampilkan semua nim yang nilai mid nya bukan yang terkecil

SELECT nim FROM Nilai WHERE MID > ANY (SELECT MID FROM Nilai);

Sedangkan jika subquery diawali kata kunci ALL, maka syarat hanya akan bernilai TRUE jika dipenuhi semua nilai yang dihasilkan subquery itu.

Contoh :

SELECT NIM FROM Nilai WHERE MID >= ALL (SELECT MID FROM Nilai);

d. Penggunaan EXIST dan NOT EXIST EXIST

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.

Contoh :

SELECT NIM, NAMA FROM Mahasiswa WHERE EXISTS (SELECT * FROM Nilai WHERE NIM=Nilai.NIM);

e. Contoh Subquery

1) Ambil nilai mid dan final dari mahasiswa yang bernama Astuti. SELECT MID, FINAL FROM NILAI WHERE NIM=(SELECT NIM FROM MAHASISWA WHERE NAMA='Astuti')
2) Ambil nilai kode matakuliah, mid dan final dari mahasiswa yang tinggal di jakarta. SELECT KD_MK, MID, FINAL FROM NILAI WHERE NIM IN(SELECT NIM FROM MAHASISWA WHERE ALAMAT = 'Jakarta')
3) Ambil nama-nama mahasiswa yang mengikuti ujian. SELECT NAMA FROM MAHASISWA WHERE EXISTS (SELECT NIM FROM NILAI WHERE NILAI.NIM= MAHASISWA.NIM)
4) Ambil nama-nama mahasiswa yang tidak mengikuti ujian. SELECT NAMA FROM MAHASISWA WHERE NOT EXISTS (SELECT NIM FROM NILAI WHERE NILAI.NIM= MAHASISWA.NIM)

BAB X

BASIS DATA TERDISTRIBUSI

10.1 Pengertian Basis Data Terdistribusi

Tujuan utama di balik peerkembangan sistem basis data adalah suatu keinginan untuk mengintegrasikan berbagai data-data operasional dari suatu organisasi dan menyediakan pengaksesan data yang terkontrol. Walaupun integrasi dan pengontrolan pengaksesan data akan mengimplikasikan suatu sentralisasi data dan sistem. Pada kenyataannya perkembangan jaringan computer mengarah kepada model kerja yang desentralisasi. Desentralisasi tersebut akan memperkenalkan struktur organisasi banyak perusahaan, yang secara logis terdistribusi ke dalam divisi-divisi, departemen-departemen, proyek-proyek, dan lain-lain, dan secara fisik terdistribusi ke dalam offices (kantor-kantor), plants (bangunan-bangunan untuk pekerjaan produksi suatu perusahaan), factories (pabrik-pabrik), dan lain-lain, dimana masing-masing unit akan merancang dan membiayai pengolahan data mereka masing-masing.

Perkembangan sistem basis data terdistribusi akan merefleksikan dan mencerminkan struktur organisasional tersebut, membuat data diseluruh unit dapat diakses dengan baik dan menyimpan data-data penting dan sering digunakan ketempat-tempat yang paling sering digunakan dan mudah ditemukan, serta meningkatkan kemampuan data untuk digunakan secara bersama-sama berikut efisiensi pengaksesannya.

Menurut (Ladjamudin, 2004) menyampaikan bahwa 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). Menurut (Ladjamudin, 2004), DBMS terdistribusi memiliki beberapa karakteristik antara lain:

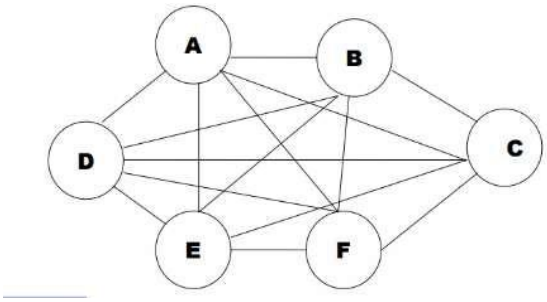
1. Kumpulan data-data logic (yang dapat digunakan secara bersama) terdistribusi pada beberapa unit computer yang berbeda.
2. Komputer tersebut terkoneksi ke dalam suatu jaringan komunikasi
3. Data pada masing-masing unit computer (work-station) terkontrol oleh suatu DBMS.
4. DBMS pada masing-masing bagian dapat menangani aplikasi-aplikasi local, secara otomatis.
5. Masiing-masing DBMS berpartisipasi paling tidak pada sati aplikasi global.

10.2 Topologi Distribusi Data

Menurut (Fathansyah, 2012), sebuah 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 simpul-simpul 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 untukmengamodasi sejumlah workstation dan sekaligus user lokal.

