DIKTAT

BASIS DATA

Oleh:

Abdul Halim Hasugian, M.Kom
NIDN : 0127038801

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA
MEDAN
2017

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillah segala puji hanya milik Allah Tuhan sekalian alam. Atas berkat rahmat dan karuniaNya, saya dapat menyelesaikan penulisan diktat ini dengan judul "**Basis Data** ". Shalawat dan salam senantiasa tercurah kepada Muhammad SAW beserta kerabat, sahabat, para pengikutnya sampai akhir zaman, adalah sosok yang telah membawa manusia dan seisi alam dari kegelapan ke cahaya sehingga kita menjadi manusia beriman, berilmu, dan tetap beramal shaleh agar menjadi manusia yang berakhlak mulia.

Penulisan diktat ini bertujuan untuk melengkapi persyaratan pengusulan kenaikan pangkat di Fakultas Sains dan Teknologi Program Studi Ilmu Komputer Sumatera Utara Medan. Diktat ini juga diharapkan dapat menambah wawasan ilmu pengetahuan, khususnya ilmu komputer dalam instalasi nilai-nilai Islam yang terpadu dalam proses pembelajaran di lingkungan UIN Sumatera Utara Medan.

Dalam penulisan diktat ini, saya sangat menyadari bahwa masih banyak kekurangan yang perlu perbaikan di sana sini, sumbangan pemikiran yang membangun sangat penulis harapkan dari rekan-rekan sejawat terutama dari dosen-dosen senior yang terhimpun dalam mata kuliah serumpun. Juga usulan dari para pengguna bahan ajar ini terutama mahasiswa ilmu komputer, semoga konten pembelajaran matematika dapat diperkaya melalui evaluasi terus menerus. Atas segala budi baik yang telah penulis terima dari semua pihak untuk itu saya ucapkan ribuan terima kasih. Semoga Allah SWT membalas kebaikan seluruh rekan sekalian dengan ganjaran yang berlipat ganda, Amiin.

Medan, November 2017
Penulis

Abdul Halim Hasugian, M.Kom
NIDN:0127038801

DAFTAR ISI

BAB I

PENGANTAR BASIS DATA

1. Pengertian Basis Data

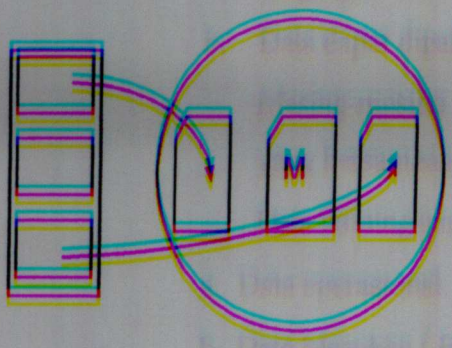
Dalam setiap langkah kehidupan modern manusia tidak bisa dilepaskan dari basis data. Sejak seorang bayi lahir sudah dicatat dan diisikan datanya pada rumah sakit tempat bayi tersebut dilahirkan, kemudian saat membuat akte kelahiran kembali dicatat di pemerintahan. Saat akan memasuki usia sekolah SD, SMP, SMA hingga perguruan tinggi bahkan saat bekerja maupun kematiannya akan selalu dicatat datanya dalam organisasi yang kompeten.

Sebenarnya basis data tidak selalu berhubungan dengan komputer karena pencatatan secara manual ke dalam suatu buku besar pun bisa dianggap sebagai suatu basis data, akan tetapi ini bila berbicara mengenai teknologi basis data, maka akan selalu dihubungkan dengan teknologi komputer karena dua hal tersebut berjalan beriringan.

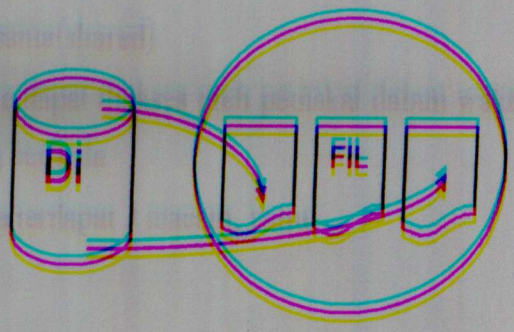
Banyak definisi yang digunakan untuk menjelaskan istilah basis data. Disini disimpulkan bahwa basis data adalah Basis data terbagi menjadi 2 kata yaitu basis dan data. **Basis** dapat diartikan markas, gudang, tempat berkumpulnya. Sedangkan **Data** merupakan representasi fakta dunia nyata yang mewakili sesuatu objek seperti manusia (pegawai, siswa, pembeli, dll), barang, hewan, peristiwa, dll.

Basis data sendiri dapat diartikan dalam sejumlah sudut pandang, yaitu sebagai berikut :

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan file /tabel/arsip saling berhubungan yang disimpan dalam media penyimpanan elektronik.



Gambar 1. lemari Arsip



Gambar 2. 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 arsip menggunakan lemari sebagai media penyimpanannya, maka basisdata mnenggunakan media penyimpanan elektronis seperti disk (disket, harddisk).

Yang perlu diingat adalah bahwa tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data. Yang sangat ditonjolkan dalam basisdata adalah pengaturan/pemilaha/pengelompokkan/pengorganisasian data yang akan kita simpan sesuai fungsi/jenisnya. Pemilahan/ pengelompokan ini dapat berbentuk sejumlah file/ tabel terpisah atau dalam bentuk pendefinisian kolom-kolom/field-field data dalam setiap file/tabel.

Data yang tersimpan dalam basis data bisa beragam ukurannya, misalnya data karyawan dalam suatu perusahaan mungkin akan hanya akan berjumlah puluhan, ratusan atau ribuan data. Karena informasi adalah data yang sudah diolah, maka dapat dikatakan bahwa basis data merupakan sumber utama dari suatu sistem informasi. Basis data yang buruk akan menyebabkan sistem informasi yang buruk pula.

2. Ciri-ciri data dalam basis data

Adapun ciri-ciri data dalam basis data ada 2 macam, yaitu :

a. Data disimpan secara terintegrasi(integrated)

Database merupakan kumpulan dari berbagai macam file dari aplikasi- aplikasi

yang berbeda, yang disusun dengan cara menghilangkan bagian- bagian yang rangkap.

b. Data dapat dipakai secara bersama-sama(shared)

Masing-masing bagian dari database dapat diakses oleh pemakai dalam waktu yang bersamaan untuk aplikasi yang berbeda.

Data hubungannya dengan basis data terdapat 3 macam, yaitu :

a. Data operasional
b. Data Masukan (Input Data)
c. Data Keluaran (Output Data)

3. Tujuan Basis Data

a. Kecepatan & kemudahan (*speed*)

Pemanfaatan basis data memungkinkan kita menyimpan, memanipulasi, menampilkan data dengan lebih cepat dan mudah.

b. Efisiensi ruang penyimpanan (*space*)

Penekanan redudansi data dengan menerapkan sejumlah teknik pengkodean atau membuat relasi = relasi antar file, akan mengurangi pemakaian space di media penyimpanan.

c. Keakuratan (*accuracy*)

Pemanfaatan kode dan relasi antar file dengan aturan ketat berguna untuk menekan ketidakakuratan pemasukan data.

d. Ketersediaan (*availability*)

Ketersediaan data dapat didukung dengan teknologi jaringan komputer, agar data dapat diakses

e. Kelengkapan (*completeness*)

Kelengkapan data yang disimpan dalam sebuah database bersifat relative (terhadap kebutuhan dan waktu), bisa jadi saat ini dianggap sudah lengkap, tetapi belum tentu pada suatu saat dianggap lengkap. Untuk mengantisipasi perubahan kebutuhan data, maka basis data sebaiknya dirancang untuk dapat menambah data dan dapat juga merubah struktur basis data.

f. Keamanan (*security*)

Penerapan kemanan basis data berkaitan dengan pemberian hak akses kepada user dengan tingkatan tertentu beserta operasi–operasi yang boleh dilakukan.

g. Kebersamaan pemakaian (*sharability*)

Kebersamaan pemakaian data umumnya ada dalam basis data yang menggunakan aplikasi multiuser.

4. Mempelajari Basis Data

Alasan mengapa kita harus mempelajari basis data adalah sebagai berikut :

a. perpindahan dari komputasi ke informasi

b. himpunan elemen data semakin banyak dan beragam

1) perpustakaan digital. Video interaktif

2) kebutuhan untuk memperluas DBMS

c. DBMS mencakup bidang ilmu lain

System operasi, bahasa pemrograman, teori komputasi, AI, logika, multimedia.

5. Operasi Dasar Basis Data

a. Pembuatan Basis Data (*Create Database*)

Yang identik dengan pembuatan lemari arsip yang baru.

b. Penghapusan Basis Data (*Drop Database*)

Yang identik dengan perusakan lemari arsip (sekaligus beserta isinya, jika ada)

c. Pembuatan File/Table baru ke suatu basis data (*Create Table*)

Yang identik dengan penambahan map arsip baru ke sebuah lemari arsip yang telah ada.

d. Penghapusan File/Table dari suatu basis data (*Drop Table*)

Yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.

e. Penambahan data baru ke suatu file/table di sebuah basis data (*insert*)

Identik dengan penambahan lembaran arsip baru kesebuah map arsip.

f. Pengambilan data dari sebuah file/table (*Retrieve/Search/select*)

Identik dengan pencarian lembaran arsip dari sebuah map arsip.

g. Pengubahan data dari sebuah file/table (*Update*)

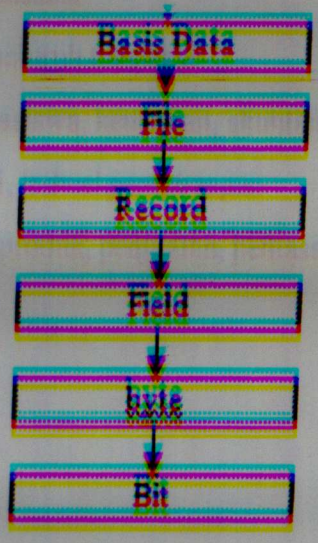
Identik dengan perbaikan isi lembaran arsip yang ada di sebuah map arsip.

h. Penghapusan data dari sebuah file/table (*Delete*)

Identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

6. Hirarki Data

Hirarki adalah urutan atau aturan dari tingkatan abstraksi menjadi seperti struktur pohon. Berdasarkan tingkat kompleksitas nilai data, tingkatan data dapat disusun kedalam sebuah hirarki, mulai dari yang paling sederhana hingga yang paling komplek.



Gambar 3. Hirarki Data

Ada 7 tingkatan hirarki basis data yaitu sebagai berikut :

a. Sistem Basis Data
b. Basis data
c. File / tabel
d. Record/ baris
e. Field / kolom
f. Byte
g. Bit

7. Aplikasi Basis Data

Aplikasi basis data adalah pintu masuk ke dalam sumber daya basis data. Aplikasi basis data terdiri atas sekumpulan menu, formulir, laporan dan program yang memenuhi kebutuhan suatu fungsional unit bisnis / organisasi atau instansi.

Kebutuhan akan aktivitas menentukan kebutuhan akan suatu aplikasi dan

kebutuhan akan aplikasi menentukan akan kebutuhan suatu basis data. Tujuan aplikasi adalah untuk menyediakan informasi dan membantu pemakai membuat keputusan. Basis data dapat digunakan pada beberapa aplikasi antara lain :

a. Industri manufaktur : produksi, persediaan, pemesanan
b. Manajemen rumah sakit : registerasi, rekam medis, perawatan
c. Manajemen perpustakaan : seluruh transaksi
d. Perhotelan : seluruh transaksi
e. Perbankan : melayani seluruh transaksi
f. Perguruan tinggi : mahasiswa, keuangan, akuntansi, lulusan
g. Penerbangan : reservasi, jadwal penerbangan
h. Penjualan : pelanggan, produk, penjualan, pemasaran

BAB II

SISTEM BASIS DATA

1. Pengertian Sistem Basis Data

Sistem adalah tatanan yang terdiri dari komponen-komponen fungsional yang saling berhubungan untuk mencapai tujuan tertentu.

Sistem Basis Data adalah kumpulan file / tabel yang saling berhubungan dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai dan / atau program lain untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut.

2. Komponen Sistem Basis Data

Komponen utama sistem basis data adalah sebagai berikut :

a. Perangkat Keras (Hardware)

Perangkat keras yang biasanya terdapat dalam sebuah basis data :

1) Komputer
2) Harddisk
3) Tape atau removable disk untuk backup data
4) Media / perangkat komunikasi untuk sistem jaringan

Perangkat keras pendukung operasi pengolahan data yang utama adalah sistem komputer yang mempunyai komponen sebagai berikut :

1) Perangkat keras masukan
2) Perangkat keras keluaran
3) Perangkat keras unit pengolahan

b. Perangkat Lunak (Software)

Secara sederhana, SO adalah program yang mengaktifkan dan mengendalikan seluruh sumber daya komputer.

Secara lebih luas, perangkat lunak dapat dikategorikan dalam tiga bagian, yaitu :

1) Perangkat lunak sistem operasi
2) Perangkat lunak bahasa
3) Perangkat lunak aplikasi

c. Basis Data

Sebuah sistem basis data dapat terdiri dari beberapa basis data. Setiap basis data dapat berisi sejumlah file dan strukturnya.

d. Database Management System / DBMS

Pengelolaan basis data secara fisik ditangani oleh sistem khusus, yang disebut DBMS. Perangkat lunak yang termasuk DBMS seperti, dBase III+, dBase IV, Foxbase, Rbase, MS-Access dan Borland Paradox (kelas sederhana) atau Borland-Interbase, MS-SQLServer, CA-Open Ingres, Oracle, Informix dan Sybase (untuk kelas komplex, berat)

e. PEMAKAI

Ada beberapa tipe pemakai sistem basis data yang dibedakan berdasarkan cara berinteraksi dengan sistem :

1) Programmer Aplikasi

Pemakai yang berinteraksi dengan basis data melalui DML, yang disertakan dalam program yang ditulis dalam bahasa pemrograman induk (seperti Visual Basic, Borland Delphi, C, Pascal, Cobol dll.)

2) User Mahir (Casual User)

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

3) User Umum (End User / Naive User)

Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (executable program) yang telah ditulis / sediakan sebelumnya (contoh : orang akses database melalui web, teller di bank)

4) User Khusus (Specialized User)

Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus, seperti untuk aplikasi AI, Sistem Pakar, dll.

f. Aplikasi (Perangkat Lunak) Lain

Aplikasi ini sifatnya opsional, tergantung kebutuhan.

3. Tujuan Sistem Basis Data

Ada beberapa tujuan sistem basis data antara lain yaitu :

a. Mencegah data redundancy dan inconsistency

Kerangkapan data dalam basis data terjadi akibat penyusunan basis data untuk aplikasi-aplikasi tidak memperhatikan kriteria sebuah basis data. Kerangkapan data juga dapat terjadi akibat penyusunan basis data dilakukan oleh perancang yang berbeda dalam selang waktu yang cukup lama. Penyebab utama munculnya data tidak konsisten adalah akibat munculnya kerangkapan data dalam file.

b. Mempermudah dalam melakukan akses terhadap data

c. Mempertimbangkan data isolation

d. Mencegah concurent access anomaly atau penyimpangan akses secara bersama-sama

e. Mempertimbangkan masalah keamanan data

f. Mempertimbangkan masalah integritas data

Integritas data berhubungan dengan kinerja sistem agar dapat melakukan kendali/kontrol pada semua bagian sistem. Integritas dimaksudkan sebagai suatu sarana untuk menyakinkan bahwa data-data yang tersimpan dalam basis data selalu berada dalam kondisi yang benar.

4. Keuntungan Sistem Basis Data

a. Mengurangi redundansi

Data yang sama pada beberapa aplikasi cukup disimpan sekali saja.

b. Menghindarkan inkonsistensi

Karena redundansi berkurang, sehingga umumnya update hanya sekali saja.

c. Terpeliharanya integritas data.

d. Data tersimpan secara akurat.

e. Data dapat dipakai bersama-sama

f. Data yang sama dapat diakses oleh beberapa user pada saat bersamaan.

g. Memudahkan penerapan standarisasi

Menyangkut keseragaman penyajian data.

h. Jaminan security

Data hanya dapat diakses oleh yang berhak.

i. Menyeimbangkan kebutuhan

Dapat ditentukan prioritas suatu operasi, misalnya antara update (mengubah data) dengan retrieval (menampilkan data) didahulukan update.

5. Kerugian Sistem Basis Data

a. MAHAL

1) Diperlukan hardware tambahan

a) CPU yang lebih besar

b) Terminal yang lebih banyak

c) Alat untuk komunikasi

2) Biaya performance yang lebih besar

a) Listrik

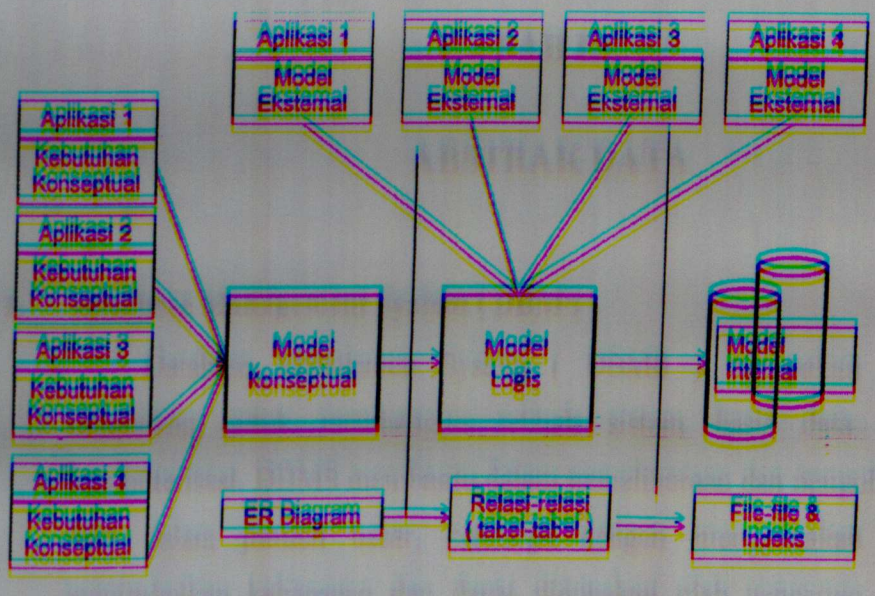
b) Personil yang lebih tinggi klasifikasinya

c) Biaya telekomunikasi yang antar lokasi/kota

b. KOMPLEKS

c. PROSEDUR BACKUP & RECOVERY SULIT

6. Struktur Basis Data



Gambar 4. Struktur Basis Data

BAB III

ABSTRAK DATA

1. Database Management System (DBMS)

Database Manajemen System (DBMS) merupakan software yang digunakan untuk membangun sebuah sistem basis data yang berbasis komputerisasi. DBMS membantu dalam pemeliharaan dan pengolahan kumpulan data dalam jumlah besar, sehingga dengan menggunakan DBMS tidak menimbulkan kekacauan dan dapat digunakan oleh pengguna sesuai dengan kebutuhan.

