KONSEP DASAR

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

SISTEM DATABASE adalah suatu sistem penyusunandan pengelolaan record-record dengan menggunakan komputer, dengan tujuan untuk menyimpan atau merekam serta memelihara data operasional lengkap sebuah organisasi/perusahaan sehingga mampu menyediakan informasi yang diperlukan pemakai untuk **kepentingan proses pengambilan keputusan.**

KOMPONEN DASAR DARI SISTEM DATABASE

Terdapat 4 komponen pokok dari system database:

A. DATA, dengan ciri-ciri :

1. Data disimpan secara terintegrasi (*Integrated*) Terintegrated yaitu Database merupakan kumpulan dari berbagai macam file dari aplikasi-aplikasi yang berbeda yang disusun dengan cara menghilangkan bagian-bagian yang rangkap (*redundant*)
2. Data dapat dipakai secara bersama-sama(*shared*) Shared yaitu Masing-masing bagian dari database dapat diakses oleh pemakai dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

B. Perangkat Keras (HARDWARE)

Terdiri dari semua peralatan perangkat keras computer yang digunakan untuk pengelolaan sistem database berupa :

1. Peralatan untuk penyimpanan misalnya disk, drum, tape
2. Peralatan input dan output
3. Peralatan komunikasi data, dll

C. Perangkat Lunak (SOFTWARE)

Berfungsi sebagai perantara (*interface*) antara pemakai dengan data phisik pada database, dapat berupa :

1. Database Management System (DBMS)
2. Program-program aplikasi & prosedur-prosedur

D. Pemakai (USER)

Terbagi menjadi 3 klasifikasi :

1. Database Administrator (DBA), orang/tim yang bertugas mengelola system database secara keseluruhan
2. Programmer, orang/tim membuat program aplikasi yang mengakses database dengan menggunakan bahasa pemprograman
3. End user, orang yang mengakases database melalui terminal dengan menggunakan query language atau program aplikasi yang dibuat oleh programmer

DATA PADA DATABASE DAN HUBUNGANNYA

Ada 3 jenis data pada sistem database, yaitu:

1. Data operasional dari suatu organisasi, berupa data yang disimpan didalam database
2. Data masukan (input data), data dari luar system yang dimasukan melalui peralatan input (keyboard) yang dapat merubah data operasional
3. Data keluaran (output data), berupa laporan melalui peralatan output sebagai hasil dari dalam system yang mengakses data operasional

CONTOH PENGGUNAAN DATABASE

- Pembelian barang di supermarket, kasir akan melakukan scan barcode yang ada di barang, pada saat tersebut program akan mengakses data pada database kemudian mengurangi stok barang yang ada sesuai dengan jumlah pembelian konsumen
- Pembelian barang dengan menggunakan kartu kredit, card reader akan membaca apakah kartu kredit tersebut memiliki limit yang cukup, dan memasukkan data pembelian dalam database kartu kredit, juga ada pemeriksaan apakah kartu tersebut tidak dalam daftar kartu di curi/hilang

KEUNTUNGAN PEMAKAIAN SISTEM DATABASE

1. Terkontrolnya kerangkapan data dan inkonsistensi
2. Terpeliharanya keselarasan data
3. Data dapat dipakai secara bersama-sama
4. Memudahkan penerapan standarisasi
5. Memudahkan penerapan batasan-batasanpengamanan.
6. Terpeliharanya intergritas data
7. Terpeliharanya keseimbangan atas perbedaan kebutuhan data dari setiap aplikasi
8. Program / data independent

KERUGIAN PEMAKAIAN SISTEM DATABASE

1. Mahal dalam implementasinya
2. Rumit/komplek
3. Penanganan proses recovery & backup sulit
4. Kerusakan pada sistem basis data dapat mempengaruhi departemen yang terkait

ISTILAH-ISTILAH YG DIPERGUNAKAN DALAM SISTEM BASIS DATA

a. Enterprise yaitu suatu bentuk organisasi

   Contoh :

   Enterprise: Sekolah → Database Nilai

   Enterprise: Rumah sakit → Database AdministrasiPasien

b. Entitas yaitu suatu obyek yang dapat dibedakan dengan objek lainnya

   Contoh :

   Database Nilai → entitas: mahasiswa, Matapelajaran

   Database Administrasi Pasien entitas: pasien, dokter, obat.

c. Atribute/field yaitu setiap entitas mempunyai atribut atau suatu sebutan untuk mewakili suatu entitas.