Bentuk-Bentuk Topologi Terdistribusi :

1. *Fully Connected Network*



Gambar 10.1
Fully Connected Network

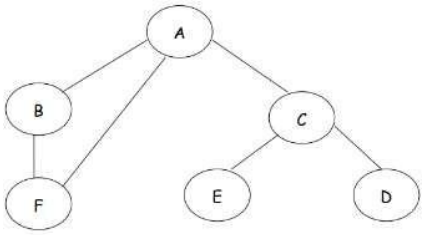
Keuntungan:

Kalau salah satu node rusak, yang lainnya masih dapat berjalan

Kerugian :

Control manajement tidak terjamin

2. *Partialy Conncted Network*



Gambar 10.2
Partially Connected Network

Keuntungan:

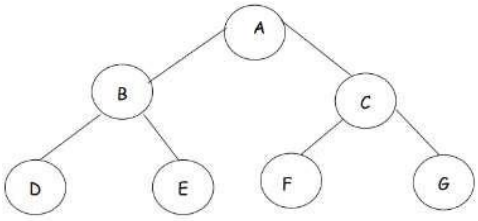
a. Reliability rendah

b. Biaya dapat ditekan

Kerugian :

Control manajement tidak terjamin

3. Tree Structured network



Gambar 10.3
Tree Structured Network

Keuntungan:

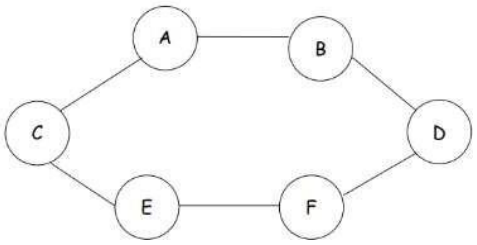
a. Bersifat sentral

b. Control manajement terjamin

Kerugian :

Kalau node pusat rusak maka semua akan rusak

4. Ring Network



Gambar 10.4
Ring Network

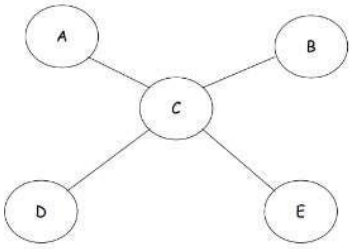
Keuntungan :

Apabila ada satu node yang rusak maka yang lain masih berjalan

Kerugian :

Control manajemen kurang terjamin kearena bersifat desentralisasi.

5. Star Network



Gambar 10.5
Star Network

Keuntungan:

Control manajement lebih terjamin, karena bersifat sentral dan reliability rendah

Kerugian:

Kalau ada node pusat rusak maka yang lainnya rusak juga.

10.3 Keuntungan dan Kerugian Basis Data Terdistribusi

1. Keuntungan Basis Data Terdistribusi

   a. Secara alami mengikuti struktur organisasi

   b. Adanya otonomi lokal

   c. Sifatnya dapat dipakai secara bersama

   d. Peningkatan ketersediaan

   e. Peningkatan kehandalan

   f. Peningkatan kinerja

   g. Ekonomis

   h. Pertumbuhan yang modular

2. Kerugian Basis Data Terdistribusi

   a. Harga software mahal (Biaya)

   b. Kompleksitas

   c. Kelemahan dalam keamanan

   d. Sulitnya menjaga keutuhan data

   e. Kurangnya standar

   f. Kurangnya pengalaman

   g. Perancangan basisdata lebih kompleks

10.4 Fragmentasi Data

1. Pengertian Fragmentasi

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.

Menurut (Hariyanto, 2004), alasan-alasan diperlukannya fragmentasi, yaitu :

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.

2. Aturan Fragmentasi

Beberapa Peraturan Yang Harus Didefinisikan Ketika Mendefinisikan Fragment, adalah :

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

Menurut (Hariyanto, 2004), kerugian fragmentasi yaitu :

a. Kinerja

Kinerja aplikasi yang memerlukan data dari fragmen-fragmen yang berlokasi terpisah dapat lebih lambat.

b. Integritas

Kendali integritas dapat lebih sulit jika data dan kebergantungan fungsional difragmentasi dan berlokasi di situs-situs yang berbeda.

3. 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.

Cara yang sederhana untuk membangun fragmentasi campuran sbb :

1) Menggunakan fragmentasi horizontal pada fragmentasi vertikal
2) Menggunakan fragmentasi vertical pada fragmentasi horizontal

Contoh Penggunaan Fragmentasi :

Kasus Jenis-Jenis Fragmentasi Ujian (NIM, Nama_Mhs, Kode_MK, Mt_Kuliah, Nil_Akhir, Grade)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
123	Fathi	101	Sistem Basis Data	78	B
124	Farah	102	Peranc. Sistem	60	C
125	Sarah	101	Sistem Basis Data	40	D
126	Salsabila	101	Sistem Basis Data	90	A
127	Azizah	103	Visual Basic	70	B
128	Farhan	103	Visual Basic	40	D
129	Faiz	102	Peranc. Sistem	80	A

Fragmentasi Horizontal terbagi menjadi 3 fragment yang berbeda berdasarkan Mt_Kuliah

a) Relasi Mt_Kuliah="Sistem Basis Data"

σ Mt_Kuliah="Sistem Basis Data" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
123	Fathi	101	Sistem Basis Data	78	B
125	Sarah	101	Sistem Basis Data	40	D
126	Salsabila	101	Sistem Basis Data	90	A

b) Relasi Mt_Kuliah="Peranc. Sistem"

σ Mt_Kuliah="Peranc. Sistem" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
124	Farah	102	Peranc. Sistem	60	C
129	Faiz	102	Peranc. Sistem	80	A

c) Relasi Mt_Kuliah="Visual Basic"

σ Mt_Kuliah="Visual Basic" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
127	Azizah	103	Visual Basic	70	B
128	Farhan	103	Visual Basic	40	D

Contoh Fragmentasi Vertical

Fragment di atas memenuhi kondisi jika Nama_Mhs dan Mt_Kuliahadalah hal-hal yang memenuhi syarat Fragmentasi vertical: berdasarkan dekomposisi-nya dengan menambahkan tupel_id