Selain itu, DBMS dapat diartikan sebagai perantara bagi pemakai dengan basis data. Untuk berinteraksi dengan DBMS (basis data) menggunakan bahasa basis data yang telah ditentukan oleh perusahaan DBMS.

Database Management System (DBMS) terdiri dari :

a. Sekumpulan data yang saling berhubungan
b. Suatu himpunan program yang melakukan akses terhadap data tersebut

Tujuan DBMS yang paling utama adalah :

a. Memelihara informasi
b. Informasi tersedia pada saat yang dibutuhkan

Data yang disimpan perlu diatur dalam Manajemen Data, oleh karena itu perlu dipelajari :

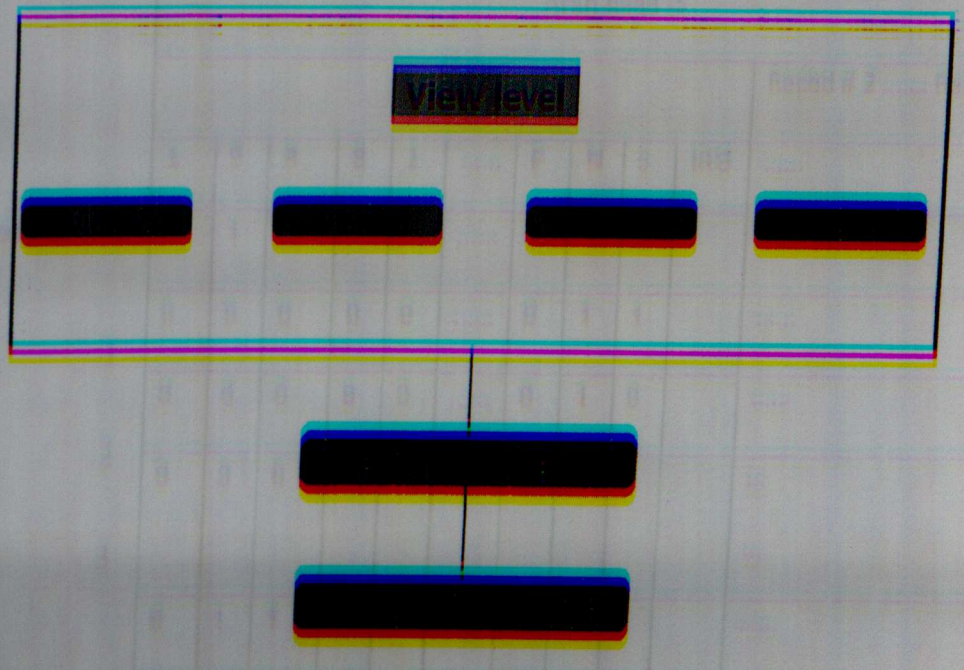
a. Struktur informasi dan
b. Mekanisme dalam melakukan manipulasi terhadap informasi

2. Abstraksi Data

Salah satu tujuan dari DBMS adalah untuk menyediakan fasilitas/antarmuka (*interface*) kepada user. untuk itu sistem tersebut akan menyembunyikan detail tentang bagaimana data disimpan dan dipelihara, sehingga data yang terlihat oleh user sebenarnya berbeda dengan yang tersimpan secara fisik.

Abstraksi data merupakan tingkatan-tingkatan pengguna dalam memandang bagaimana sebenarnya data diolah dalam sebuah sistem database sehingga

menyerupai kondisi yang sebenarnya dihadapi oleh pengguna sehari-hari. Sebuah DBMS seringkali menyembunyikan detail tentang bagaimana sebuah data disimpan dan dipelihara (diolah) dalam sebuah sistem database, dengan tujuan untuk memudahkan pengguna dalam menggunakan DBMS tersebut. Karena itu seringkali data yang terlihat oleh pemakai sebelumnya berbeda dengan yang tersimpan secara fisik.



Gambar 5. Abstraksi Data

Abstraksi data terbagi atas 3 level yaitu sebagai berikut :

a. Level Fisik (*Physical Level*)

Merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana sesungguhnya data tersebut disimpan. Untuk level ini pemakai melihat data sebagai gabungan dari struktur dan datanya sendiri. Pada level ini kita berurusan dengan data sebagai teks, sebagai angka bahkan melihat data sebagai himpunan bit data.

Contoh :

NIM : 19981150001

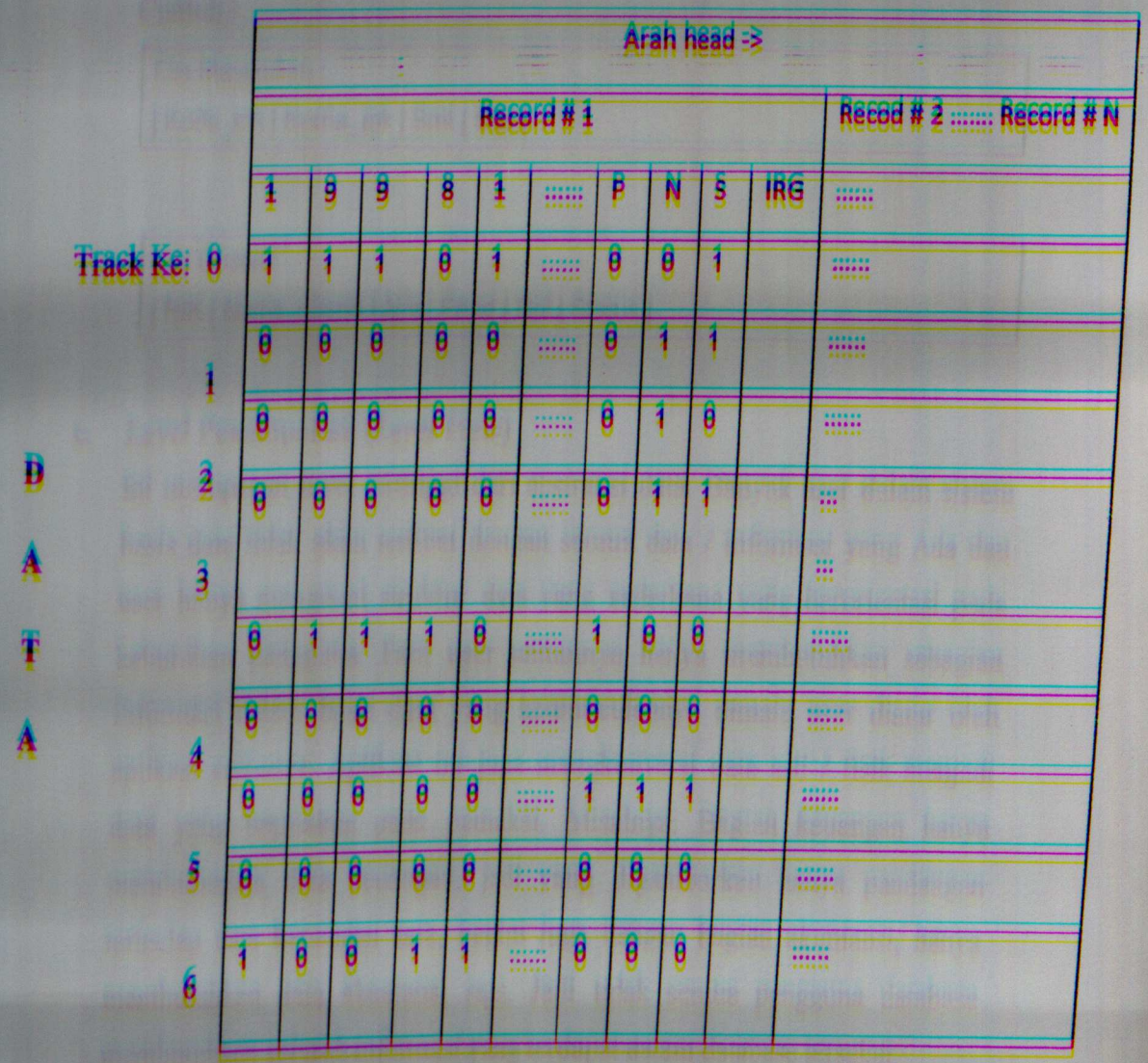
Nama : Budi

Alamat : Pangkalpinang

Tempat lahir : Bali

Tanggal lahir : 15 April 1975

Pekerjaan : PNS



b. **Level Lojik / Konseptual (*Conceptual level*)**

Pada level ini abstraksi data menggambarkan data apa yang sebenarnya (secara fungsional) disimpan dalam basis data dan hubungannya dengan data yang lain.

Contohnya, Seorang pengguna dalam level ini dapat mengetahui bahwa data mahasiswa disimpan pada tabel mahasiswa, tabel krs, tabel transkrip dan lain sebagainya.

Contoh :

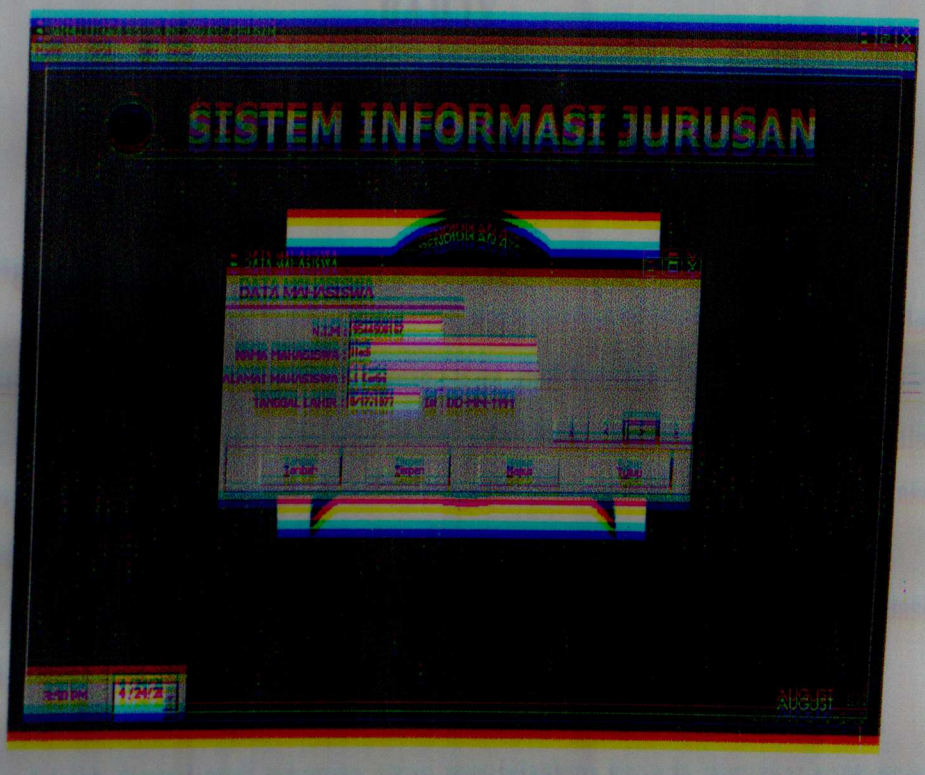
File Matakuliah :
| Kode_mk | Nama_mk | Smt | Sks |

File Dosen :
| NIK | Nama_dosen | Alm | Pend | Gol | Status |

c. **Level Penampakan (*Level View*)**

Ini merupakan level tertinggi dari abstraksi data. Banyak user dalam sistem basis data tidak akan terlibat dengan semua data / informasi yang Ada dan user hanya mengenal struktur data yang sederhana yang berorientasi pada kebutuhan pengguna. Para user umumnya hanya membutuhkan sebagian informasi dalam basis data yang kemunculannya dimata user diatur oleh aplikasi *end-user*. Aplikasi ini juga mengkonversi data asli / fisik menjadi data yang bermakna pada pemakai. Misalnya: Bagian keuangan hanya membutuhkan data keuangan, jadi yang digambarkan hanya pandangan terhadap data keuangan saja, begitu juga dengan bagian akuntansi, hanya membutuhkan data akuntansi saja. Jadi tidak semua pengguna database membutuhkan seluruh informasi yang terdapat dalam database tersebut.

Contoh :


SISTEM INFORMASI JURUSAN
DATA MAHASISWA
Tambah
Simpan
Batal
Tutup

BAB IV

MODEL DATA

1. Pengertian Model Data

Model data adalah kumpulan perangkat konseptual untuk menggambar data, hubungan antara data, makna (semantik) data, dan batasan data.

Model database adalah suatu konsep yang terintegrasi dalam menggambarkan hubungan (*relationships*) antar data dan batasan-batasan (*constraint*) data dalam suatu sistem database.

Ada dua cara dalam merepresentasikan model data dalam membuat basis data, yaitu :

a. Record-Based Logical Models

Adalah record atau rekaman yang menjelaskan kepada pemakai mengenai hubungan logik antar data dalam basis data.

Record based logical model terbagi atas 3 yaitu :

- Hierarchical Model
- Network Model
- Relational Model

b. Object-Based Logical Models

Adalah himpunan data dan prosedur/relasi yang menjelaskan hubungan logic antar data dalam suatu basis data berdasarkan obyek datanya.

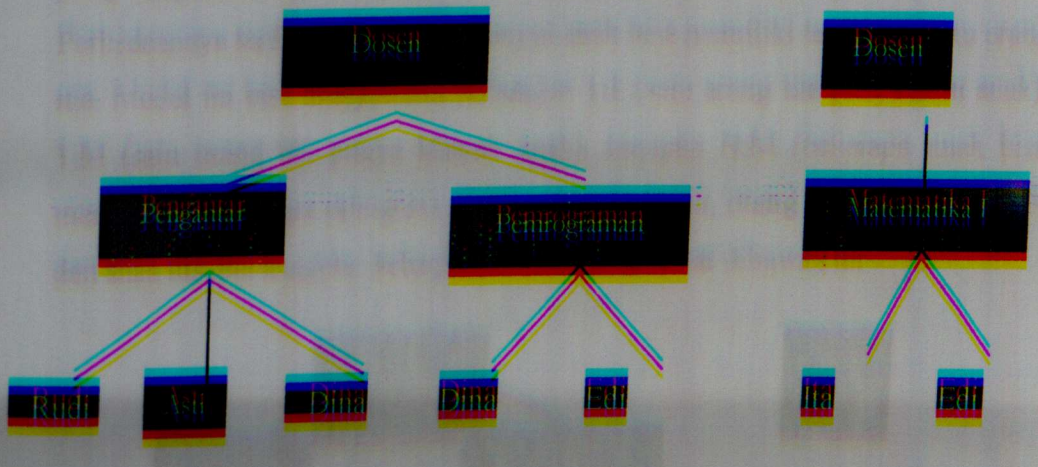
Object based logical model terbagi atas 3 yaitu :

- Semantik Model
- Entity Relationship Model
- Object Oriented Model

2. Hierarchical Model

Model hirarki atau biasa disebut model pohon, karena menyerupai pohon. Model ini menggunakan pola hubungan dengan istilah orang tua dan anak. Terdapat juga istilah simpul (bercirikan kotak atau lingkaran). Simpul yang berada diatas yang terhubung ke simpul pada level dibawahnya disebut orang tua. Setiap orang tua bisa memiliki satu (hubungan 1:1) atau beberapa anak (hubungan 1:M),

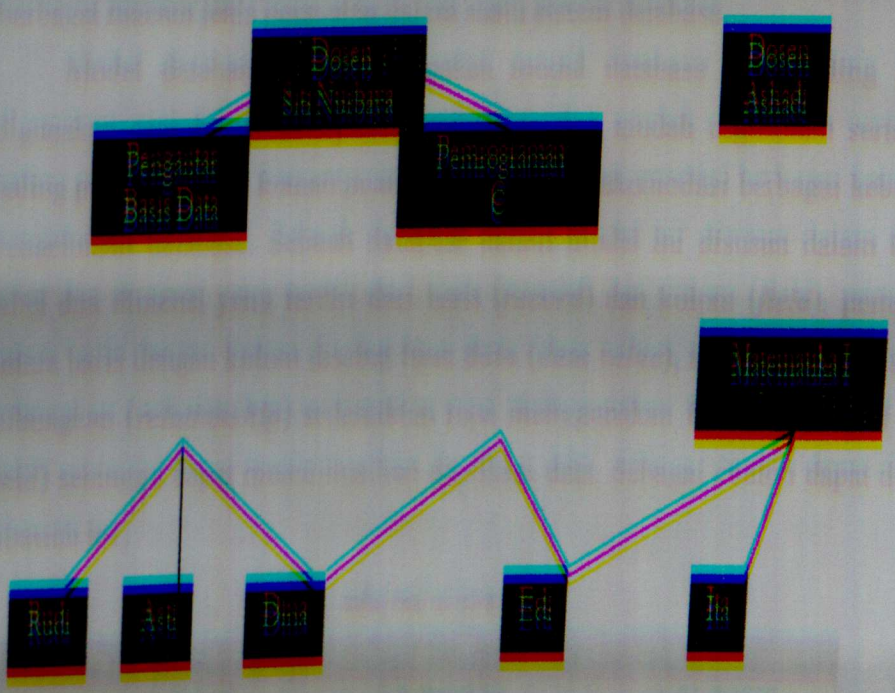
tetapi setiap anak hanya memiliki satu orang tua. Simpul = simpul yang dibawahi oleh simpul orang tua disebut anak. Adapun hubungan antara anak dan orang tua disebut cabang. Perbedaannya adalah, record-record diorganisasikan sebagai tree (pohon) daripada graf. Sebagai contoh lihat gambar seperti di bawah ini :



Gambar 6. Model Hirarki

3. Network Model

Model jaringan direpresentasikan dengan sekumpulan record dan relasi antar data yang direpresentasikan oleh record & link. Model ini menyerupai model hirarki. Perbedaannya terdapat pada suatu simpul anak bisa memiliki lebih dari satu orang tua. Model ini bisa menyatakan hubungan 1:1 (satu arang tua punya satu anak), 1:M (satu orang tua punya banyak anak), maupun N:M (beberapa anak bisa mempunyai beberapa orangtua). Pada model jaringan, orang tua disebut pemilik dan anak disebut anggota. Sebagai Contoh lihat gambar dibawah ini :



Gambar 7. Model Jaringan

4. Relational Model

Model relasional berbeda dengan model jaringan & hirarki. Pada model data relasional pemodelan menggunakan tabel untuk merepresentasikan data & relasi antar data. Setiap tabel terdiri atas kolom, dan setiap kolom mempunyai nama variable tertentu.Inti dari model ini adalah relasi, yang dimisalkan sebagai himpunan dari record. Pada model relasional, skema atau deskripsi data pada model relasi ditentukan oleh nama, nama dari tiap field (Atribut atau kolom), dan tipe dari tiap field.