   Contoh :

   Entity siswa → field = Nim, nama_siswa,alamat,dll

   Entity nasabah → field=Kd_nasabah,nama_nasabah,dll

d. Data value yaitu data aktual atau informasi yang disimpan pada tiap data elemen atau atribute.

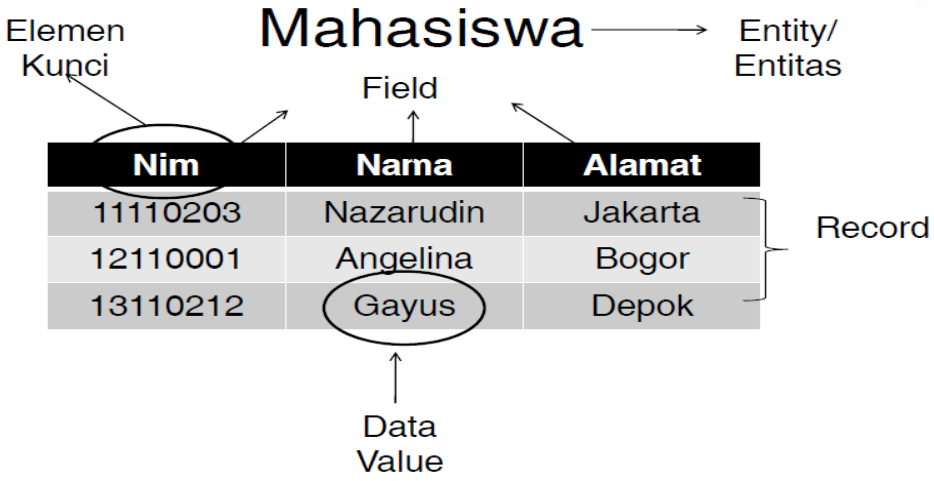
   Contoh : Atribut nama_karyawan → sutrisno, budiman, dll

e. Record/tuple yaitu kumpulan elemen-elemen yang salingberkaitan menginformasikan tentang suatu entity secara lengkap.

   Contoh : record mahasiswa → nim, nm_mhs, alamat.

f. File yaitu kumpulan record-record sejenis yang mempunyai panjang elemen sama, atribute yang sama namun berbeda-beda data valuenya

g. Kunci elemen data yaitu tanda pengenal yang secara unik mengindentifikasikan entitas dari suatu kumpulan entitas.



TUJUAN PERANCANGAN DATABASE:

1. Untuk memenuhi informasi yang berisi kebutuhan–kebutuhan user secara khusus dan aplikasinya.
2. Memudahkan pengertian struktur informasi

3. Mendukung kebutuhan–kebutuhan pemrosesan dan beberapa objek penampilan (*respone time, processing time* dan *strorage space*).

APLIKASI DATABASE DALAM LIFE CYCLE

1. Pendefinisian Sistem (*System definition*) Pendefinisian ruang lingkup dari sistem database, pengguna dan aplikasinya.
2. Perancangan Database (*Database Design*) Perancangan database secara logika dan fisik pada suatu sistem database sesuai dengan sistem manajemen database yang diinginkan.
3. Implementation Pendefinisian database secara konseptual, eksternal dan internal, pembuatan file–file database yang kosong serta implementasi aplikasi software.
4. Pengambilan dan Konversi Data (*Loading atau data convertion*) Database ditempatkan dengan baik, sehingga jika ingin memanggil data secara langsung ataupun merubah file–file yang ada dapat di tempatkan kembali sesuai dengan format sistem databasenya
5. Konversi Aplikasi (*Aplication conversion*) Pengkonversian aplikasi agar dapat berjalan dengan database baru.
6. Pengujian dan Validasi (*Testing dan Validation*) Pengujian dengan menjalankan database dengan memberikan data-data "real" untuk menemukan kesalahan yang mungkin terjadi.
7. Monitoring dan Maintenance Montoring adalah proses pemantau performa dari database, jika performa database menurun maka dapat dilakukan proses *tuning* dan *reorganized* Maintenance adalah proses manajemen database selama database berjalan dan jika ada perubahan maka dapat dilakukan *upgrade*.