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
123	Fathi	101	Sistem Basis	78	B	1
124	Farah	102	Data	60	C	2
125	Sarah	101	Peranc. Sistem	40	D	3
126	Salsabila	101	Sistem Basis	90	A	4
127	Azizah	103	Data	70	B	5
128	Farhan	103	Sistem Basis	40	D	6
129	Faiz	102	Data	80	A	7
			Visual Basic			
			Visual Basic			
			Peranc. Sistem			

Relasi 1 = NIM, Nama_Mhs, Mt,Kuliah, Nil_Akhir, Grade, Tuple_ID

π NIM,Nama_Mhs,Mt,Kuliah,Nil_Akhir,Grade,Tuple_ID (UJian)

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
123	Fathi	Sistem Basis Data	78	B	1
124	Farah	Peranc. Sistem	60	C	2
125	Sarah	Sistem Basis Data	40	D	3
126	Salsabila	Sistem Basis Data	90	A	4
127	Azizah	Visual Basic	70	B	5
128	Farhan	Visual Basic	40	D	6
129	Faiz	Peranc. Sistem	80	A	7

Relasi 2 = NIM,Kode_MK,Nil_Akhir,Grade,Tuple_ID

π NIM,Kode_MK,Nil_Akhir,Grade,Tuple_ID (Ujian)

NIM	Kode_MK	Nil_Akhir	Grade	Tuple_ID
123	101	78	B	1
124	102	60	C	2
125	101	40	D	3
126	101	90	A	4
127	103	70	B	5
128	103	40	D	6
129	102	80	A	7

Contoh Fragmentasi Campuran

Terdapat relasi berdasarkan Mata Kuliah yang sama

Relasi 1a.

π NIM, Nama_Mhs, Mt_Kuliah, Nil_Akhir, Grade, Tuple_ID (σ Mt_Kuliah = “Sistem Basis Data” (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
123	Fathi	Sistem Basis Data	78	B	1
125	Sarah	Sistem Basis Data	40	D	3
126	Salsabila	Sistem Basis Data	90	A	4

Relasi 1b.

π NIM, Nama_Mhs, Mt_Kuliah, Nil_Akhir, Grade, Tuple_ID(σ Mt_Kuliah= “Peranc. Sistem” (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
124	Farah	Peranc. Sistem	60	C	2
129	Faiz	Peranc. Sistem	80	A	7

Relasi 1c

π NIM, Nama_Mhs, Mt_Kuliah, Nil_Akhir, Grade, Tuple_ID(σ Mt_Kuliah= “Visual Basic” (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
127	Azizah	Visual Basic	70	B	5
128	Farhan	Visual Basic	40	D	6

BAB XI

PERANCANGAN DAN IMPLEMENTASI MENGGUNAKAN DB DESIGNER

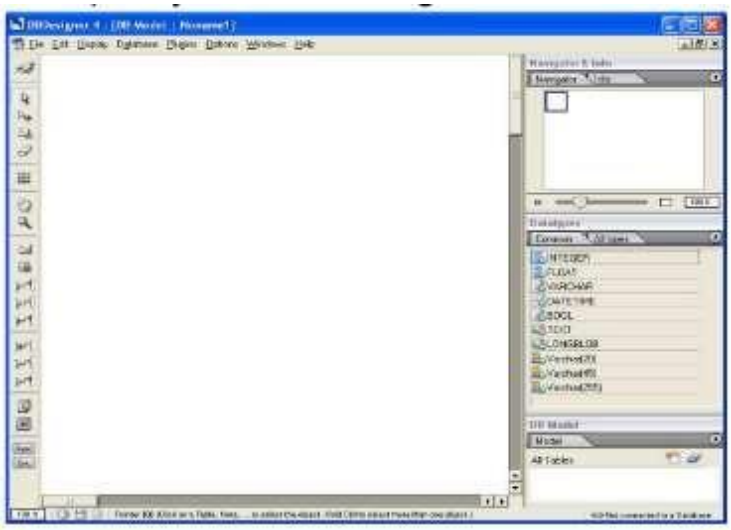
Perancangan dan Implementasi Basis Data Menggunakan SQL

Gambar 11.1
Perancangan dan Implementasi Basis Data Menggunakan SQL

Perangkat Lunak Bantu untuk Perancangan Basis Data Pada perangkat lunak bantu telah tersedia komponen-komponen (notasi-notasi) perancangan basis data. Salah satu perangkat lunak bantu untuk keperluan semacam itu adalah DBDesigner yang dioptimalkan untuk MySQL Database.