Model Relasional merupakan model yang paling sederhana sehingga mudah digunakan dan dipahami oleh pengguna. Model ini menggunakan sekumpulan tabel berdimensi dua (yang disebut relasi atau tabel), dengan masing-masing relasi tersusun atas tupel atau baris dan atribut. DBMS yang bermodelkan relasional biasa disebut RDBMS (*Relational Data Base Management System*). Model database ini dikemukakan pertamakali oleh EF codd, seorang pakar basisdata. Model ini sering disebut juga dengan database relasi.

Model database hirarki dan jaringan merupakan model database yang tidak banyak lagi dipakai saat ini, karena adanya berbagai kelemahan dan hanya cocok untuk struktur hirarki dan jaringan saja. Artinya tidak mengakomodir untuk berbagai macam jenis persoalan dalam suatu sistem database.

Model database relasi merupakan model database yang paling banyak digunakan saat ini, karena paling sederhana dan mudah digunakan serta yang paling penting adalah kemampuannya dalam mengakomodasi berbagai kebutuhan pengelolaan database. Sebuah database dalam model ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (*record*) dan kolom (*field*), pertemuan antara baris dengan kolom disebut **item data** (*data value*), table-tabel yang ada di hubungkan (*relationship*) sedemikian rupa menggunakan **field-field kunci** (*Key field*) sehingga dapat meminimalkan duplikasi data. Sebagai contoh dapat dilihat dibawah ini :

tabel mahasiswa

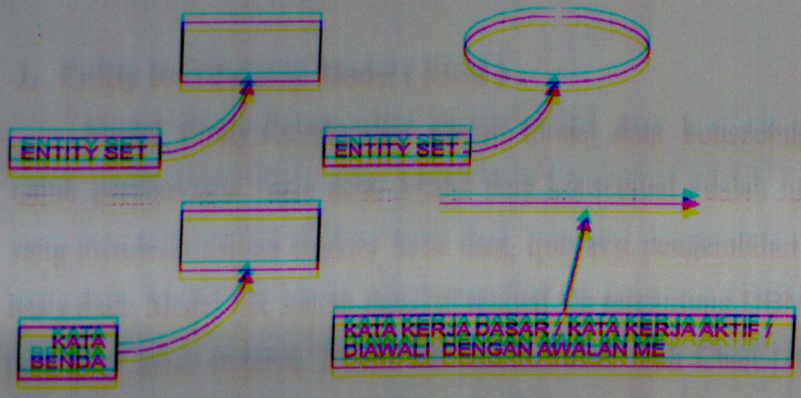
Nim	Nama_mhs	Alamat_mhs	Tgl_lahir
011234	Ahmad	Jl. Melati 50	21-3-1980
011345	Bobby	Jl. Mawar 103	13-5-1980
011456	Charles	Jl. Mangga 145	17-8-1980

5. Semantik Model

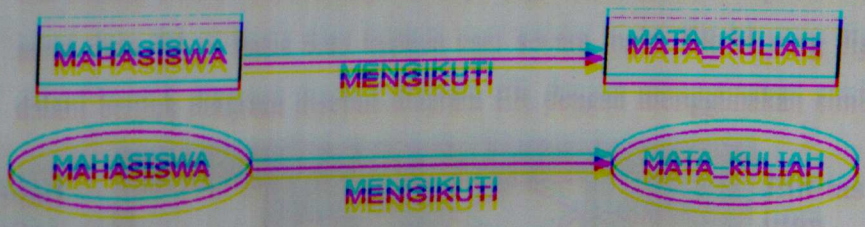
Model Semantik adalah model yang menggambarkan hubungan antara data yang ada dan tidak menggambarkan proses-proses yang terjadi. Model semantik hampir sama dengan entity relationship model. Perbedaannya hanya terletak pada pernyataan adanya relasi antara obyeknya. Jika pada entity relationship model menyatakan adanya relasi antar objek menggunakan simbol-simbol namun pada semantic model menggunakan kata-kata (semantik).

Tanda-tanda yang digunakan dalam semantic data model adalah sebagai berikut :

a. Menunjukkan adanya entitas
b. Menunjukkan adanya relasi berupa garis
c. Menunjukkan adanya atribut



Contoh :



BAB V

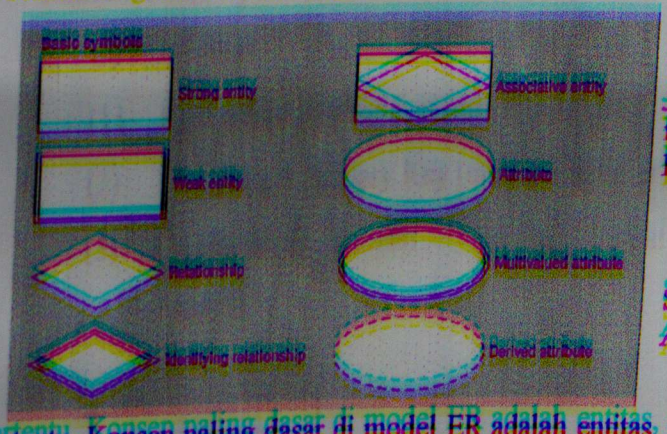
ENTITY RELATIONSHIP DIAGRAM (ERD)

1. Entity Relationship Model (ERM)

Model Entity-Relationship adalah model data konseptual tingkat tinggi untuk perancangan basis data. Model data konseptual adalah himpunan konsep yang mendeskripsikan struktur basis data, transaksi pengambilan dan pembaruan basis data. Model ER adalah data konseptual tak tergantung DBMS dan platform perangkat keras tertentu. Model ER dikemukakan oleh Chen [1976]. Sejak itu, telah memperoleh banyak perhatian dan perluasan.

Model ER adalah persepsi terhadap dunia nyata sebagai terdiri objek- objek dasar yang disebut entitas dan keterhubungan (*relationship*) antar entitas-entitas itu. Model Entity relationship biasanya digunakan untuk menjelaskan hubungan antar data dalam basis data kepada user secara logic. ER-M yang digambarkan dalam bentuk diagram disebut diagram ER dengan menggunakan simbol-simbol

Simbol Entitas



Entitas yang juga merupakan Relasi

Simbol Relasi

Simbol Atribut

grafis tertentu. Konsep paling dasar di model ER adalah entitas, relationship dan atribut.

Komponen-komponen utama model ER adalah sebagai berikut :

a. Entitas (*entity*),

Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sebuah kursi yang kita duduki, seseorang yang menjadi pegawai di sebuah perusahaan dan sebuah mobil yang melintas di depan kita adalah entitas.

Sekelompok entitas yang sejenis dan berada dalam lingkup yang sama

membentuk sebuah himpunan entitas (entity sets). Sederhananya, entitas menunjuk pada individu suatu objek, sedang himpunan entitas menunjuk pada rumpun (*family*) dari individu tersebut.

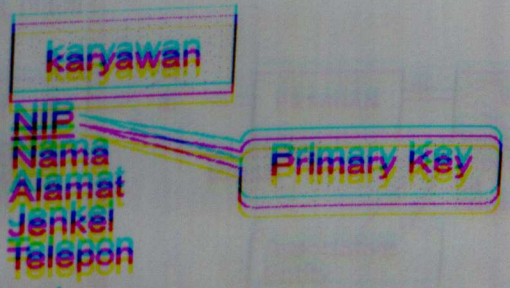
Seorang pasien, misalnya akan dimasukkan dalam himpunan entitas pasien. Sedang seorang dokter akan ditempatkan dalam himpunan entitas dokter.

1) Adapun syarat-syarat sebuah Entitas antara lain sebagai berikut :
   a) Merupakan objek yang memiliki lebih dari satu entity instances (contoh) dalam database. Entity Instance untuk Entitas Mahasiswa adalah Rika, Andi, Della, dll
   b) Merupakan objek yang memiliki beberapa atribut.
   c) Bukan seorang user dari sistem.
   d) Bukan sebuah output dari sistem (contoh: laporan)
   e) Beri nama dengan kata Benda

2) Jenis Entitas

   Entitas terbagi 2 jenis yaitu sebagai berikut :

   a) Strong Entitas
      (1) Keberadaanya berdiri sendiri.
      (2) Mempunyai Primary Key (unique identifier)
      (3) Digambarkan dengan Persegi Empat dengan Garis Tunggal.

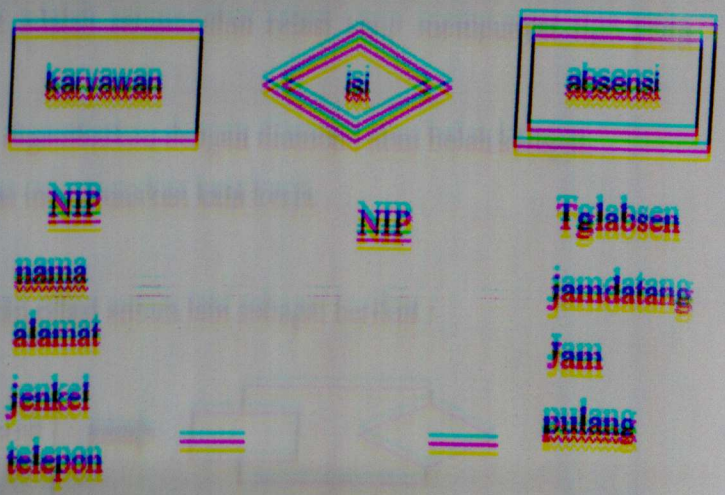


   Contoh :

   b) Weak Entitas
      (1) Tergantung pada strong entity
      (2) Tidak Dapat berdiri sendiri.
      (3) Tidak Mempunyai Primary Key (unique identifier)

(4) Digambarkan dengan Persegi Empat dengan Garis double.

Contoh :



c) Associative Entitas

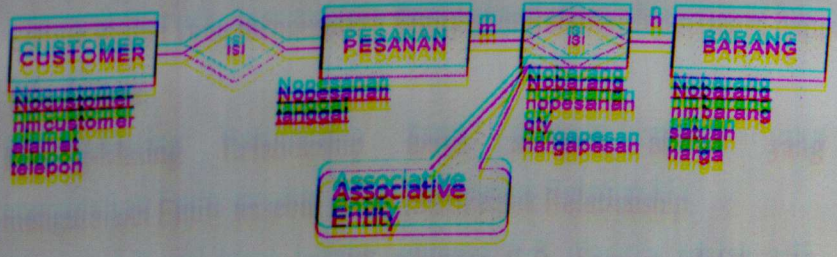
- Merupakan entity yang mempunyai attributes
- Dan merupakan relationship merupakan penghubung entitas.
- Kapan sebaiknya relationship dengan attribut menjadi sebuah associative entity ?

(1) Semua Relationships pada associative entity harus many

(2) The associative entity bisa mempunyai arti tidak terikat pada Entity lain

(3) The associative entity Lebih disukai mempunyai unique identifier, dan juga harus mempunyai attributes lain.

Contoh :



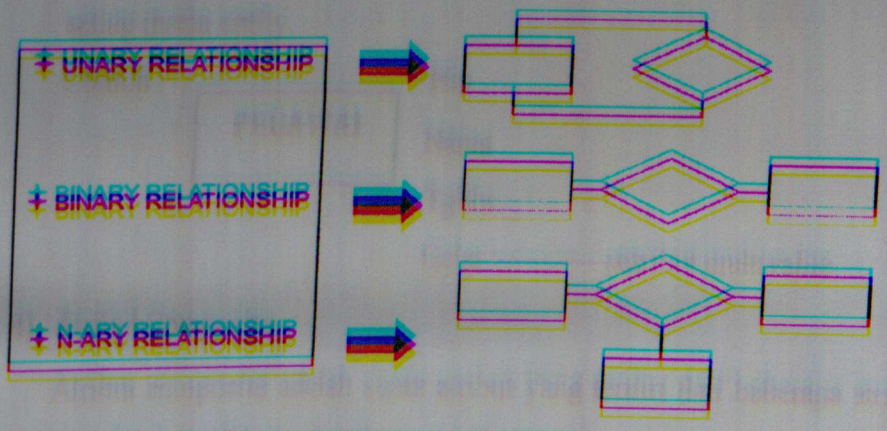
b. Relationship

Relationship memodelkan koneksi/hubungan di antara entitas-entitas. Relationship adalah hubungan diantara beberapa entiti. Kumpulan semua

relasi diantara entitas-entitas yang terdapat pada himpunan entitas- himpuan entitas tersebut membentuk himpunan relasi (*relationship sets*). Adapun syarat-syarat sebuah relasi adalah sebagai berikut :

1) Relationship set adalah sekumpulan relasi yang mempunyai tipe yang sama.
2) Relationship set digambarkan dengan diamond atau belah ketupat.
3) Nama relasi harus menggunakan kata kerja.

Ada 3 bentuk hubungan relasi antara lain sebagai berikut :



c. Atribut –atribut (properti –properti)

Setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik property atau atribut dari sebuah entitas tersebut. Penentuan / pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lainnya dalam pembentukan model ER. Contoh : nim, nama, alamat, kode dll.
Adapun syarat-syarat sebuah atribut adalah sebagai berikut :

1) Nama atribut harus unik di dalam sistem.
2) Semua atribut yang menguraikan Entity atau Relationship tertentu harus diberi nama.
3) Masing-Masing Relationship harus meliputi atribut yang menguraikan Entity tersebut dalam membentuk Relationship.
4) Nama penuh arti harus terpilih sehingga E-R diagram adalah self-explanatory (menjelaskan isi dari dirinya)

Jenis-jenis atribut adalah sebagai berikut :

1) Atribut Key

Atribut key adalah atribut yang digunakan untuk menentukan suatu entity secara unik.

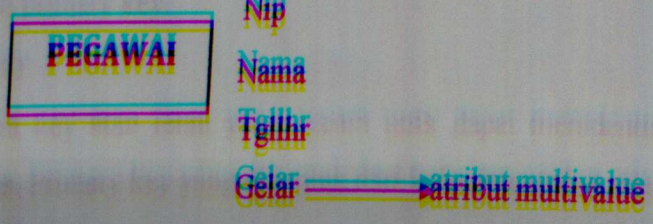
2) Atribut Simple

Atribut simple adalah atribut yang bernilai tunggal. Contoh : kota

3) Atribut Multivalue

Atribut Multivalue adalah atribut yang memiliki sekelompok nilai untuk setiap instan entity.

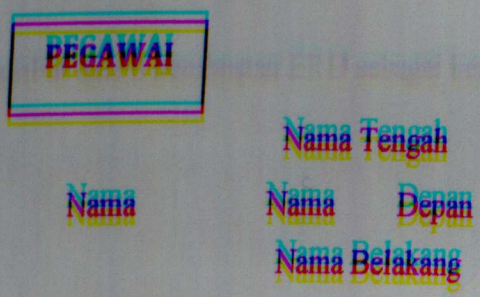
Contoh :



4) Atribut Composite

Atribut composite adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.

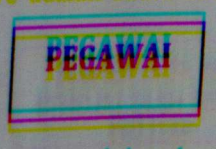
Contoh :



5) Atribut Derivatif

Atribut derivative adalah suatu atribut yang dihasilkan dari atribut yang lain.

Contoh :



Adapun jenis-jenis key antara lain sebagai berikut :

1) Primary key

Salah satu atrribut dari candidat key dapat dipilih menjadi primary key dengan 3 kriteria sbb :

- Key tersebut lebih natural untuk dijadikan acuan
- Key tersebut lebih sederhana

- Bersifat unik

2) Secondary key

Secondary key adalah key yang tidak terpilih menjadi primary key.

3) Foreign key

Jika sebuah primary key terhubungan ke table/entity lain, maka keberadaan primary key pada entity tersebut disebut sebagai foreign key.

4) Candidate key

Sebuah attribute atau lebih yang secara unit mengidentifikasi sebuah record, disebut candidate key. Attribute ini mempunyai nilai yang unik pada hampir setiap recordnya. Fungsi dari candidate key ini adalah sebagai calon primary key.