ADA 6 FASE PROSES PERANCANGAN DATABASE

1. Pengumpulan dan analisa
   a. Menentukan kelompok pemakai dan bidang-bidang aplikasinya
   b. Peninjauan dokumentasi yang ada
   c. Analisa lingkungan operasi dan pemrosesan data
   d. Daftar pertanyaan dan wawancara
2. Perancangan database secara konseptual
   a. Perancangan skema konseptual
   b. Perancangan transaksi yang akan terjadi dalam database.
3. Pemilihan DBMS
   a. Faktor teknis

      Contoh faktor teknik :

      Tipe model data (hirarki, jaringan atau relasional), Struktur penyimpanan dan jalur pengaksesan yang didukung system manajemen database, Tipe interface dan programmer, Tipe bahasa query

b. Faktor Ekonomi dan Politik organisasi Faktor-faktor ekonomi: Biaya penyiadaan hardware dan software, Biaya konversi pembuatan database, Biaya personalia, dll Faktor Organisasi :

Analisa 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 :
  a. Setiap calon anggota yang akan menjadi anggota harus mengisi formulir dengan biaya administrasi Rp.10.000,-
  b. Anggota dapat meminjam buku maksimal 3 buku
  c. Untuk masa peminjaman selama 1 minggu (7 hari)
  d. Keterlambatan pengembalian dikenakan denda sesuai dengan kondisi denda, diantaranya :

1. Denda keterlambatan pengembalian dikenakan biaya administrasi Rp.500 perharinya (bukti surat denda terlampir)
2. Denda Buku perpustakaan rusak maka dikenakan biaya revisi buku perpustakaan(biaya ini dikenakan setelah buku diperbaiki).(bukti surat denda terlampir)
3. Denda Buku Hilang, maka dikenakan biaya penggantian seharga buku tersebut.(bukti surat denda terlampir)
4. Perpustakaan smart dapat menerima sumbangan dari donatur statusnya (anggota atau masyrakat luas).

Analisa Kasus Enterprise

(Pembahasan di Kelas)

- ✓ Buat Enterprise dari "Perpustakaan Smart" yang ditentukan dari :**Entitas, Atribute/Field, value data, record dan bentuk tabel – tabel dari Enterprise**
- ✓ Bentuk Gambar dari Enterprise Perpustakaan (yang menghubungkan relasi antara Entitas, Atribute, value data, record dan tabel-tabel)

4. Perancangan database secara logik (data model mapping)

a. Pemetaan (Transformasi data)

Transformasi yang tidak tergantung pada sistem, pada tahap ini transformasi tidak mempertimbangkan karakteristik yang spesifik atau hal– hal khusus yang akan diaplikasikan pada sistem manajemen database

b. Penyesuaian skema ke DBMS

Penyesuaian skema yang dihasilkan dari tahap Pemetaan untuk dikonfirmasikan pada bentuk implementasi yang spesifik dari suatu model data seperti yang digunakan oleh sistem manajemen database yang terpilih.

5. Perancangan database secara fisik

a. Response Time

Waktu transaksi database selama eksekusi untuk menerima respon

b. Space Utility

Jumlah ruang penyimpanan yang digunakan oleh database file dan struktur jalur pengaksesannya

c. Transaction Throughput

Merupakan nilai rata–rata transaksi yang dapat di proses permenit oleh sistem database dan merupakan parameter kritis dari sistem transaksi

6. Phase Implementasi Sistem Database

DBMS (Database Management Systems)

DBMS adalah perangkat lunak yang menangani semua pengaksesan database yang mempunyai fasilitas membuat, mengakses, memanipulasi dan memelihara basis data

BAHASA dalam DBMS

a. Data Definision Language (DDL)

Hasil kompilasi dari perintah DDL adalah satu set dari table yang disimpan dalam file khusus disebut data dictionary/directory.

b. Data Manipulation Language (DML)

Bahasa yang memperbolehkan pemakai untuk akses atau memanipulasi data sebagai yang telah diorganisasikan sebelumnya dalam model data yang tepat

Secara dasar ada dua tipe DML :

1. Prosedural, yang membutuhkan pemakai untuk menspesifikasikan data apa yang dibutuhkan dan bagaimana untuk mendapatkannya contoh dbase III, foxbase

2. Non prosedural, yang membutuhkan pemakai untuk menspesikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana untuk mendapatkannya. Contoh SQL, QBE.