Gambar 11.2
DB Designer 4

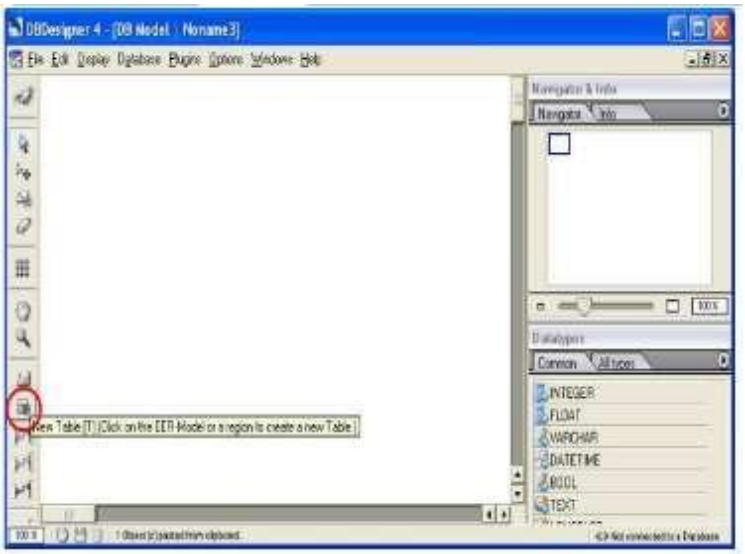
Tampilan Jendela DB Designer



Gambar 11.3
Tampilan awal DB Designer

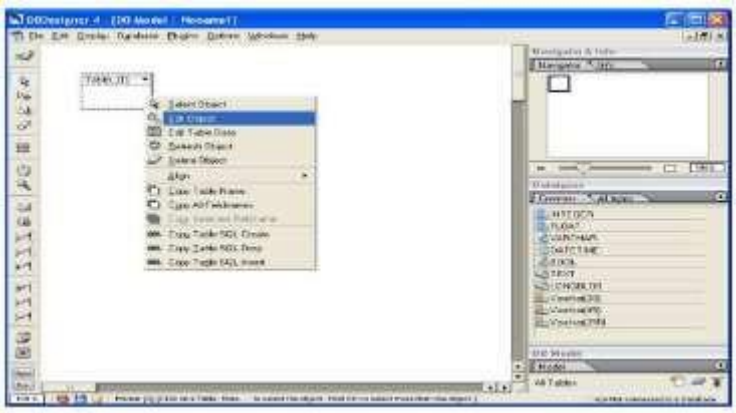
Contoh Penggunaan DBDesigner

1. Menggunakan Komponen TABEL dan RELASI Klik komponen Tabel pada toolbar seperti di gambar berikut :



Gambar 11.4
Komponen Tabel dan Relasi DB Designer

2. Letakan komponen tersebut pada page area sehingga muncul komponen Tabel (Table_01) pada page area, kemudian klik kanan komponen tsb sehingga muncul menu dan pilihlah Edit Object seperti berikut :

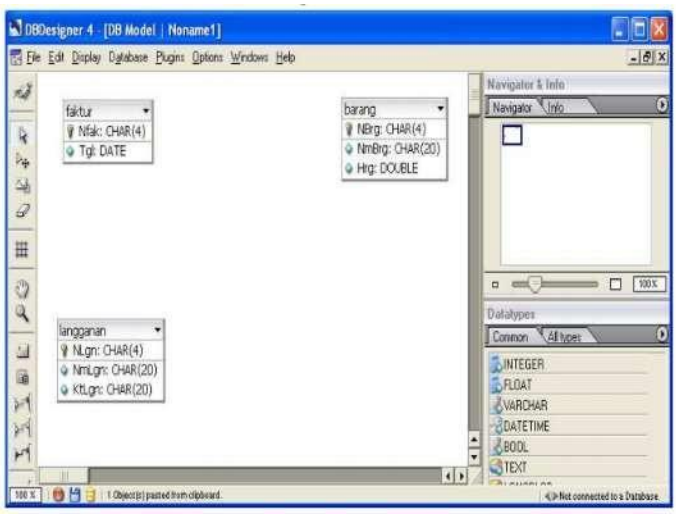


Gambar 11.5
Pembuatan tabel DB Designer

3. Menu Edit Object akan menampilkan jendela Table Editor. Pada Table Editor kita bisa menentukan properties dari tabel seperti nama tabel, tipe data, primary key dsb. Ubah dan simpanlah properties tabel (Table _01) menjadi tabel faktur (struktur tabel seperti pada pembahasan LRS tanpa ada FK) seperti berikut.

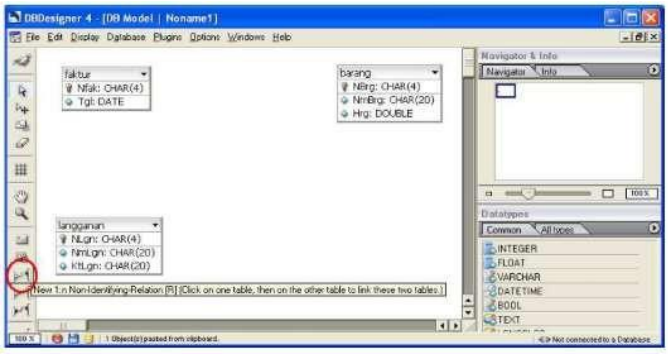
Gambar 11.6
Membuat Properties

4. Ulangi langkah-langkah menggunakan komponen Table di atas (tabel faktur) untuk tabel barang dan langganan (struktur tabel seperti pada pembahasan LRS tanpa ada FK). Sehingga ada 3 komponen Table seperti gambar berikut



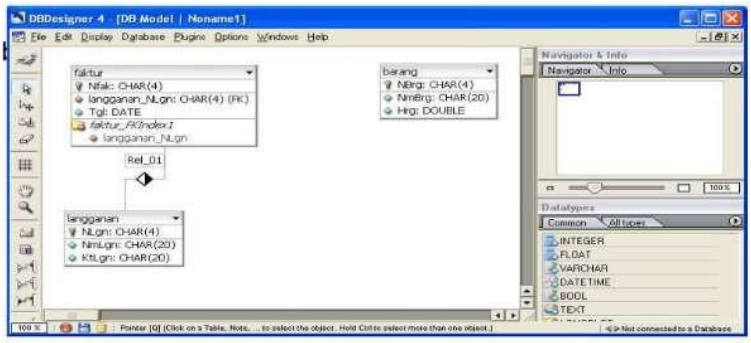
Gambar 11.7
Tabel Barang, Tabel Langganan, Tabel Faktur

5. Langkah berikutnya membuat relasi 1-M antara langganan dengan faktur dengan cara klik komponen 1-n Relation pada toolbar seperti di gambar berikut.