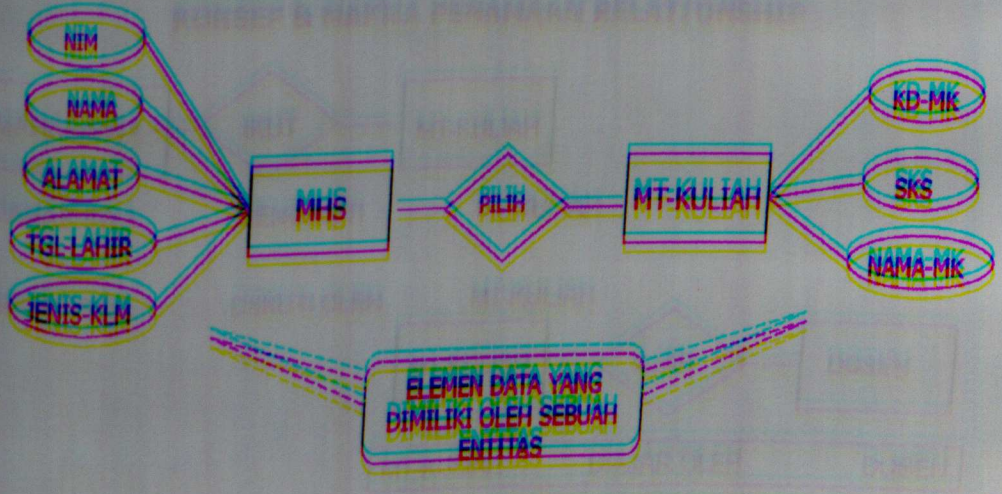
5) Composite key

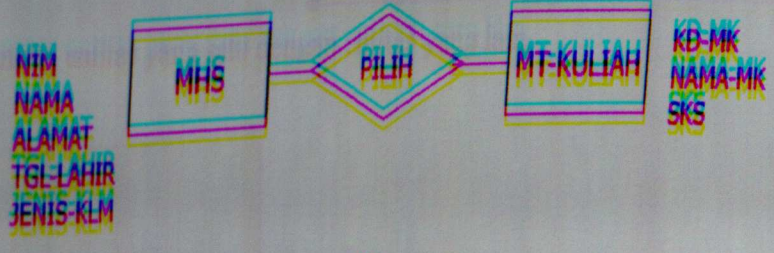
gabungan dua key atau lebih yang secara unik dapat menidentiifikasi sebuah entitas. Primary key yang dibentuk dari beberapa atribut relasi.

6) Alternate key

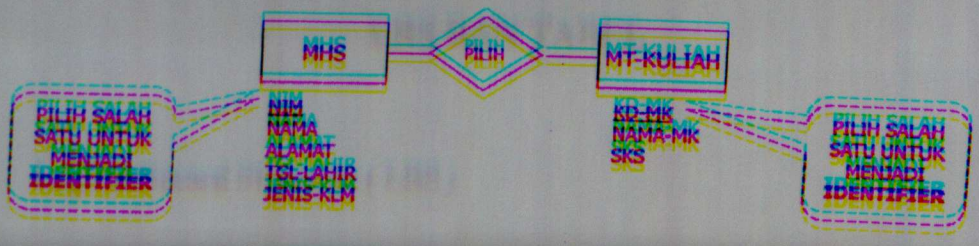
Setiap atribut dari candidate key yang tidak terpilih sebagai primary key akan dinamakan alternate key.

Contoh penulisan atribut dalam pembuatan ERD sebagai berikut :





Contoh penulisan key dalam pembuatan ERD sebagai berikut :



d. Kardinalitas / derajat relasi

Kardinalitas atau derajat relasi adalah tingkat hubungan dan derajat relasi

Notasi	Derajat Relasi Minimum-Maksimum
atau atau	(0 : N)
atau atau	(1 : N)
atau	(1 : 1)
atau	(0 : 1)

ATAU

- ONE TO ONE (1 : 1)
- ONE TO MANY (1 : M) atau (M : 1)
- MANY TO MANY (M : N)

KONSEP & MAKNA PENAMAAN RELATIONSHIP

MAHASISWA MENGIKUTI MT-KULIAH

MAHASISWA DIIKUTI OLEH MT-KULIAH

MT-KULIAH DIAJAR OLEH DOSEN

MT-KULIAH MENGAJAR DOSEN

antara entitas yang satu dengan entitas yang lain.

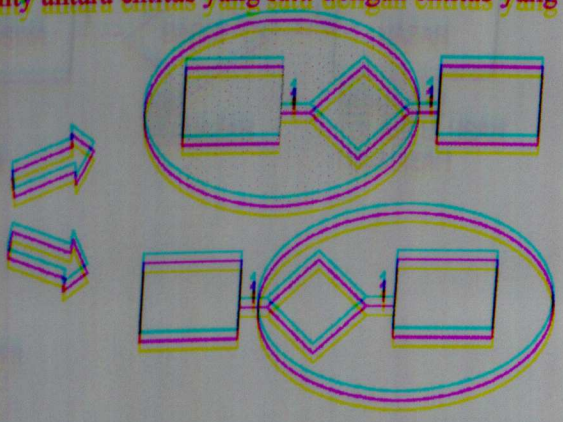
BAB VI

LRS DAN TABEL

1. Logical Record Structure (LRS)

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah 1:1

Pilih Salah Satu



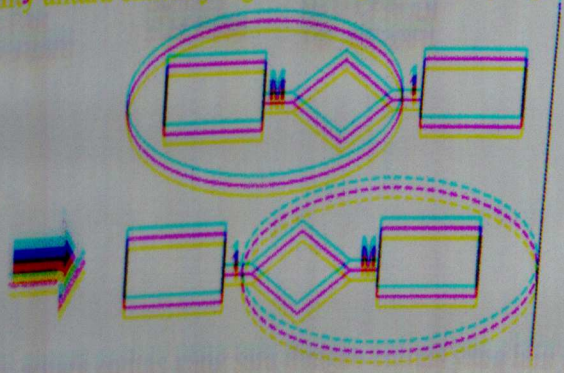
Cara penggabungan :

k) Ke entity yang membutuhkan referensi

l) Ke entity yang mempunyai atribut lebih sedikit

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah 1:M

Pilih Salah Satu



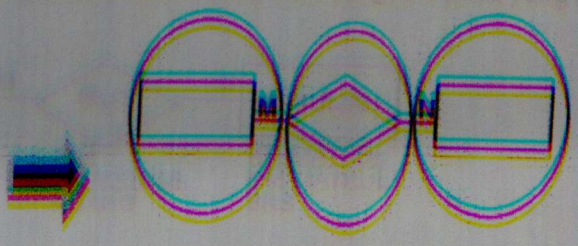
Cara penggabungan :

m) Selalu kearah many

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah

M : N

Pilih Salah Satu

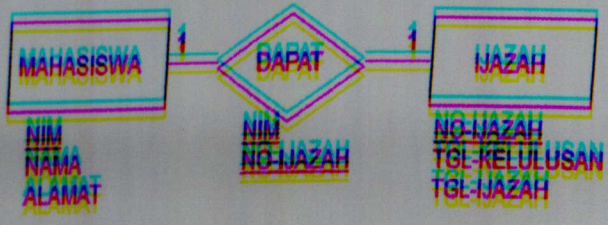


Cara penggabungan :

n) Entitas masing-masing berdiri sendiri

2. Tabel

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah



1:1

Jd tabelnya :

o) Tabel mahasiswa

p) Tabel ijazah

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah

1:M

Jd tabelnya :

q) Tabel fakultas

r) Tabel prodi

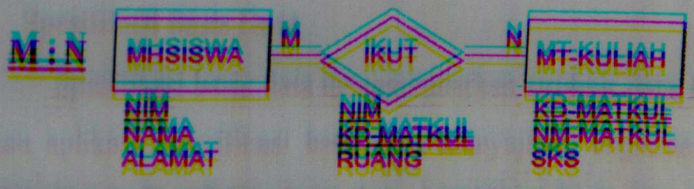
Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah

Jd tabelnya :

q) Tabel fakultas

r) Tabel prodi

Jika hubungan kardinality antara entitas yang satu dengan entitas yang lain adalah



Jd Tabelnya :

s) Tabel MAHASISWA

t) Tabel MT-KULIAH

u) Tabel IKUT

BAB VII

SPESIFIKASI BASIS DATA

1. Spesifikasi Basis Data

Spesifikasi basis data menggambarkan struktur data fisik pada suatu sistem atau aplikasi. Spesifikasi basis data menyajikan bagaimana penyimpanan data dilakukan pada software basis data. Spesifikasi basis data dibuatkan berdasarkan table yang ada.

Contoh :

Tabel Mahasiswa

NIM	NAMA	ALAMAT	KOTA
PK			

Jika tabel diatas dibuatkan spesifikasi basis data, maka akan menjadi seperti dibawah ini :

SPESIFIKASI BASIS DATA

NAMA FILE : Diambil dari Nama Relasi

MEDIA

ISI

ORGANISASI

PRIMARY KEY

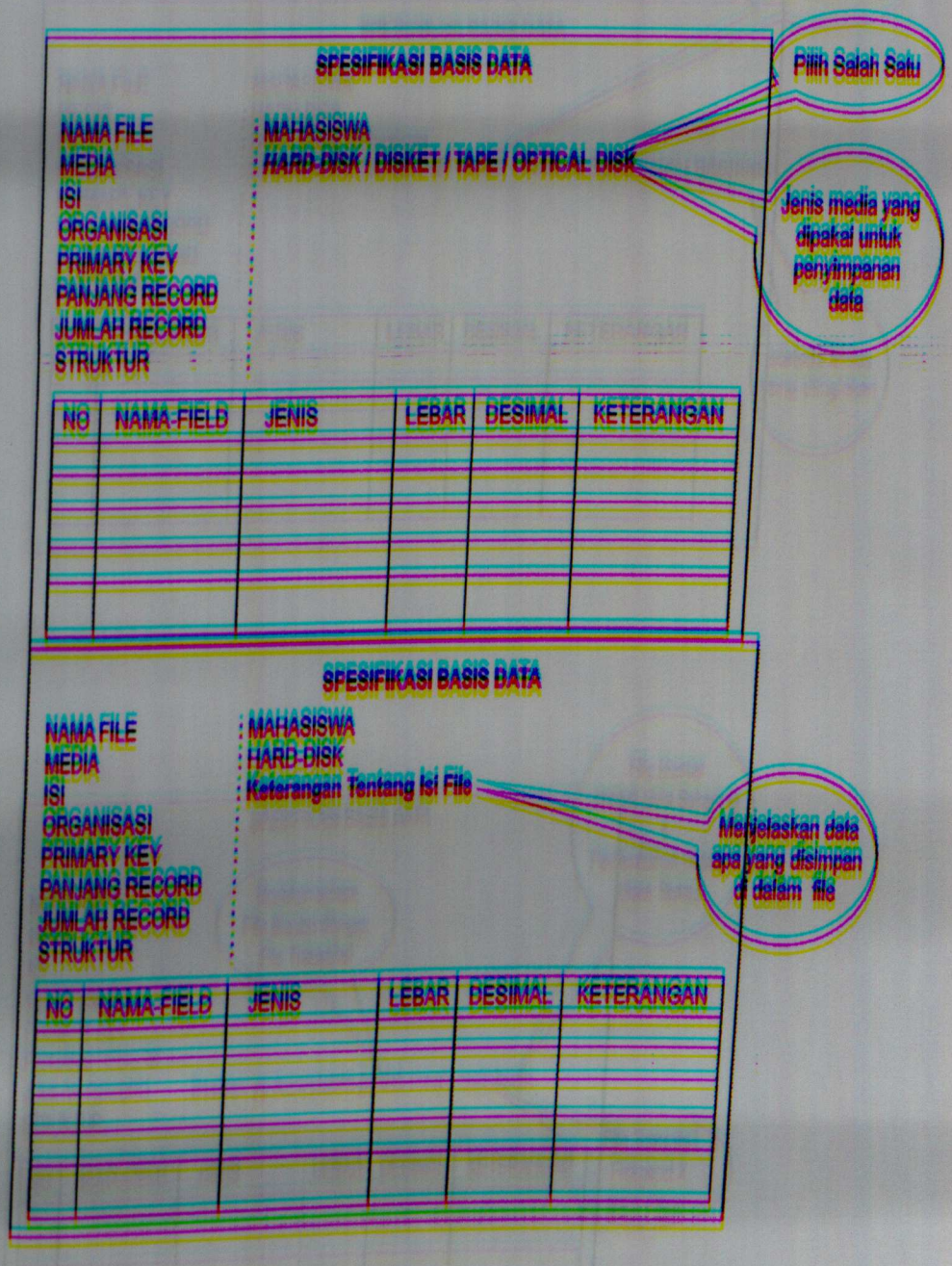
PANJANG RECORD

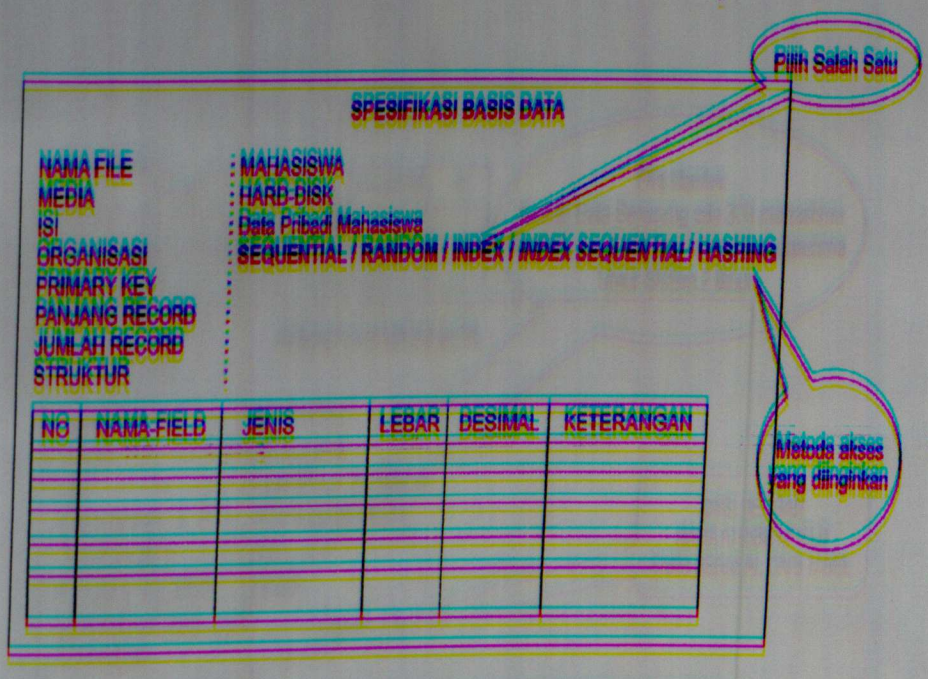
JUMLAH RECORD

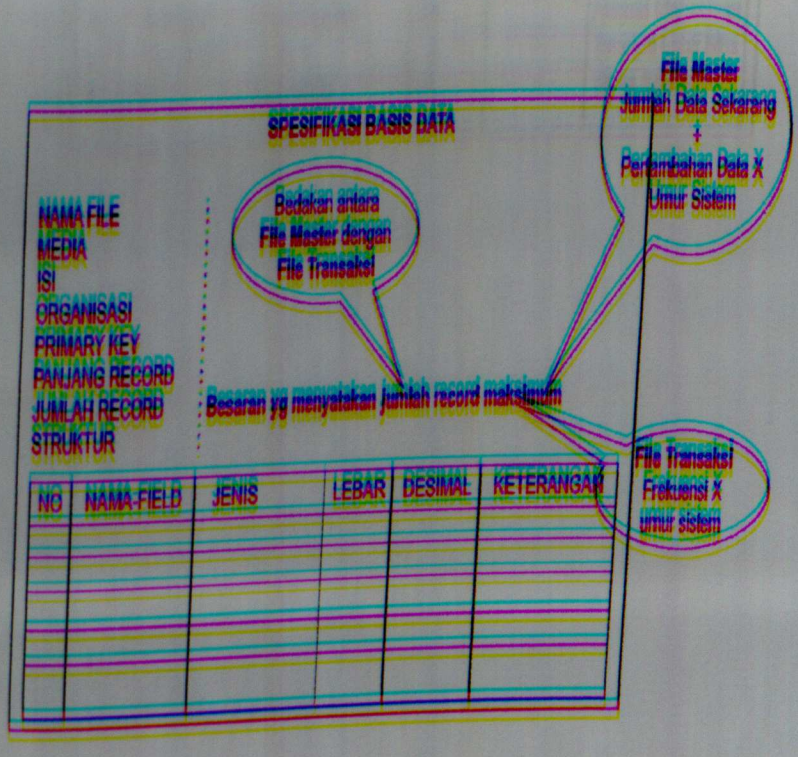
STRUKTUR

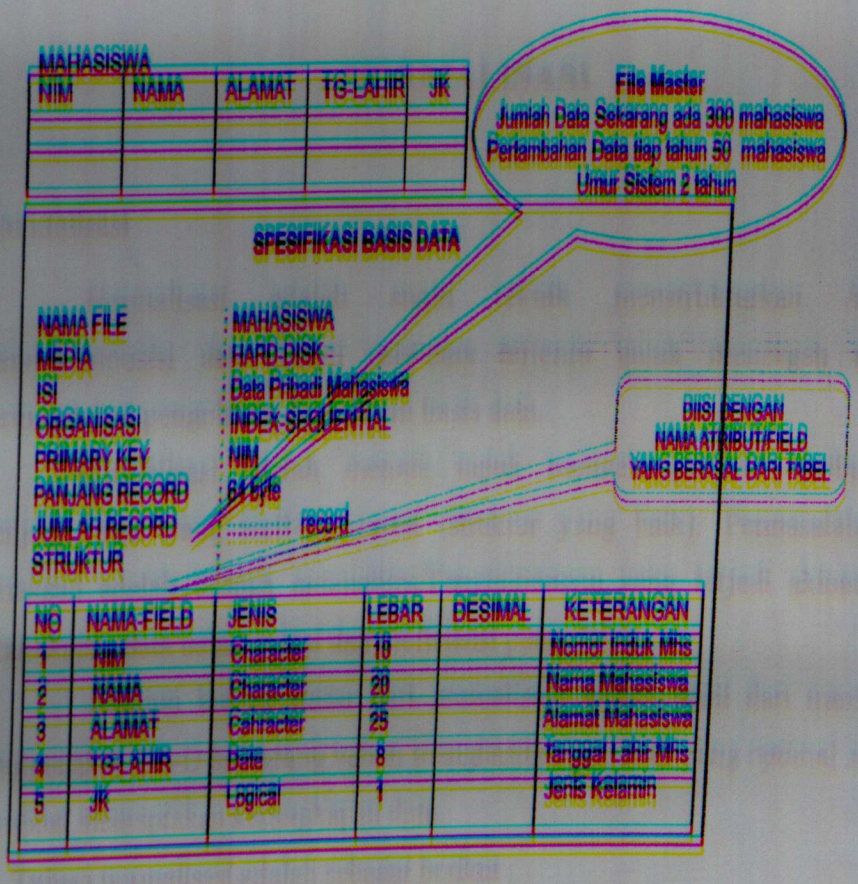
Boleh memakai nama yg berbeda asal dituliskan aliasnya pada Kamus Data