FUNGSI DBMS

1. Data Definition, DBMS harus dapat mengolah pendefinisian data
2. Data Manipulation, DBMS harus dapat menangani permintaan dari pemakai untuk mengakses data
3. Data Security & Integrity, DBMS harus dapat memeriksa security dan integrity data yang didefinisikan oleh DBA.
4. Data Recovery & Concurency, DBMS harus dapat menangani kegagalan – kegagalan pengaksesan database yang dapat disebabkan oleh sesalahan sistem, kerusakan disk, dsb
5. Data Dictionary, DBMS harus menyediakan data dictionary.
6. Performance, DBMS harus menangani unjuk kerja dari semua fungsi seefisien mungkin.

KOMPONEN DBMS

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

PERBEDAAN TRADITIONAL FILE MANAGEMENT (FMS) DENGAN DATABASE MANAGEMENTSISTEM (DBMS)

TRADITIONAL FILE MANAGEMENT

1. Bersifat program oriented
2. Bersifat kaku
3. Terjadi kerangkapan data dan tidak terjaminnya keselarasan data (data inkonsistensi)

DATABASE FILE MANAGEMENT (DBMS)

1. Bersifat data oriented
2. Bersifat luwes/fleksible
3. Kerangkapan data serta keselarasan data dapat terkontrol

Keterangan :

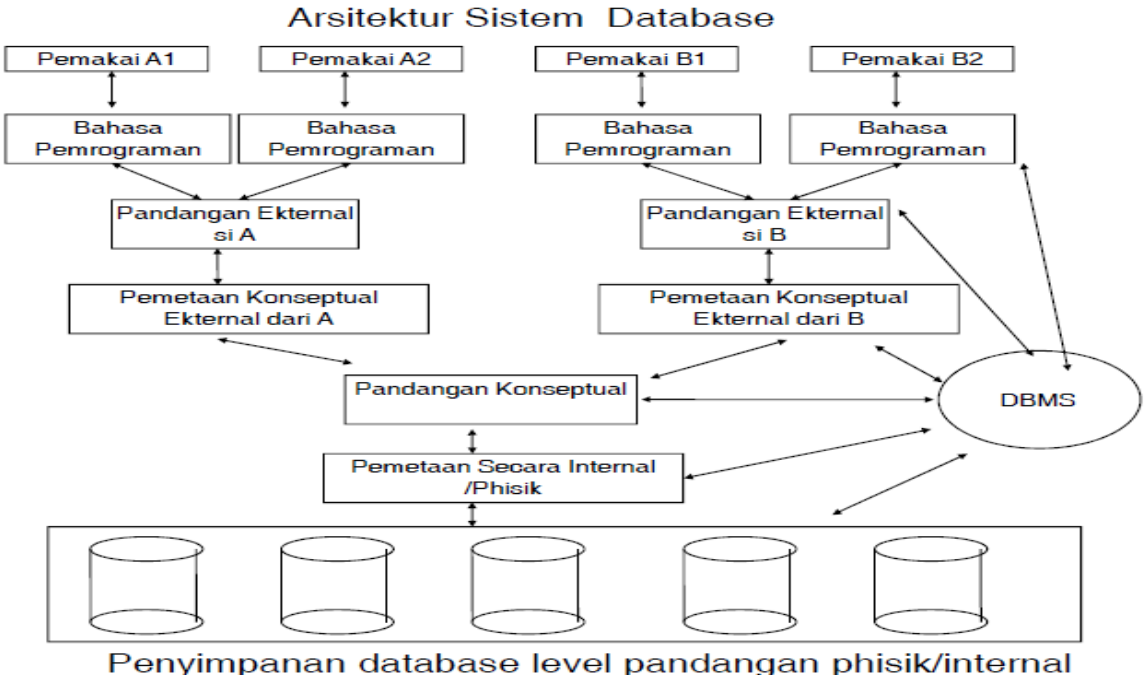
Program oriented " Susunan data di dalam file , distribusi data pada peralatan strorage, dan organisasi filenya dipilih sedemikian rupa, sehingga program aplikasi dapat menggunakan secara optimal "

Data oriented " Susunan data, organisasi file pada database dapat dirubah, begitu pula strategi aksesnya tanpa mengganggu program aplikasi yang sudah ada ".

ARSITEKTUR SISTEM DATABASE

Terbagi menjadi 3 tingkatan :

1. Internal level yaitu menerangkan struktur penyimpanan basisdata secara fisik dan organisasi file yang digunakan "
2. konseptual level yang menerangkan secara menyeluruh dari basisdata dengan menyembunyikan penyimpanan data secara fisik " Ekternal level yang menerangkan View basisdata dari sekelompok pemakai.