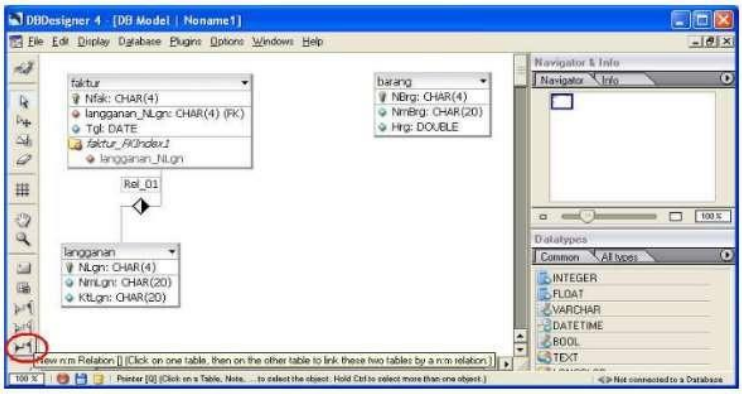
Gambar 11.8
Membuat Relasi One to one antar Tabel

6. Klik di tabel langganan kemudian klik di tabel faktur, sehingga muncul komponen relasi yang menghubungkan kedua tabel tsb.



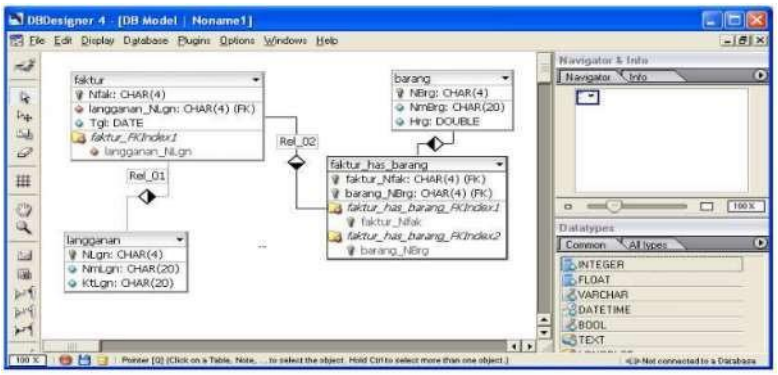
Gambar 11.9
Tampilan Relasi One to Many antar Tabel

7. Langkah berikutnya membuat relasi M-M antara faktur dengan barang dengan cara klik komponen n-m Relation pada toolbar seperti di gambar berikut



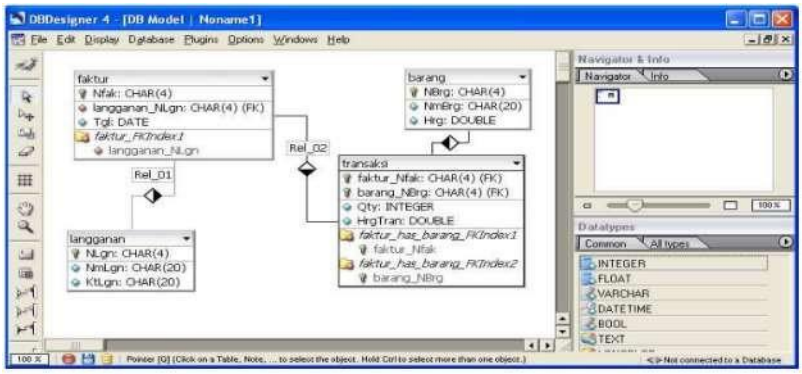
Gambar 11.10
Membuat Relasi Many to Many antar Tabel

8. Klik di tabel faktur kemudian klik di tabel barang, sehingga muncul komponen relasi yang disertai munculnya tabel baru (faktur_has_barang) dan FK (Nfak & NBrg) berada pada tabel tsb, seperti gambar berikut.



Gambar 11.11
Tampilan Relasi Many to Many antar Tabel

9. Edit properties tabel faktur_has_barang yaitu dengan mengganti nama menjadi tabel transaksi dan menambahkan field Qty dan HrgTran. Sehingga menjadi seperti gambar berikut.

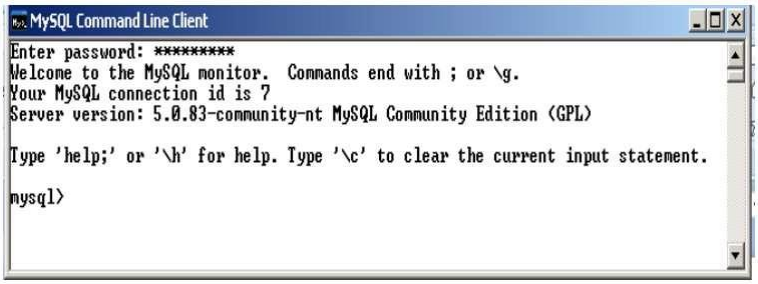


Gambar 11.12
Menambahkan Atribut Pada Tabel

Untuk mengekspor hasil rancangan database ke dalam database digunakan Database Synchronization. Database yang digunakan pada contoh ini adalah MySQL. Sebelum melakukan sinkronisasi, kita perlu membuat koneksi ke database MySQL terlebih dahulu. Jika remote connection dengan root diperbolehkan maka gunakan user root. Jika tidak maka kita butuh membuat user baru terlebih dahulu. Berikut ini adalah cara bagaimana membuat user baru yaitu db_owner.