NO	NAMA-FIELD	JENIS	LEBAR	DESIMAL	KETERANGAN


SPESIFIKASI BASIS DATA
NAMA FILE : MAHASISWA
MEDIA : HARD-DISK / DISKET / TAPE / OPTICAL DISK
ISI
ORGANISASI
PRIMARY KEY
PANJANG RECORD
JUMLAH RECORD
STRUKTUR
NO
NAMA-FIELD
JENIS
LEBAR
DESIMAL
KETERANGAN
Pilih Salah Satu
Jenis media yang dipakai untuk penyimpanan data
SPESIFIKASI BASIS DATA
NAMA FILE : MAHASISWA
MEDIA : HARD-DISK
ISI : Keterangan Tentang Isi File
ORGANISASI
PRIMARY KEY
PANJANG RECORD
JUMLAH RECORD
STRUKTUR
NO
NAMA-FIELD
JENIS
LEBAR
DESIMAL
KETERANGAN
Menjelaskan data apa yang disimpan di dalam file


Pilih Salah Satu
SPESIFIKASI BASIS DATA
NAMA FILE : MAHASISWA
MEDIA : HARD DISK
ISI : Data Pribadi Mahasiswa
ORGANISASI : SEQUENTIAL / RANDOM / INDEX / INDEX SEQUENTIAL/ HASHING
PRIMARY KEY :
PANJANG RECORD :
JUMLAH RECORD :
STRUKTUR :
NO
NAMA-FIELD
JENIS
LEBAR
DESIMAL
KETERANGAN
Metoda akses yang diinginkan


File Master
Jumlah Data Sekarang
+
Pertambahan Data X
Umur Sistem
SPESIFIKASI BASIS DATA
Bedakan antara File Master dengan File Transaksi
NAMA FILE
MEDIA
ISI
ORGANISASI
PRIMARY KEY
PANJANG RECORD
JUMLAH RECORD
STRUKTUR
Besaran yg menyatakan jumlah record maksimum
File Transaksi
Frekuensi X
umur sistem
NO
NAMA-FIELD
JENIS
LEBAR
DESIMAL
KETERANGAN


MAHASISWA
NIM
NAMA
ALAMAT
TG-LAHIR
JK
File Master
Jumlah Data Sekarang ada 300 mahasiswa
Pertambahan Data tiap tahun 50 mahasiswa
Umur Sistem 2 tahun
SPESIFIKASI BASIS DATA
NAMA FILE : MAHASISWA
MEDIA : HARD-DISK
ISI : Data Pribadi Mahasiswa
ORGANISASI : INDEX-SEQUENTIAL
PRIMARY KEY : NIM
PANJANG RECORD : 84 byte
JUMLAH RECORD : ..... record
STRUKTUR :
DIISI DENGAN
NAMA ATRIBUT/FIELD
YANG BERASAL DARI TABEL
NO
NAMA-FIELD
JENIS
LEBAR
DESIMAL
KETERANGAN
1
NIM
Character
10
Nomor Induk Mhs
2
NAMA
Character
20
Nama Mahasiswa
3
ALAMAT
Cahracter
25
Alamat Mahasiswa
4
TG-LAHIR
Date
8
Tanggal Lahir Mhs
5
JK
Logical
1
Jenis Kelamin

BAB VIII

NORMALISASI

1. Normalisasi

Normalisasi adalah suatu teknik menstrukturkan /memecah/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data.

Normalisasi adalah metode untuk mengelompokkan attribut-attribut menjadi relasi yang well-structured (struktur yang baik). Permasalahan yang dimaksud adalah adanya anomallies/Penyimpangan yang terjadi akibat adanya kerangkapan data dalam relasi dan inefisiensi pengolahan

Adapun konsep dasar dari normalisasi adalah hasil dari transformasi menggunakan ER-D terkadang belum menghasilkan struktur yang optimal sehingga membuat menciptakan kerangkapan data.

Tujuan normalisasi adalah sebagai berikut :

a. Untuk menghindari duplikasi data

b. Merupakan proses mendekomposisikan relasi yang masih memiliki beberapa anomali untuk menghasilkan relasi yang lebih sederhana dan *well-structured*.

Sebuah relasi (relasi) yang memiliki data redundancy yang minimal dan memungkinkan user untuk melakukan insert, delete, dan update baris (record) tanpa menyebabkan inkonsistensi data.

a. Tujuannya untuk menghindari beberapa anomali:

1) Insertion Anomaly = menambah record baru mempengaruhi user untuk membuat duplikasi data
2) Deletion Anomaly = menghapus record mungkin menyebabkan hilangnya data yang akan dibutuhkan pada record lain

3) Modification Anomaly = merubah data pada sebuah record mempengaruhi perubahan pada record lain karena adanya duplikasi.

Contoh Relasi :

KARYAWAN

NIP	NAMA	BAGIAN	GAJI	KURSUS	SELESAI
100	AGUS	KEUANGAN	1,000	SPSS	10-10-2005
100	AGUS	KEUANGAN	1,000	AUDIT	10-10-2005
200	BUDI	SDM	1,200	KOMPUTER	20-10-2000
200	BUDI	SDM	1,200	AUDIT	10-10-2005
300	BAGONG	KEBERSIHAN	500		
400	MAMOX	MARKETING	1,500	MARKETING	01-01-2005

Penjelasan :

Insertion = tidak dapat memasukkan data karyawan baru yang tidak mengambil kursus

Deletion = jika pegawai 400 dihapus, kita akan kehilangan informasi tentang keberadaan kelas Marketing

Modification = menaikan gaji pegawai 100 mengharuskan kita untuk meng- update beberapa records

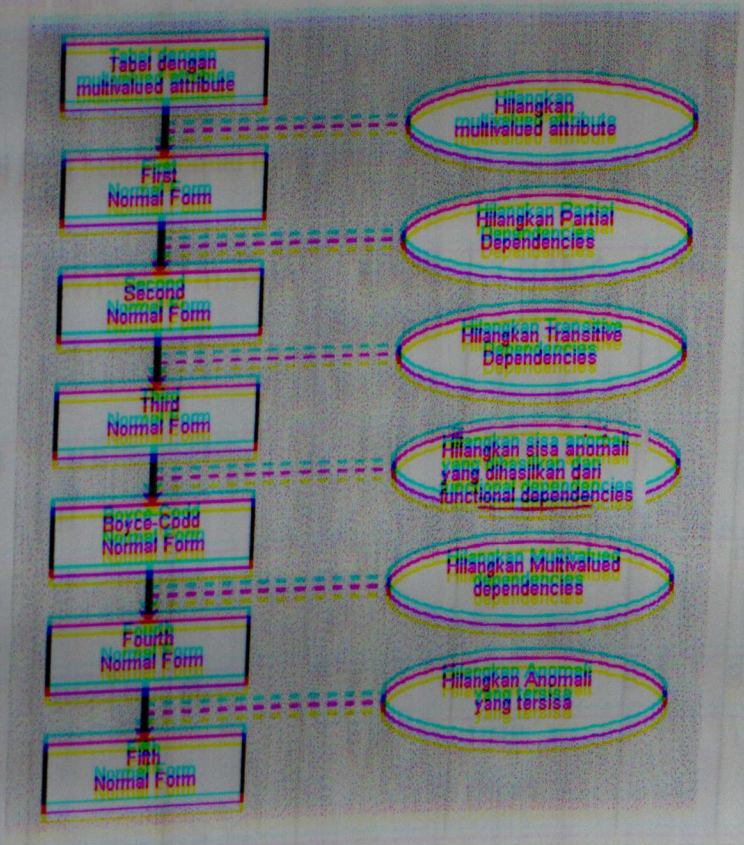
= Mengapa beberapa anomali ini muncul ? Karena kita telah menyatukan 2 tema (entity) dalam satu relasi. Hal ini menyebabkan adanya duplikasi, dan ketergantungan antar entitas.

2. Fuctional Dependency

Functional Dependency adalah nilai sebuah atribut (the *determinant*) yang menentukan nilai atribut yang lainnya. Normalisasi dilakukan berdasarkan analisa dari functional dependencies (relationship antar atribut). Setiap yang bukan key secara fungsional harus tergantung pada setiap candidate key (primary Key).

Contoh : Nim ⟹ Nama, Alamat

Langkah-langkah dalam normalisasi adalah sebagai berikut :



a. First Normal Form (1 NF)

Syarat Tahap 1 NF yaitu :

- Sudah tidak ada repeating group
- Tidak memiliki multivalued attributes
- Setiap nilai atribut hanya mempunyai nilai tunggal.

Contoh :

STMIK Atma Luhur

Laporan nilai semester ganjil 2012/2013

Nim : 0611502350
Nama : Desi R
Alamat : Pangkalpinang

Kode	Mata Kuliah	Sks	Nilai	Bobot
K1C	IMK	3	B	3
P1A	Bahasa Inggris	2	A	4

Relasi yang belum normal tahap pertama

NIM	NAMA	ALAMAT	KODE MATKUL	NAMA MATKUL	SKS	NILAI	BOBOT
0611500678	AGUSH	PKP	K3C	SBD	3	A	4
			P1A	BAHASAC	3	B	3
0622500567	DESI R	PKP	K1C	IMK	3	B	3
			P1A	BHS. INGG	2	A	4

Relasi yang sudah normal tahap pertama

NIM	NAMA	ALAMAT	KODE MATKUL	NAMA MATKUL	SKS	NILAI	BOBOT
0611500678	AGUSH	PKP	K3C	SBD	3	A	4
0611500678	AGUSH	PKP	P1A	BAHASAC	3	B	3

0622500567	DESI R	PKP	K1C	IMK	3	B	3
0622500567	DESI R	PKP	P1A	BHS.ING G	2	A	4

b. Second Normal Form (2 NF)

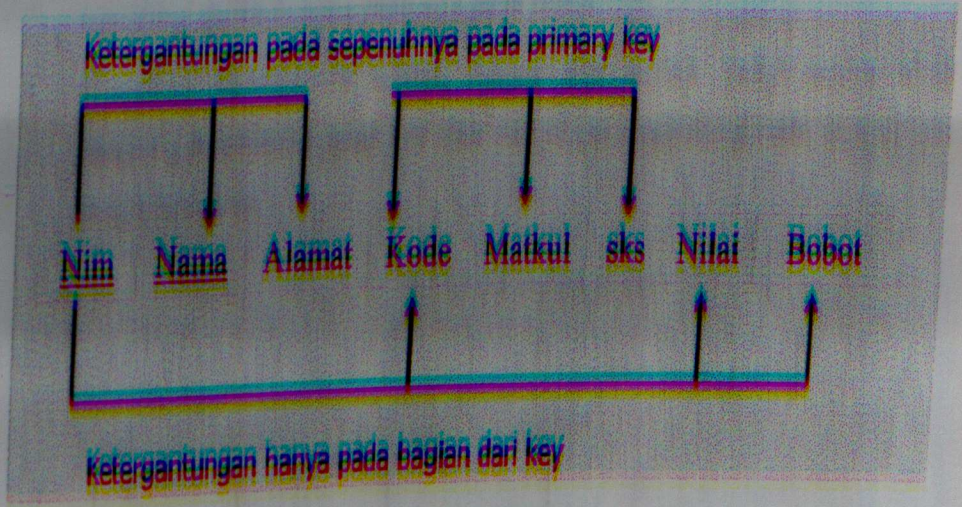
1) Sudah memenuhi 1NF dan setiap atribut non-key sepenuhnya secara fungsional tergantung pada semua primary key.

a. Setiap atribut non-key harus didefinisikan oleh semua key, bukan oleh bagian dari key.

b. Tidak memiliki partial functional dependencies atau penentuan identitas tunggal.

2) relasi dibawah ini belum dalam 2nd Normal Form (2NF)

Contoh :

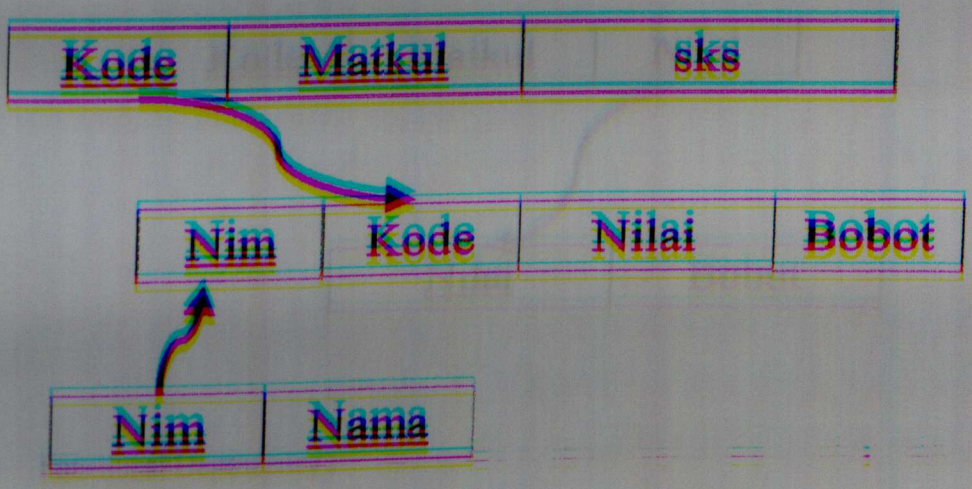


Functional Dependency :

- Nim → Nama, Alamat
- Kode → Matkul, sks
- Nim, Kode → Nilai, Bobot

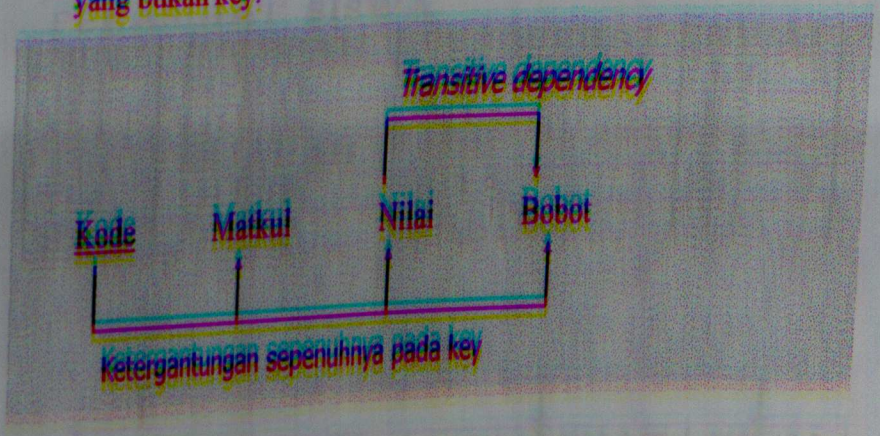
Maka tabel relasi masih belum normal ke-2 (2NF)

- Karena atribut Bobot belum tergantung pada primary key, maka atribut-atribut tsb harus dipisahkan dari relasi induknya.



c. Third Normal Form (3 NF)

- Dalam *Third Normal Form (3NF)* relasi harus dalam 2NF dan tidak ada *transitive dependencies* yang ada pada relasi.
- *Transitive dependency* adalah ketika ada atribut yang secara tidak langsung tergantung sama key dan atribut tsb tergantung pada atribut lain yang bukan key.

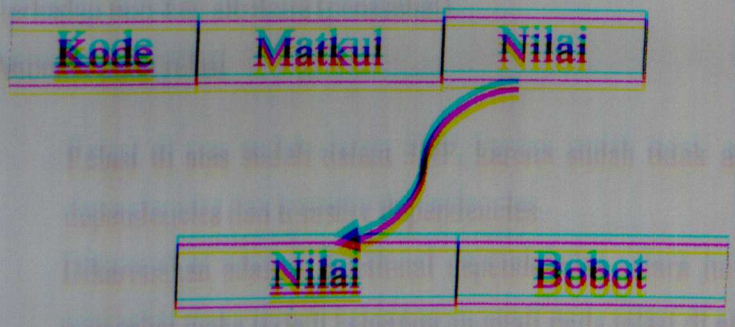


FD :

- Kode→Matkul, Nilai, Bobot

Maka relasi nilai masih belum normal ke-3 (3NF)

- Karena atribut Bobot masih tergantung pada atribut yang bukan keynya, maka atribut-atribut tsb harus dipisahkan dari relasi induknya.



Functional Dependency :

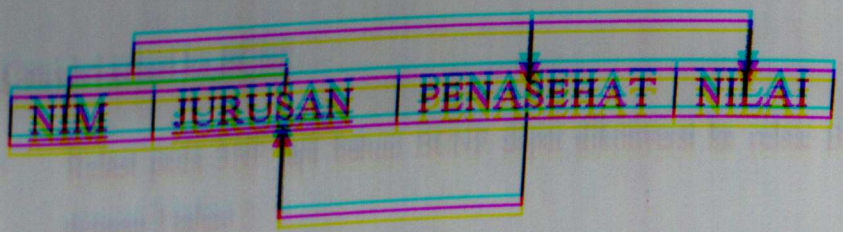
- Kode □Matkul, Nilai
- Nilai □Bobot

d. Boyce –Codd Normal Form (BCNF)

- Ketika sebuah relasi memiliki lebih dari 1 candidate keys, anomali bisa terjadi walau relasi sudah dalam 3NF
- Misal: relasi di bawah ini sudah normal ke-3

PENASIHAT_SISWA

NIM	JURUSAN	PENASEHAT	NILAI
123	FISIKA	PARTO	4.0
133	MUSIK	RITA	3.3
456	SEJARAH	RINI	3.2
789	MUSIK	RITA	3.7
678	FISIKA	PARTO	3.5



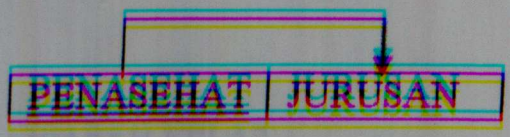
- Primary key pada relasi diatas adalah composite key yaitu nim dan jurusan.
- Tiap siswa boleh memilih lebih dari satu jurusan dan setiap siswa pada satu jurusan memiliki satu penasehat dan 1 nilai, kemudian satu jurusan hanya memiliki satu penasehat.
- Pada relasi di atas attribut key (jurusan) functional dependencies

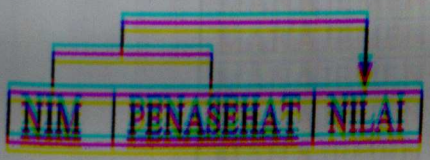
terhadap non-key attribute (penasehat).