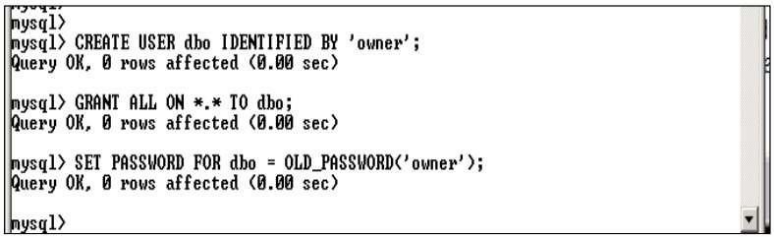
Melakukan Sinkronisasi :

1. Lakukan login terlebih dahulu ke MySQL dengan memasukkan password root.



Gambar 11.13
Tampilan Login

2. Buat user baru bernama dbo dengan password ”owner”. Ketikkan 3 perintah dibawah ini.

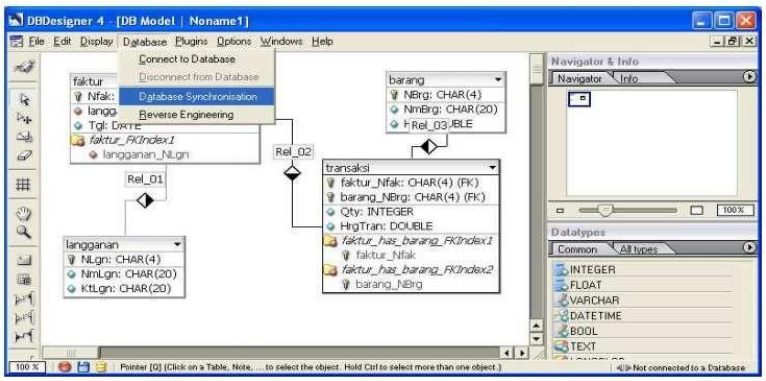


Gambar 11.14
Membuat User Baru

3. Buat Database baru yaitu dbpenjualan

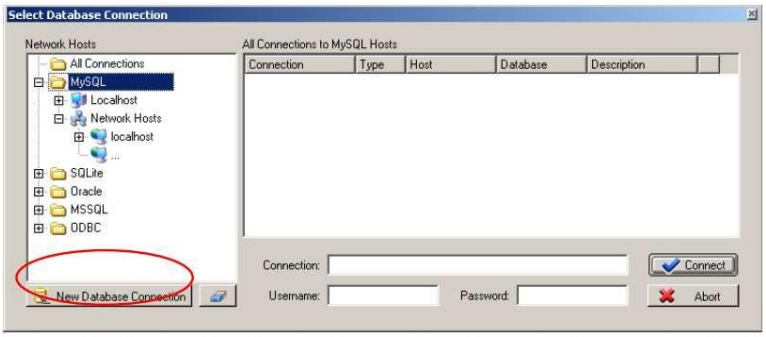
Gambar 11.15
Membuat Database

4. Mengekspor Tabel Hasil Rancangan Ke Server Database Mengekspor tabel ke server database bisa dilakukan dari menu Database → DatabaseSychronisastion seperti gambar berikut :

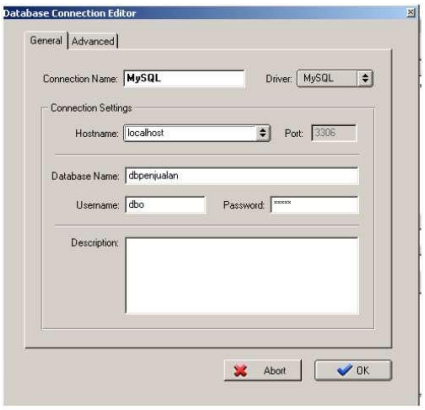


Gambar 11.16
Melakukan Sinkronisasi

5. Lalu pilih MySQL sebagai database dan kemudian klik New Database Connection



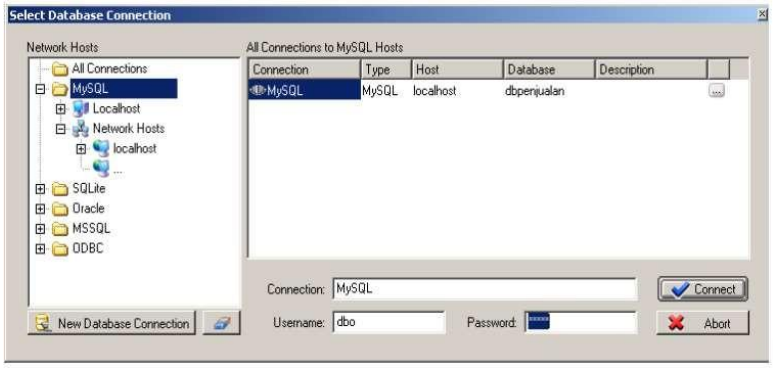
Gambar 11.17
Pilih New Database Connection



Gambar 11.18
Tampilan New Database Connection

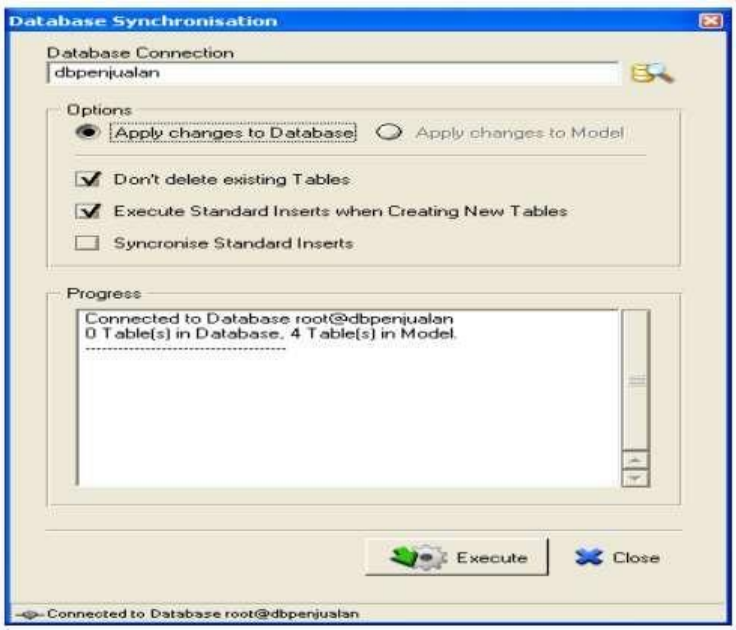
6. Masukkan Nilai berikut: Connection Name : MySQL Hostname : localhost Database Name : dbpenjualan UserName : dbo Password : owner Lalu klik OK

7. Klik Connect untuk terkoneksi ke MySQL



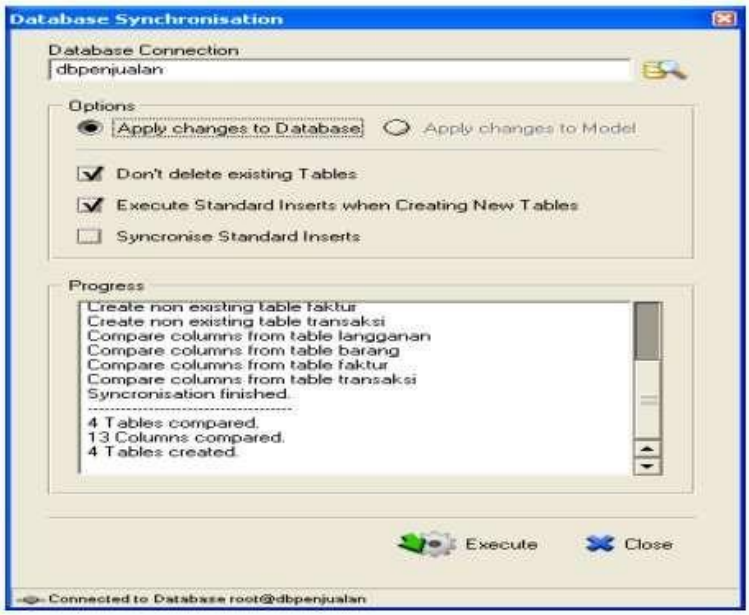
Gambar 11.19
Tampilan Koneksi ke MySQL

8. Klik Execute untuk mengeksekusi sinkronisasi



Gambar 11.20
Mengeksekusi Sinkronisasi

9. Setelah tampil jendela seperti di atas, selanjutnya klik tombol EXECUTE untuk mengekspor tabel ke server database MySQL dan akan tampil progress report seperti berikut



Gambar 11.21
Tampilan Behasil Ekspor tabel Ke MySQL

BAB XII

LINGKUNGAN BASIS DATA

12.1 Konkurensi *(CONCURRENCY)*

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.

Transaksi A	Waktu	Transaksi B
=	↓	=
Baca R	t1	=
=	↓	=
=	t2	Baca R
=	↓	=
Modifikasi R	t3	=
=	↓	=
=	T4	Modifikasi R
=	↓	=

Data transaksi pada rekening bersama (Ika dan Susi)

Waktu	Transaksi Ika	Transaksi Susi	Saldo
T1	Read Saldo	.....	1.000.000
T2	.....	Read Saldo	1.000.000
T3	Saldo:=Saldo-50.000	.....	1.000.000
T4	Write Saldo	.....	950.000
T5	.....	Saldo:= saldo+100.000	1.000.000
T6	.....	Write Saldo	1.100.000

Nilai saldo menjadi tidak benar disebabkan transaksi Susi membaca nilai saldo sebelum transaksi Ika mengubah nilai tersebut dalam database, sehingga nilai yang sudah di update yang dihasilkancdari transaksi Ika menjadi hilang.

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*).