- Anomali pada relasi
  - Relasi di atas sudah dalam 3NF, karena sudah tidak ada partial dependencies dan transitiv dependencies.
  - Dikarenakan adanya functional dependencies antara jurusan dan penasehat maka terjadi beberapa anomali pada relasi di atas :
    - Jika penasehat fisika diganti dari parto menjadi didi maka harus mengganti pada 2 record pada tabel (update anomali).
    - Jika ingin menambah baris untuk memasukan indri menjadi penasehat jurusan komputer, tidak dapat dilakukan jika tidak ada siswa yang ingin mengambil jurusan computer.
    - Jika siswa dengan nim 456 keluar dari sekolah, maka jurusan sejarah dan penasehatnya akan terhapus dari relasi.
- Anomali pada relasi di atas adalah hasil dari ketergantungan key attribut pada non key atribut(candidate key).
- Sedangkan BCNF menuntut untuk ketergantungan tetap ada dalam relasi, dan setiap key yang dijadikan determinant harus menjadi candidate key.
- Relasi di atas tidak dalam BCNF karena atribut penasehat bukan candidat key (tapi attribut key jurusan functional dependencies terhadap penasehat).

Convert relasi ke BCNF

- Relasi pada 3NF tapi belum BCNF dapat dikonversi ke relasi BCNF dengan 2 tahap :
  1. rubah relasi sehingga setiap determinant yang bukan candidate key menjadi komponen dari primary key.
  2. hasil dari perubahan diatas membuat relasi memiliki partial dependencies (belum 2NF), maka lakukan tahapan normalisasi tahap ke–2.





SISWA

NIM	PENASEHAT	NILAI
123	PARTO	4.0
123	RITA	3.3
456	RINI	3.2
789	RITA	3.7
678	PARTO	3.5

PENASIHAT

PENASEHAT	JURUSAN
PARTO	FISIKA
RITA	MUSIK
RINI	SEJARAH

e. Fourth Normal Form (4 NF)

- Walaupun BCNF mehilangkan anomali pada functional dependencies, jenis yang lain dari ketergantungan disebut multi- valued dependency dapat juga menyebabkan data redundancy.
- Pada user view dibawah ini terdapat beberapa repeating group:

PENAWARAN KELAS

KURSUS	INSTRUKTUR	TEXTBOOK
MANAJEMEN	PARTO DINI YADI	PETER HOFFMAN
KEUANGAN	TUTI	JONES CHANG

Penjelasan :

1. Tiap kursus memiliki beberapa instruktur.
2. Tiap kursus memiliki beberapa buku yang digunakan.
3. Tiap buku yang diberikan digunakan oleh tiap intruktur yang mengajar.

- Conversi dari user view di atas dengan mengisi cell yang masih kosong menjadikan relasi dalam 1NF

PENAWARAN KELAS

KURSUS	INSTRUKTUR	TEXTBOOK
MANAJEMEN	PARTO	PETER
MANAJEMEN	PARTO	HOFFMAN
MANAJEMEN	DINI	PETER
MANAJEMEN	DINI	HOFFMAN
MANAJEMEN	YADI	PETER
MANAJEMEN	YADI	HOFFMAN
KEUANGAN	TUTI	JONES
KEUANGAN	TUTI	CHANG

- Primary key table di atas adalah kombinasi dari 3 atribut (kursus+instruktur+textbook).
- Contoh ketergantungan di atas disebut multivalued depedencies dimana timbul ketika paling sedikit 3 atribut (contoh A, B dan C) pada sebuah relasi dan untuk tiap nilai A ada beberapa nilai B dan beberapa nilai untuk C tapi nilai B independent terhadap nilai C.
- Untuk menghilangkan multivalued dependencies, relasi dibagi menjadi 2 relasi baru.
- Relasi sekarang dalam 4NF dimana sudah BCNF dan tidak ada multivalued dependencies

INSTRUKTUR

KURSUS	INSTRUKTUR
MANAJEMEN	PARTO
MANAJEMEN	DINI
MANAJEMEN	YADI
KEUANGAN	TUTI

TEXTBOOK

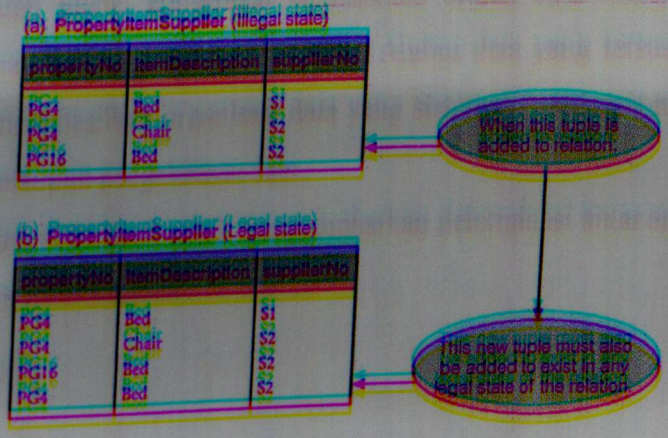
KURSUS	TEXTBOOK
MANAJEMEN	PETER
MANAJEMEN	HOFFMAN
KEUANGAN	JONES
KEUANGAN	CHANG

f. Fifth Normal Form (5 NF)

➢ Dekomposisi sebuah relasi ke dalam 2 relasi haru mempunyai lossless-join property, yang memastikan tidak ada baris tambahan

yang yang dihasilakan ketika dua relasi disatukan melalui operasi natural join.

- Namun, ada beberapa syarat untuk mendekomposisi sebuah relasi ke dalam lebih dari 2 tabel. Meski jarang, kasus seperti ini diatur dengan join dependency dan fifth normal form (5NF).
- 5NF : sebuah relasi yang tidak mempunyai join dependency.



- Karena relasi di atas mengandung join dependencies, maka relasi belum 5NF.
- Untuk menghilangkan join dependencies, decompose ke dalam 3 relasi 5NF.

PropertyItem

propertyNo	itemDescription
PG4	Bed
PG4	Chair
PG16	Bed

itemSupplier

itemDescription	supplierNo
Bed	S1
Chair	S2
Bed	S2

PropertySupplier

propertyNo	supplierNo
PG4	S1
PG4	S2
PG16	S2

BAB IX

DIAGRAM DETERMINAN DAN KODE DATA

3. Diagram Determinan

Determinan adalah faktor penentu antar atribut. Basis data yang bebas dari data rangkap adalah syarat utama yang harus dipenuhi dalam penyusunan basis data. Pembuatan model konseptual berdasarkan aturan data merupakan sarana untuk mencapai kondisi basis data tersebut. Aturan data yang tersusun baik dan cermat akan menghasilkan organisasi data yang sistematis sehingga mudah dalam penyimpanan ataupun pengaksesan data.

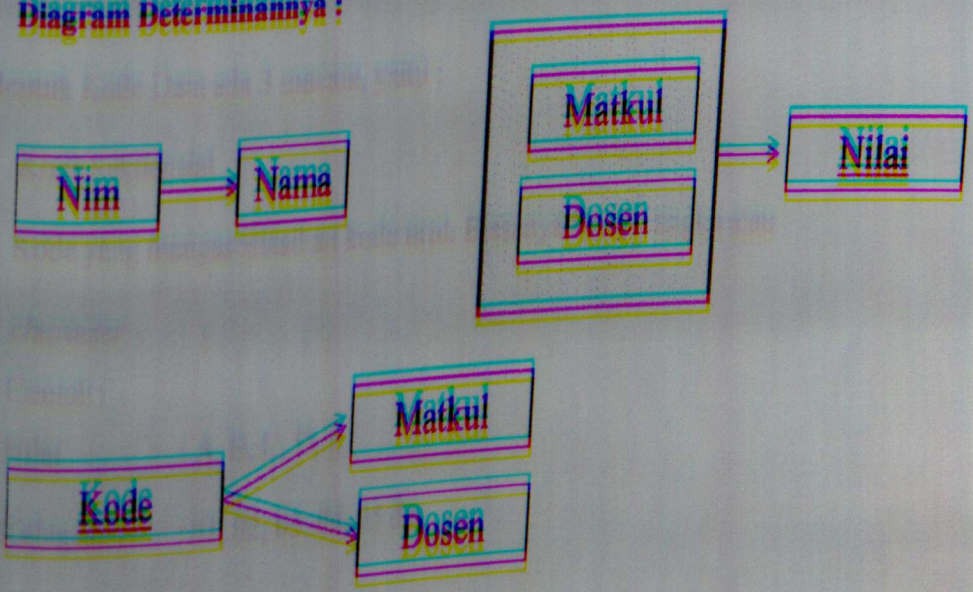
Cara yang paling sederhana untuk menggambarkan determinasi antar atribut adalah dengan diagram determinansi.

Contoh :

FD :

- Nim → nama
- Kode → Matkul,Dosen, Ruang
- Nim,Kode → Nilai

Diagram Determinannya :



4. Kode Data

Basis data yang optimal akan memberikan keuntungan, antara lain :

1. Efisiensi penggunaan media penyimpanan yang tinggi
2. Akses data dapat dilakukan dengan cepat dan mudah
3. Pemeliharaan dan pengolahan data dapat dilakukan dengan mudah
4. Terjaganya konsitensi dan integritas data

Kode yang baik memenuhi beberapa kriteria, antara lain :

1. Urut

   Beberapa contoh kode yang berurutan adalah Nomor_Induk_Mahasiswa, Nomor_Anggota, Nomor_Nota dsb.

2. Singkat / Pendek

   Kode yang baik tidak lebih dari 10 character

3. Dapat di-*generate* oleh sistem

   Para pemakai tidak harus mengetikkan kode pada saat *entry* data. Karena akan mengakibatkan kesalahan input diakibatkan oleh kesalahan manusia (*human error*)

Bentuk Kode Data ada 3 macam, yaitu :

1. Kode Sekuensial

   Kode yang mengasosiasikan kode urut. Biasanya berupa angka atau *character*.

   Contoh :

   Nilai : A, B, C, D, E

   Kabupaten : 01, 02, 03, 04, 05 dst.

2. Kode Mnemonic

   Kode yang dibentuk menggunakan singkatan atau huruf pertama data yang

dikodekan. Tersusun atas satu atau beberapa character.

Contoh :

Agama : I, K, P, H, B (Islam, Katolik, Protestan, Hindu, Budha)

Jenis Kelamin : L, P (Laki-laki, Perempuan)

Jurusan : MI, KA, TI, TK dst.

3. Kode Blok

Kode yang disusun membentuk blok kode dengan menggunakan format tertentu. Gabungan angka dan character.

Contoh :

NIM : TI-2002-I-1000

NIK : 1996-0372-100-E

Kode Barang : A-01-01-1000

BAB X

QUERY LANGUAGE

1. Pengertian Query Language

QUERY LANGUAGE adalah suatu bahasa yang digunakan user untuk mengambil informasi dari basis data atau digunakan untuk mengekspresika modifikasi ataupun queri terhadap data yang terkandung dalam suatu database relasional. Pada umumnya level bahasa ini lebih tinggi dari bahasa pemrograman standar.

QUERY LANGUAGES dibagi menjadi dua (2) kategori, yaitu :

- Procedural Language (What and how)
- Non-procedural Language (what; not how)

Dalam bahsa prosedural, user menginstruksikan ke sistem agar membentuk serangkaian operasi dalam basis data untuk mengeluarkan hasil yang diinginkan. Dalam bahasa non-prosedural, user mendeskripsikan informasi yang diinginkan tanpa memberikan prosedur detail untuk menghasilkan informasi tersebut. Sebagian besar system basis data relasional yang beredar dipasaran menawarkan bahasa query dengan pendekatan prosedural & non-prosedural. QUERY LANGUAGES dapat dibagi menjadi dua

(2) tipe, yaitu :

a. Formal Query Languages, terdiri dari :

1) Relational Algebra, adalah procedural query language atau Relational Algebra(aljabar relasional) merupakan kumpulan operasi terhadap relasi dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru dan termasuk kategori prosedural dan juga menyediakan seperangkat operator untuk memanipulasi data.
2) Relational Calculus, adalah non-procedural query language dan Pemakai mendiskripsikan informasi yang dikehendaki tanpa memberikan prosedur (deret operasi) spesifik untuk memperoleh informasi. Pada model relasional, bahasa formal non prosedural adalah bahasa kalkulus (predikat(relasional yaitu diekspresikan dengan menspesifikasikan predikat terhadap tuple atau domain yang harus dipenuhi.

b. Commercial Query Languages, terdiri dari :

1) SQL (Structured Query Language), adalah kombinasi procedural dan non-procedural language.
2) QBE (Query By Example), adalah non-procedural query yang berbasis pada domain relational calculus.
3) QUEL, adalah non-procedural query language yang berbasis pada tuple relational calculus.

2. Relational Algebra (Aljabar Relasional)

Aljabar relasional (Relational Algebra) adalah relational operation yang digunakan untuk memanipulasi data pada basis data relasional. Aljabar relasional merupakan kumpulan operasi terhadap relasi di mana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru. Aljabar relasional termasuk dalam kategori bahasa prosedural yang Menyediakan seperangkat operator untuk memanipulasi data.

Ada 2 operator dalam aljabar relasional, yaitu operator dasar yang fungsinya unik dan operator tambahan yang merupakan turunan dari satu atau lebih dari operator dasar dan mempunyai fungsi utama untuk menyederhanakan suatu ekspresi yang kompleks.

Operator Dasar , terdiri dari :

a. Selection
b. Projection
c. Cartesian Product
d. Union
e. Difference dan
f. Renama

Operator Tambahan, terdiri dari :

a. Intersection
b. Division
c. Natural Join

Semua operator diatas dapat diekspresikan ke dalam satu atau lebih tabel dan hasilnya berupa table.

Berikut akan dijelaskan contoh kasus dengan menggunakan tiga statement

dan 3 tabel / relasi, yaitu :

Tiga Statement tersebut antara lain sebagai berikut :

a. Natural Language (NL) merupakan bahasa ilmiah/sehari-hari yang digunakan oleh manusia dalam berkomunikasi dengan sistem
b. Aljabar Relasional (AR) merupakan contoh penggunaan sintaks aljabar relasional dalam menjawab / menyelesaikan permintaan dari manusia (user) diatas
c. Tabel merupakan bentuk tampilan jawaban aljabar relasional dalam bentuk tabel/relasi dua dimensi yang terdiri dari kolom dan baris.

Berikut adalah table yang akan digunakan dalam implementasi aljabar relasional :

Tabel Pelanggan

22870	Agus	Cimone	Tangerang
48616	Jojo	Merdeka	Tangerang
03214	Fatah	Cikokol	Tangerang
37662	Jumali	Cimone	Tangerang

Tabel Rekening

012.145.002	Checking	8.000.000
771.225.421	Checking	3.000.000
315.222.310	Saving	4.000.000
342.256.010	Checking	6.000.000
511.333.279	Saving	1.000.000
122.003.007	Saving	8.500.000

Tabel Punya

a. **Selection**

22870	342.256.010
22870	511.333.279
48616	771.225.421
48616	315.222.310
03214	012.145.002
37662	122.003.007

Operator ini menggunakan simbol σ (low-case omega). Operasi Selection menyeleksi tuple-tuple (record) pada sebuah relasi, yaitu tuple-tuple (record) yang memenuhi predicate syarat yang sudah ditentukan sebelumnya.
Selection operation disebut UNARY OPERATION karena beroperasi pada sebuah relasi. Komparasi dapat dilakukan dengan menggunakan : =, ≠ atau <>, <, ≤, >, ≥ pada predikat seleksi dan untuk menggabungkan serangkaian predikat adalah dengan menggunakan **and** (∩) atau **or** (V)

Sintaks :

σ [karakteristik/kondisi] (tabel)

Contoh :

1. NL : Cari semua tuple/record pada rekening yang saldonya lebih dari 4000000. Bila diterjemahkan ke dalam aljabar relasional menjadi : **AR :**

   σ Saldo ≥ 4000000 (Rekening)

012.145.002	Checking	8.000.000
342.256.010	Checking	6.000.000
122.003.007	Saving	8.500.000

Table yang dihasilkan :

2. NL : Cari semua tuple/record pada rekening dengan tipe saving dan Saldonya lebih dari 4000000. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : σ Tipe = "Saving" ∧ Saldo > 4000000 (Rekening)

122.003.007	Saving	8.500.000

Table yang dihasilkan :

3. NL : Cari semua pelanggan yang tinggal di jalan Cimone. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : σ Jalan = "Cimone" (Pelanggan)

Agus	22870	Cimone	Tangerang
Jumali	37552	Cimone	Tangerang

Table yang dihasilkan :

b. **Projection**

Projection / Project (π), adalah operasi untuk memperoleh kolom = kolom tertentu. Operasi project adalah operasi unary yang mengirim relasi argumen dengan kolom = kolom tertentu. Karena relasi adalah himpunan, maka baris = baris duplikasi dihilangkan.

Sintaks yang digunakan dalam operasi proyeksi ini adalah sebagai berikut :

π colum1,....,column (tabel)

Contoh :

1. NL : Tampilkan semua nomor rekening dan saldo dari tabel rekening. Bila diterjemahkan ke dalam aljabar relasional menjadi : AR : ∧ No_Rekening,Saldo (Rekening)

012.145.002	8.000.000
771.225.421	3.000.000
315.222.310	4.000.000
342.256.010	6.000.000
511.333.279	1.000.000
122.003.007	8.500.000

Table yang dihasilkan :

2. NL : Tampilkan semua no. rekening dan saldo dari semua rekening yang saldonya lebih dari 4000000. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : ∏ No_Rekening, Saldo (σ Saldo ≥ 4000000 (Rekening))

012.145.002	8.000.000
342.256.010	6.000.000
122.003.007	8.500.000

Table yang dihasilkan :

3. NL : Tampilkan semua nomor rekening dan tipe dari semua rekening yang saldonya lebih besar dari 3000000 atau saldo lebih besar dari 5000000. Bila diterjemahkan ke dalam aljabar relasional menjadi : AR : ∏ No_Rekening, Tipe (σ Saldo ≥ 3000000 V Saldo ≥ 5000000 (Rekening))

Table yang dihasilkan :

012.145.002	Checking
315.222.310	Saving
342.256.010	Checking
122.003.007	Saving

c. Cartesian Product

Operator ini menggunakan simbol ⊗. Operator ini merupakan binary operation, yaitu operator yang beroperasi pada dua relasi. Komparasi dapat dilakukan dengan menggunakan : ≡, ≠ atau ≪, ≤, ≦, ≥, ≧ pada predikat seleksi dan untuk menggabungkan serangkaian predikat adalah dengan menggunakan and (∧) atau or (∨). R1 dan R2 adalah relasi Sintaks :

R1 X R2

Contoh :

Bila R1 mempunyai arity K1 dan R2 mempunyai arity K2, maka R1 X R2 adalah himpunan tuple yang jumlah komponen pertamanya diambil dari tuple

R1 dan K2 komponen terakhirnya diambil dari tuple R2, sehingga arity dari relasi hasil Cartesian Product adalah K1 + K2

Hasil operasi Cartesian product :

Contoh :

1. NL : Cari nomor kartu Pelanggan yang memiliki rekening ≡ 511.333.279. Bila diterjemahkan ke dalam aljabar relasional menjadi : AR : ∏ No_Kartu (σ No_Rekening ≡ 511.333.279 ∏ Pelanggan.No_Kartu ≡ Punya.No_Kartu (Pelanggan X Punya))

Langkah 1 :

22870	Agus	Cimone	Tangerang
48616	Joja	Merdeka	Tangerang
03214	Fatah	Cikokol	Tangerang
37662	Jumati	Cimone	Tangerang

Tabel Pelanggan

22870	342.256.018
22870	511.333.279
48616	771.225.421
48616	315.222.310
03214	012.145.002
37662	122.003.087

Tabel Punya

Langkah 2 :

22870	Agus	Cimone	Tangerang	22870	342.256.016
22870	Agus	Cimone	Tangerang	22870	511.333.279
22870	Agus	Cimone	Tangerang	48616	771.225.421
22870	Agus	Cimone	Tangerang	48616	315.222.310
22870	Agus	Cimone	Tangerang	03214	012.145.002
22870	Agus	Cimone	Tangerang	37662	122.003.007
22870	Agus	Cimone	Tangerang	22870	342.256.016
48616	Jojo	Merdeka	Tangerang	22870	511.333.279
48616	Jojo	Merdeka	Tangerang	48616	771.225.421
48616	Jojo	Merdeka	Tangerang	48616	315.222.310
48616	Jojo	Merdeka	Tangerang	03214	012.145.002
48616	Jojo	Merdeka	Tangerang	37662	122.003.007
48616	Jojo	Merdeka	Tangerang	22870	342.256.016
03214	Fatah	Cikokol	Tangerang	22870	511.333.279
03214	Fatah	Cikokol	Tangerang	48616	771.225.421
03214	Fatah	Cikokol	Tangerang	48616	315.222.310
03214	Fatah	Cikokol	Tangerang	03214	012.145.002
03214	Fatah	Cikokol	Tangerang	37662	122.003.007
03214	Fatah	Cikokol	Tangerang	22870	342.256.016
37662	Jumait	Cimone	Tangerang	22870	511.333.279
37662	Jumait	Cimone	Tangerang	48616	771.225.421
37662	Jumait	Cimone	Tangerang	48616	315.222.310
37662	Jumait	Cimone	Tangerang	03214	012.145.002
37662	Jumait	Cimone	Tangerang	37662	122.003.007

22870	Agus	Cimone	Tangerang	22870	511.333.279
48616	Jojo	Merdeka	Tangerang	22870	511.333.279
03214	Fatah	Cikokol	Tangerang	22870	511.333.279
37662	Jumait	Cimone	Tangerang	22870	511.333.279

22870
48616
03214
37662

Langkah 3 : hasil Akhir

d. Union

Operator ini menggunakan simbol ∪. Operator ini merupakan binary operation, yaitu operator yang beroperasi pada dua relasi.

Komparasi dapat dilakukan dengan menggunakan : ≡, ≠ atau ≪, ≤, ≦, ≫, ≧ pada predikat seleksi dan untuk menggabungkan serangkaian predikat adalah

dengan menggunakan and (∩) atau or (∨)

Sintaks :

$$R = R1 \cup R2$$

Hasil operasi Union

R1

a	b	c
d	a	f
c	b	d

R2

b	a	d
d	a	f

R = R1 ∪ R2

a	b	c
d	a	f
c	b	d
b	a	d

Contoh :

Tabel / Relasi DEPOSIT

Downtown	101	Johnson	500
Mianus	215	Smith	700
Perryridge	102	Hayes	400
Round Hill	305	Turner	350
Perryridge	201	Williams	900

Tabel / Relasi BORROW

Downtown	17	Jones	1000
Redwood	23	Smith	2000
Perryridge	15	Hayes	1500
Pownal	29	Williams	1200
North Town	16	Adams	1300
Perryridge	26	Glenn	2500
Brighton	10	Brooks	2200

1. NL : Mencari nama Customer dari cabang Perryridge yang memiliki account atau loan atau kedua-duanya. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : ∏ Customer_Name (σ Branch_Name = "Perryridge" (BORROW))

∩

∏ Customer_Name (σ Branch_Name = "Perryridge" (DEPOSIT))

Table yang dihasilkan :

Hayes
Williams
Glenn

e. **Difference**

Operator ini menggunakan simbol − : Operator ini merupakan binary operation, yaitu operator yang beroperasi pada dua relasi. Operator ini berfungsi untuk mengeliminasi entity / record dari suatu tabel yang ada pada tabel lain, dan kedua tabel harus memiliki atribut yang sama.

Sintaks :

[tabel 1] − [tabel 2] atau R = R1 − R2

Contoh :

1. NL : Mencari semua Pelanggan yang saldonya diatas 4000000 dan tidak bersifat saving. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : ∏ No_Kartu (σ tipe ≡ "cheking" (Rekening x Pelanggan))

−

∏ No_Kartu (σ tipe ≡ "saving" (Rekening x Pelanggan))

012.145.002	Checking	8.000.000
342.256.010	Checking	6.000.000

03214	Fatah
22870	Agus

Table yang dihasilkan :

f. **Rename**

Operator ini menggunakan simbol ρ: Operator ini berfungsi menyalin tabel

lama menjadi

tabel dengan nama baru.

Sintaks :

$$\rho\ [\ \text{New Name}\ ]\ [\text{Old Name}]$$

Keterangan :

New Name = nama relasi baru

Old Name = nama relasi lama

Contoh :

1. NL : Salin tabel Rekening dengan nama X. Bila diterjemahkan ke dalam aljabar relasional menjadi :

AR : □ X (Rekening)

Table yang dihasilkan :

Tabel X

012.145.002	Checking	8.000.000
771.225.421	Checking	3.000.000
315.222.310	Saving	4.000.000
342.256.010	Checking	6.000.000
511.333.279	Saving	1.000.000
122.003.007	Saving	8.500.000

g. Intersection

Operator ini menggunakan simbol ∩. Operator ini termasuk kedalam kategori operator tambahan, karena operator ini dapat diderivasi dari operator dasar.

Sintaks :

$$R = R1 \cap R2$$

342.256.010	Checking	6.000.000
511.333.279	Saving	1.000.000
122.003.007	Saving	8.500.000

g. Intersection

Operator ini menggunakan simbol ∩. Operator ini termasuk kedalam kategori operator tambahan, karena operator ini dapat diderivasi dari operator dasar.

Sintaks : $R = R1 \cap R2$

Hasil operasi intersection :

Contoh :

1. **NL : Mencari nomor kartu yang memiliki rekening bertipe saving. Bila diterjemahkan ke dalam aljabar relasional menjadi :**

AR : ∏ No_Kartu (σ tipe = "saving" ∏ Rekening.No_Rekening = Punya.No_Rekening (Rekening x Punya)) ∏ ∏ No_Kartu (σ tipe = "saving" ∏ Rekening.No_Rekening = Punya.No_Rekening (Rekening x Punya))

Table yang dihasilkan :

Tabel Rekening

012.145.002	Checking	8.000.000
771.225.421	Checking	3.000.000
315.222.310	Saving	4.000.000
342.256.010	Checking	6.000.000
511.333.279	Saving	1.000.000
122.003.007	Saving	8.500.000

Tabel Punya

22870	342.256.010
22870	511.333.279
48616	771.225.421
48616	315.222.310
03214	012.145.002
37662	122.003.007

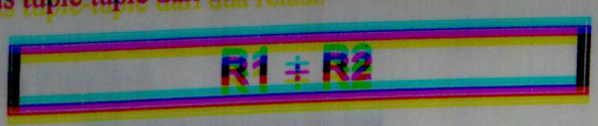
Table yang dihasilkan :

22870
48616
37662

h. **Division**

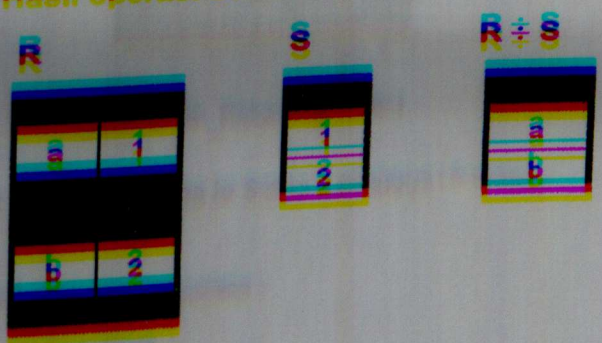
Operator ini menggunakan simbol ÷. Operator ini merupakan operasi pembagian atas tuple-tuple dari dua relasi.

Sintaks :



Contoh :

Hasil operasi Division



1. NL : Mencari No_Kartu yang mempunyai No_Rekening dengan jumlah saldo 8500000. Bila diterjemahkan ke dalam aljabar relasional menjadi :

   AR : R1 ≡ ÷ No_Kartu, No_Rekening (Punya)

   Tabel yang dihasilkan :

Tabel Punya

22870	342.256.010
22870	511.333.279
48616	771.225.421
48616	315.222.310
03214	012.145.002
37662	122.003.007

AR : R2 = □ No_Rekening (σ Saldo = 8500000 (Rekening))

Tabel yang dihasilkan :

37662

AR : R1 = π No_Kartu, No_Rekening (Punya)

÷

R2 = π No_Rekening (σ Saldo = 8500000 (Rekening))

Tabel Akhir yang dihasilkan :

37662

i. Natural Join

Operator ini dikembangkan dari operator Cartesian Product (x). Simbol yang digunakan

□:

Sintaks : R1 θ R2 atau R θ S

Keterangan :

1) Relation R1 dan R2 mempunyai attribute yang sama
2) Membentuk suatu Cartesian Product dari argumen-argumennya

3) Melakukan operasi selection yang menekankan persamaan values pada attribute yang sama, yang berada pada kedua relasi
4) Menghilangkan duplicate column

R adalah relasi dengan elemen E1, E2,:::: En
S adalah relasi dengan elemen T1, T2,:::: Tn
Maka Operasi Natural Join (□) adalah K = R □ S, dan akan menghasilkan

Hasil operasi Natural Join

BAB XI

STRUCTURE QUERY LANGUAGE (SQL)

1. Sejarah SQL

SQL pertama diterapkan oleh System R IBM, pada tahun 1970 an. SQL adalah standard query language untuk membuat dan memanipulasi pada Relational Databases. Beberapa perbedaan kecil pada syntax, tetapi mayoritas SQL adalah standar misal pada MS Access, Oracle, Sybase, Informix, etc. SQL adalah suatu alat Perintah Baris atau dapat juga ditempelkan pada bahasa pemrograman seperti: Cobol, "C", Pascal, etc

SQL adalah Bahasa distandarisasi yang dimonitor oleh American National Standards Institute (ANSI) sama halnya oleh National Institute of Standards (NIST).

a. ANSI 1990 = SQL 1 standard
b. ANSI 1992 = SQL 2 Standard (sometimes called SQL-92)
c. SQL 3 is in the works = adds some Object oriented concepts

Tipe –tipe data yang digunakan dalam sql, antara lain :

a. char(n). character string dengan panjang tetap, dengan spesifikasi panjang *n*.
b. varchar(n). character string dengan panjang bervariasi, dengan spesifikasi panjang maksimum *n*.
c. int. Integer (a finite subset of the integers that is machine-dependent).
d. smallint. Small integer.
e. numeric(p,d). Angka dengan panjang tetap, dengan pendekatan spesifikasinya adalah *p* digit, dengan *n* digit kekanan nilai desimal.
f. real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.
g. float(n). Angka pecahan, dengan pendekatan spesifikasiya paling tidak *n* digit.

SQL dapat dibagi kedalam dua bentuk, yaitu :

A. Data Definition Language (DDL)

User dapat membuat tabel baru, mengindeks, mengubah tabel, menentukan struktur penyimpanan tabel, dan sebagainya.

- Data Definition Language (DDL) terdiri dari :
  - = CREATE TABLE
  - = DROP TABLE
  - = ALTER TABLE

1. CREATE TABLE

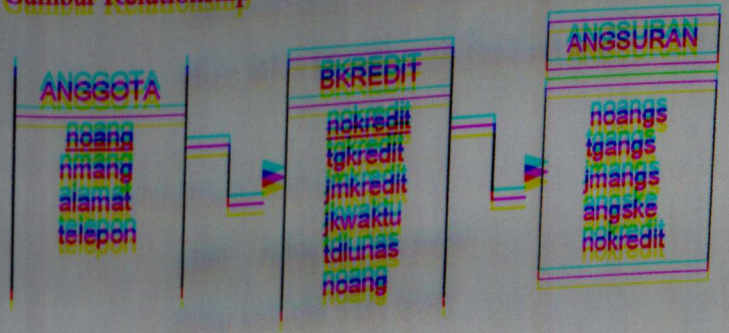
Bentuk umum create table

```
create table table-name
(column = definition
[,column-definition]...
[,primary-key-definition]
[,foreign-key-definition
[,foreign-key-definition]...])
```

untuk 'column-definition' mempunyai bentuk:

column-name data-type [not null]

- Gambar Relationship



a) Membuat tabel anggota

b) membuat tabel bkredit

```
create table anggota
(noang char(2) not null,
 nmang char(20),
 alamat char(20),
 telepon char(15),
 primary key (noang));
```

```
create table bkredit
  (nokredit char(3) not null,
   tgkrdit  date,
   jmkredit number,
   jkwaktu  number,
   tdlunas  char(5),
   noang char(2),
   primary key (nokredit)
  foreign key noang
References anggota  );
```

c) Membuat tabel angsuran

```
create table angsuran
  (noangs char(3) not null,
   tgangs date,
   jmangs number,
   angske  number,
   nokredit char(3),
   primary key(noangs)
  foreign key nokredit
  references bkredit );
```

2. ALTER TABLE

Bentuk umum **alter table**

a. Menambah atribut

```
alter  table  table-name
add column data type;
```

contoh:

menambahkan atribut discount pada relasi bkredit

```
alter table bkredit add discount int;
```

b. Menghapus atribut

```
alter  table  table-name
drop column data type;
```

contoh:

menghapus atribut discount pada relasi bkredit

```
alter table bkredit drop discount int;
```

3. DROP TABLE

a. Menghapus tabel

Bentuk Umum drop table drop table table-name;

Contoh:

menghapus tabel anggota

Drop table anggota;

iii. Data Manipulaton Language (DML)

Bentuk SQL yang berguna untuk memanipulasi (insert, delete, update) dan pengambilan data pada suatu basis data. Tujuan DML memudahkan user mengakses data seperti yang direpresentasikan model data.

1. Data Manipulation Language (DML) terdiri dari :

a. SELECT

Menampilkan sebagian atau seluruh isi dari suatu table.
Menampilkan kombinasi dari beberapa table.

b. UPDATE

Mengubah isi satu atau beberapa atribut dari suatu table.

c. DELETE

Menghapus sebagian atau seluruh isi dari suatu table.

d. INSERT

Menambah satu atau beberapa baris nilai baru kedalam suatu table.