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca R	t1	Modifkasi R
-	↓	-
-	t2	-
-	↓	-
Modifikasi R	t3	Rollback
-	↓	-

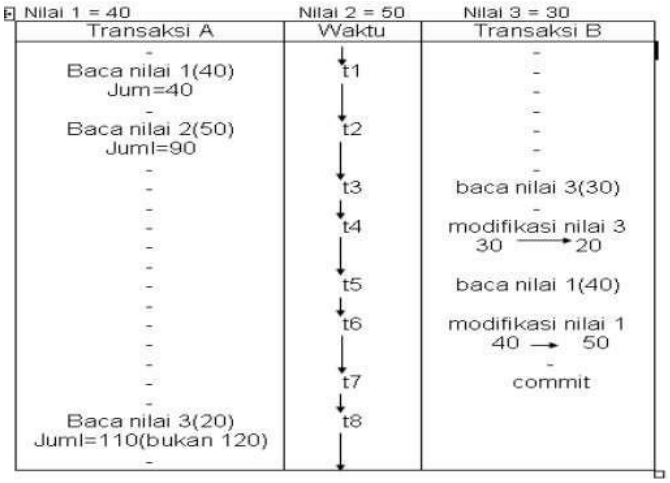
Contoh *uncommited Update Problem*

Waktu	Transaksi Simpanan	Transaksi Bunga	Saldo
T1	Read Saldo	……	1.000.000
T2	Saldo:=saldo+1.000.0000	……	1.000.000
T3	Write Saldo	……..	2.000.000
T4	…….	Read Saldo	2.000.000
T5	…….	Saldo:= saldo*0.15	2.000.000
T6	……	Write Saldo	2.300.000
T7	……	RollBack	2.300.000

Nilai saldo menjadi tidak benar disebabkan terjadi RollBack pada T7 yang membatalkan transaksi sebelumnya (T6), sehingga saldo seharusnya tetap 2.000.000

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.



Transaksi A menjumlahkan nilai 1, nilai 2 dan nilai 3

Transaksi B → nilai 1 + 10, nilai 3 –10

12.2 Locking

LOCKING adalah salah satu mekanisme pengontrol *concurrency.*

KONSEP DASAR :

Ketika sebuah transaksi memerlukan jaminan kalau record yang diinginkan tidak akan berubah secara mendadak, maka diperlukan kunci untuk record tersebut FUNGSI 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.

	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

X = kunci X
S = kunci S
N = No
Y = Yes

Masalah kehilangan modifikasi (Lost Update Problem)

Transaksi A	Waktu	Transaksi B
-		-
baca R11 (kunci S)		-
-	t2	baca R(kunci S)
-	t3	-
modifikasi R (kunci X)		-
tunggu	t4	modifikasi R (kunci X)
:		tunggu
:		:
:		tunggu
tunggu		

Masalah Modifikasi Sementara (uncommited Update Problem)

Transaksi A	Waktu	Transaksi B
-		-
-	t1	modifikasi R (kunci X)
-	t2	-
baca R kunci (S)		-
tunggu	t3	-
:		synchpoint (kunci X dilepas)
:	t4	-
tunggu		-
baca R kembali (Kunci S)		-

Transaksi A	Waktu	Transaksi B
-	t1	modifikasi R
-		(kunci X)
Modifikasi R	t2	-
Kunci (X)		-
tunggu		-
:	t3	synchpoint
:		(kunci X dilepas)
tunggu		-
modifikasi R	t4	-
(Kunci X)		-

Masalah Analisa yang tidak konsisten (Problem of inconsistensi Analisa)

Nilai 1 = 40 Nilai 2 = 50 Nilai 3 = 30

Transaksi A	Waktu	Transaksi B
-		-
Baca nilai 1(40)	t1	-
(kunci S)		-
Jumlah=40		-
-		-
Baca nilai 2(50)	t2	-
(kunci S)		-
Jumlah=90		
-	t3	baca nilai 3(30)
-		(kunci S)
-		-
-	t4	modifikasi nilai 3
-		(kunci X)
-		30 → 20
-		-
-	t5	baca nilai 1(40)
-		(kunci S)
-		-
-	t6	modifikasi nilai 1
-		(kunci X)
-		tunggu
modifikasi nilai 3	t7	:
(kunci S)		:
tunggu		tunggu

12.3 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

12.4 Crass dan Recovery

PENGERTIAN :

Crash adalah suatu *failure* atau kegagalan dari suatu sistem

PENYEBAB DARI KEGAGALAN ADALAH :

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

1. *Volatile storage*, biasanya informasi yang terdapat pada volatile akan hilang, jika terjadi kerusakan sistem (system crash) contoh: RAM
2. *Non Volatile Storage*, biasanya informasi yang terdapat pada non volatile strorage tidak akan hilang jika terjadi kerusakan sistem contoh: ROM
3. *Stable Storage*, informasi yang terdapat dalam stable storage tidak pernah hilang. contoh: Harddisk RAID

Jenis-jenis kegagalan

1. *Logical Error*, program tidak dapat lagi dilaksanakan disebabkan oleh kesalahan input, data tidak ditemukan, over flow
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

12.5 Security

SECURITY adalah suatu proteksi data terhadap perusakan data dan pemakaian oleh pemakai yang tidak mempunyai ijin.

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*)

DUA KATAGORI 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

BEBERAPA 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

12.6 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

12.7 Integrity

Integrity Berarti memeriksa keakuratan dan validasi data

BEBERAPA JENIS *INTEGRITY :*

1. Integrity Konstains, memberikan suatu sarana yang memungkinkan pengubahan database oleh pemakai berwenang sehingga tidak akan menyebabkan data inkonsistensi
2. Integrity Rule (pada basisdata 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.

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

Hidayatullah, P., & Kawistara, J. K. (2015). *Pemograman Web*. Informatika.

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

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

Ladjamudin, A. B. Bin. (2005). *Analisis dan Desain Sistem Informasi*. Graha Ilmu.

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

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