ANGGOTA

NOANG	NMANG	ALAMAT	TELEPON
A1	Ajie	Selindung	021-585375
A2	Andi	Pangkalpinang	021-123456
A3	Ani	Sungailiat	021-654321
A4	Ana	Pangkalpinang	021-585370
A5	Agus	Sungailiat	021-212121

A6	Angga	Koba	
A7	Ade	Selindung	021-585310

- BKREDIT

NOKREDIT	TGKREDIT	JMKREDIT	JKWAKTU	TDLUNAS	NOANG
B01	10-01-2006	1.000.000	2	LUNAS	A1
B02	15-01-2006	1.200.000	6		A2
B03	20-02-2006	1.500.000	2	LUNAS	A5
B04	25-08-2006	1.000.000	2	LUNAS	A1
B05	20-08-2006	2.000.000	5		A5

- ANGSURAN

NOANGS	TGANGS	JMANGS	ANGSKE	NOKREDIT
T01	01-02-2006	500.000	1	B01
T02	01-02-2006	200.000	1	B02
T03	01-03-2006	750.000	1	B03
T04	01-03-2006	500.000	2	B01
T05	01-03-2006	200.000	2	B02
T06	01-04-2006	750.000	2	B03
T07	01-05-2006	200.000	3	B02
T08	01-06-2006	200.000	4	B02
T09	01-09-2006	500.000	1	B04
T10	01-09-2006	1.000.000	1	B05
T11	01-10-2006	500.000	2	B04

a) SELECT :

Bentuk umum perintah select

```
select [distinct] field(s) from table(s)
[where predicate]
[group by field(s) [having predicate]]
[order by field(s)];
```

Catatan:

Predicate kondisi yang bisa diberikan

- Retrieval Sederhana

: Tampilkan nomor anggota yang pernah mengajukan kredit

```
select noang from bkredit;
```

Dari hasil query di atas kemungkinan mendapatkan duplikasi data, untuk mendapatkan data tuggal maka query diatas menjadi:

```
select distinct noang from bkredit;
```

: Tampilkan semua informasi mengenai seluruh anggota.

```
select * from anggota;
```

- Retrieval dengan komputasi sederhana
  - Tampilkan nomor anggota dan jumlah pinjaman dari semua kredit dalam dolar (uang dalam tabel bkredit menggunakan satuan rupiah).

```
select noang, '$', jmkredit / 10000 from bkredit;
```

- Retrieval dengan Kondisi
  - Tampilkan nama-nama anggota untuk yang beralamat di Selindung

```
select nmang from anggota
    where alamat = 'selindung';
```

- Retrieval dengan pengurutan
  - Tampilkan nama-nama anggota untuk yang beralamat di Selindung diurutkan secara descending

```
select nmang from anggota
    where alamat = 'selindung'
    order by nmang desc;
```

- SQL termasuk between sebagai operator pembanding
- Contoh:
  - Tampilkan nomor angsuran dari relasi angsuran dengan jumlah angsuran antara 500,000 dan 1.000.000.

```
select noangs from angsuran
where jmangs between 500000 and 1000000 ;
```

- Query dengan melibatkan lebih dari satu tabel
  - Simple Equijoin
    - Tampilkan semua kombinasi anggota dan bkredit untuk anggota yang melakukan kredit

```
select anggota.*, bkredit.*
from anggota, bkredit
where anggota.noang = bkredit.noang;
```

  - Join antara tiga table
    - Tampilkan nama anggota dan nomor kredit yang mempunyai nomor angsuran 'T05'

```
select distinct a.nmang, b.nokredit from
anggota a, bkredit b, angsuran c
where c.noangs='T05' and c.nokredit=b.nokredit and
b.noang=a.noang ;
```

Function Pada Sql ada 5 macam, yaitu :

- COUNT

Banyaknya nilai-nilai pada satu kolom.

- **SUM**

  Jumlah nilai dari satu kolom.

- **AVG**

  Rata-rata nilai dari satu kolom.

- **MAX**

  Nilai terbesar yang ada pada satu kolom.

- **MIN**

  Nilai terkecil yang ada pada satu kolom.

Contoh :

-Tampilkan banyaknya anggota yang ada select count(*) from anggota

-Tampilkan banyaknya anggota yang mengajukan kredit

select count (distinct noang) from bkredit;

-Tampilkan jumlah kuantitas kredit yang diajukan oleh anggota A1

select sum(jmkredit) from bkredit where noang='A1';

b) UPDATE

Bentuk Umum UPDATE

update table
set field = expression [,
field = expression]...
[where predicate];

Contoh :

- Isi nomor telepon dengan '021-1111111' untuk nomor anggota 'A6'

```
update anggota
    set telepon = '021-1111111'
    where noang= 'A6';
```

- Beri tanda 'Lunas' untuk bukti kredit yang belum lunas

```
update bkredit
```

```
set      tdlunas='Lunas'
where tdlunas is null;
```

C. DELETE

Bentuk Umum DELETE DELETE FROM table
[WHERE predicate];

Contoh :

- Single = record delete

```
delete from anggota
where noang = 'A5';
```

- Multiple record delete

```
delete from anggota
where alamat ='Ciledug';
```

- Multiple record delete

```
delete from anggota;
```

c) INSERT

Bentuk Umum INSERT

```
insert
into table [ (field [,field]...)]
values (constant [,constant]...);
        Atau
insert into table [ (field [,field]...)] select
... from ... where ...;
```

Contoh :

- Single = record Insert

```
insert into anggota (noang, nmang, alamat)
values ('A8', 'Budi', 'Jakarta Barat');
```

- Single = record Insert dengan nama-nama field diabaikan

```
insert into anggota
values ('A9', 'Bodang', Banten, '021-22222');
```

d) Retrieval dengan LIKE

Contoh :

= Tampilkan semua Anggota yang namanya dimulai dengan huruf "A"

```
select anggota.* from anggota where
anggota.nmang like 'A%';
```

= Tampilkan semua Anggota yang namanya mengandung huruf-huruf "n"

```
select * from anggota
where nmang like '%n%';
```

e) Retrieval melibatkan NULL

Contoh :

= Tampilkan nomor anggota untuk anggota dengan telepon kosong

```
select noang from anggota
where telepon is null;
```

= Tampilkan nomor anggota untuk anggota dengan telepon tidak kosong

```
select noang from anggota
where telepon is not null;
```

= Kondisi nilai 'NULL' tidak boleh dicek dengan =, >, >=, <= atau <>

Contoh :

```
select noang from anggota where telepon=null;
```

BAB XII

MYSQL

1. MYSQL

Mysql adalah sebuah sistem manajemen database relasi (relational database management system) yang bersifat "terbuka" (open source).
Mysql memiliki kinerja, kecepatan proses, dan ketangguhan yang tidak kalah dibanding database-database besar lainnya yang komersil seperti ORACLE, Sybase, Unify dan sebagainya.
Mysql yang akan digunakan adalah mysql-4.0.13-win dengan MySQL-Front_2.5 sebagai interface

Ada beberapa pertimbangan mengapa memilih MySQL

(1) **Kecepatan**.

Berdasarkan hasil pengujian, MySQL memiliki kecepatan paling baik dibanding database server lainnya. (www.mysql.com/information/benchmarks.html)

(2) **Mudah digunakan**

Perintah, aturan dan proses instalasi relatif mudah.

(3) **Open Source**

Siapapun dapat berpartisipasi untuk mengembangkan MySQL dan hasil pengembangan itu diserahkan kepada umum atau kepada komunitas Open Source

(4) **Kapabilitas**

MySQL telah digunakan untuk mengelola database dengan jumlah 50 juta record, 60.000 tabel dengan jumlah baris 5.000.000.000 dan mendukung penggunaan index hingga 32 buah index per-tabelnya.

(5) **Biaya rendah (relatif gratis)**

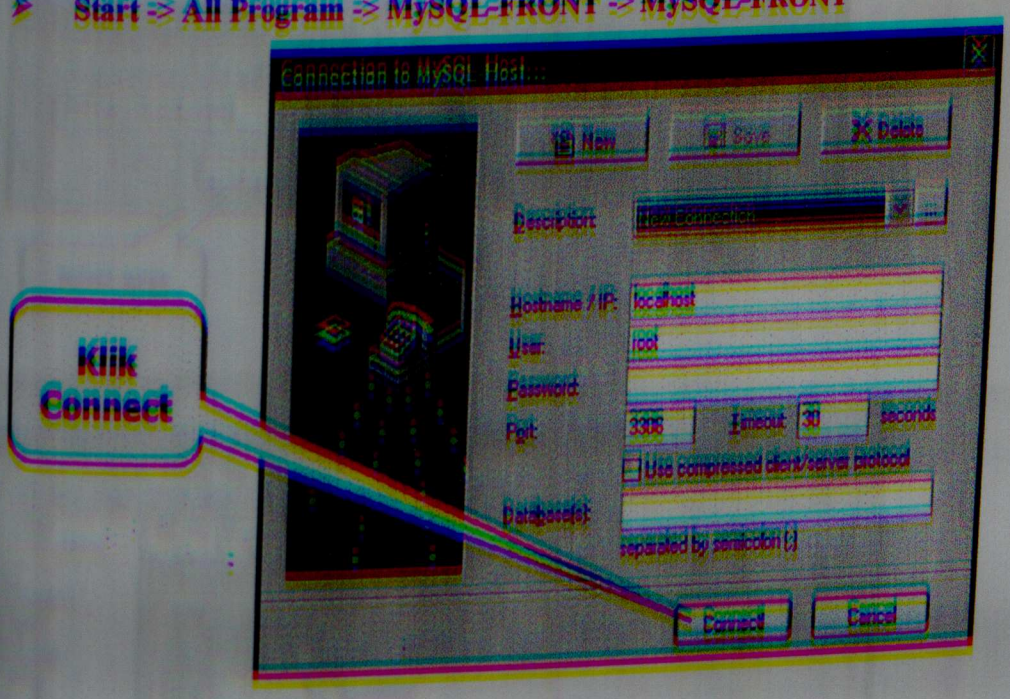
(6) Konektifitas dan keamanan

MySQL mendukung dan menerapkan sistem keamanan dan izin akses tingkat lanjut (advanced pemissions and security system)

(7) Lintas platform sistem operasi

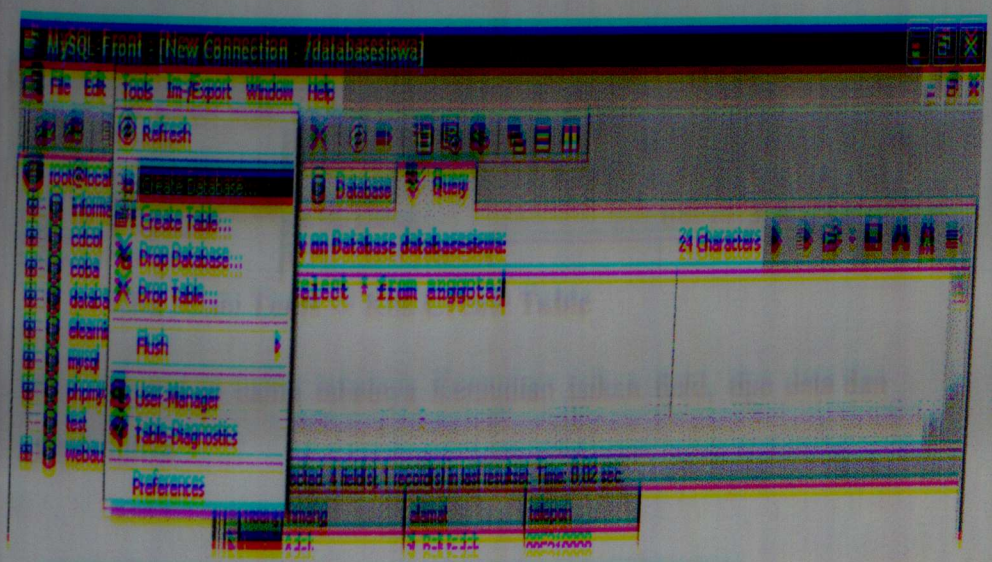
MySQL dapat dijalankan dibeberapa sistem operasi yang berbeda, seperti Linux, Microsoft Windows, FreeBSD, Sun Solaris, IBM's AIX, Mac OS X, HP-UX, AIX, QNX, Novell Netware, SCO OpenUnix, SGI Irix dan Dec OSF.

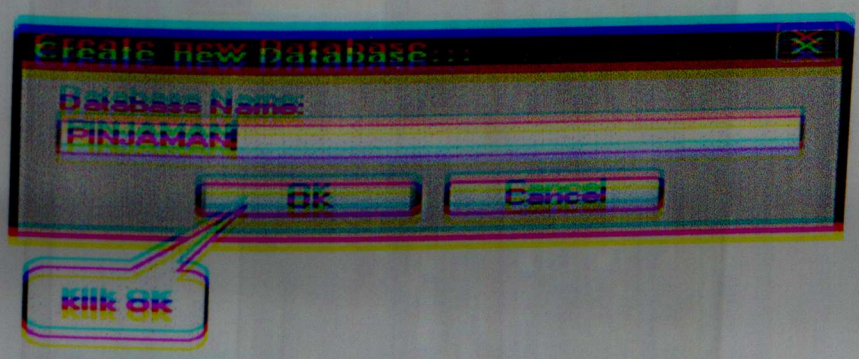
- Menjalankan MySQL Front :
  - Start -> All Program -> MySQL-FRONT -> MySQL-FRONT

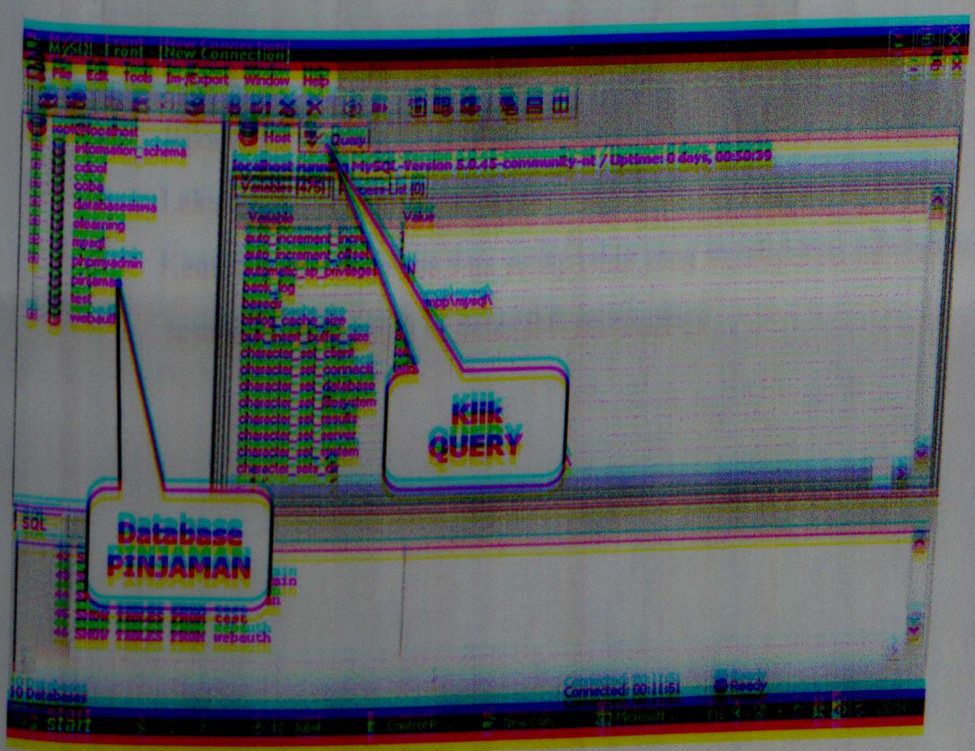


- BUAT DATABASE DENGAN NAMA : **PINJAMAN**

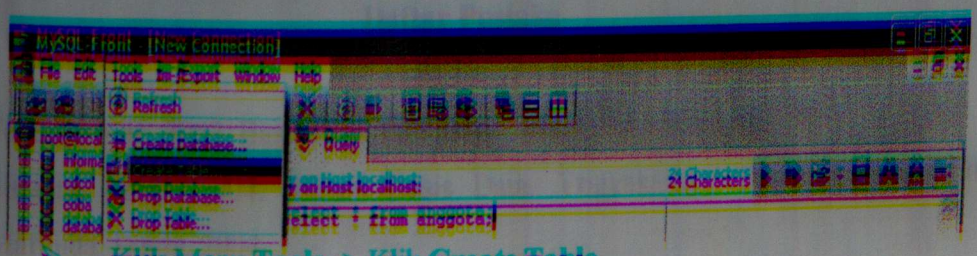
➢ Klik Menu Tools => Klik Create Database



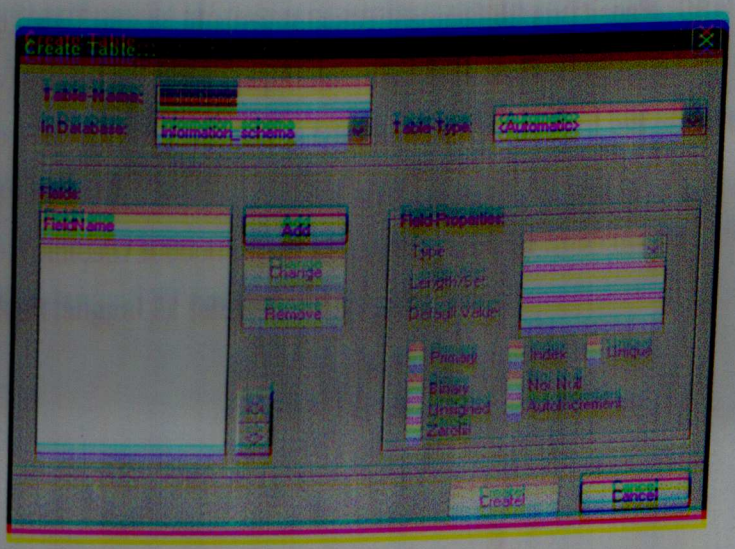




BUAT TABEL



- Klik Menu Tools -> Klik Create Table
- Tentukan nama tabelnya Kemudian isikan field, tipe data dan panjangnya



- Isikan data pada table tersebut
- Lakukan untuk table selanjutnya
- Kemudian isikan semua data setiap table yang sudah dibuat sebelumnya sesuai dengan contoh isi data table sebelumnya.

Daftar Pustaka

1. Fatansyah. Ir, 2002. Basis Data.Bandung : Informatika Bandung.
2. Sutanta, Edhy, 2004. Sistem Basis Data. Yogyakarta : Penerbit Graha Ilmu. Marlinda, Linda, 2004.
3. Sistem Basis Data. Yogyakarta : Penerbit Andi.
4. Kusrini. 2006. Strategi Perancangan dan Pengelolaan Basis Data. Yogyakarta : Penerbit Andi.
5. Kadir, Abdul, 2002. Penuntun Praktis Belajar SQL. Yogyakarta:Penerbit Andi
6. http://staff.uny.ac.id/sites/default/files/pendidikan/Diana%20Rahmawati,%20M.Si./MODEL%20DATA.pdf diakses tgl 21 februari 2013
7. http://datatik.files.wordpress.com/2010/01/materi-erd.pdf ida ayu diakses tanggal 21 februari 2013
8. http://ramos672006005.files.wordpress.com/2011/06/encapsulation- inheritance.pdf diakses tanggal 22 februari 2013