DATA INDEPENDENCE

Merupakan salah satu kelebihan sistem database dimana DBA dapat merubah struktur storage & stategi akses dalam pengembangan sistem database tanpa mengganggu program-program aplikasi yang sudah ada.

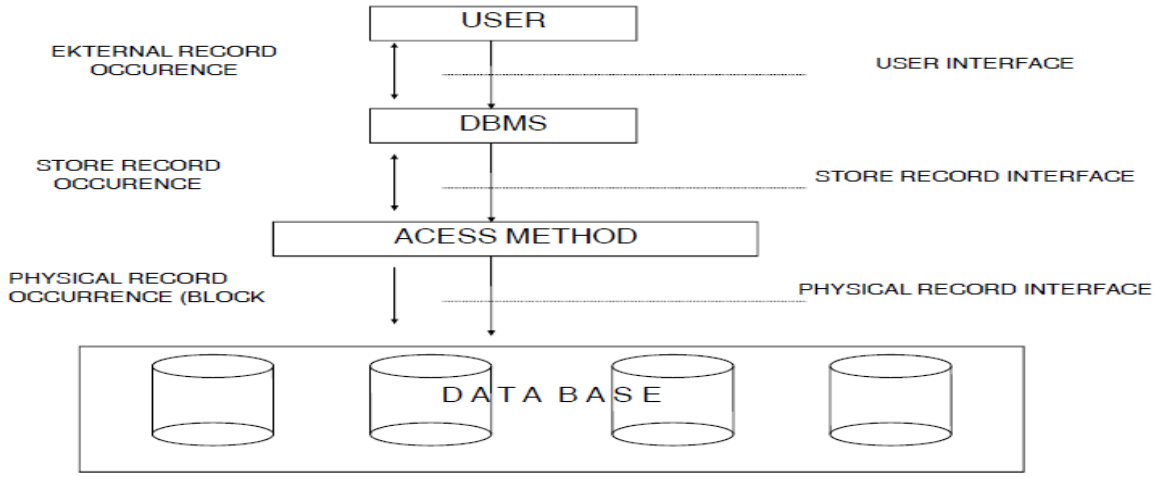
2 TINGKAT DATA INDEPENDENCE

1. Physical data independence yaitu perubahan internal schema dapat dilakukan tanpa menggangu conceptual schema
2. Logical data independence yaitu conceptual schema dapat dirubah tanpa mempengaruhi ekternal schema.

ALASAN PERLUNYA PRINSIP DATA INDEPENDENCE DITERAPKAN PADA PENGELOLAAN SISTEM DATABASE

1. Database Administrator dapat merubah isi, lokasi dan organisasi database tanpa mengganggu program aplikasi yang ada.
2. Vendor hardware & software pengelolaan data bisa memperkenalkan produk - produk baru tanpa mengganggu program - program aplikasi yang telah ada
3. Untuk memudahkan perkembangan program aplikasi
4. Memberikan fasilitas pengontrolan terpusat oleh DBA demi security dan integritas data, dengan memperhatikan perubahan - perubahan kebutuhan user.

ABSTRAKSI HUBUNGAN ANTARA USER PADA DBMS DENGAN PHYSICAL DATABASE,



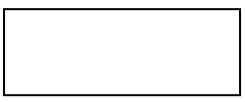
PENGERTIAN MODEL DATA :
Sekumpulan konsep-konsep untuk menerangkan data, hubungan-hubungan antara data dan batasan-batasan data yang terintegrasi di dalam suatu organisasi

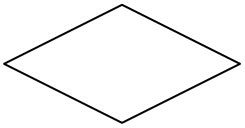
JENIS-JENIS MODEL DATA
A. Model data berbasis objek
B. Model data berbasis record
C. Model data fisik
D. Model data konseptual

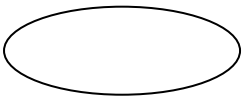
A. OBJECT BASED DATA MODEL
Model data berbasis objek menggunakan konsep entitas, atribut dan hubungan antar entitas.
Terdiri dari :
1. Entity Relationship model
2. Semantik data model

1. *ENTITY RELATIONSHIP MODEL*
Model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa real word terdiri dari objek-object dasar yang mempunyai hubungan atau relasi antara objek-objek tersebut

E-R MODEL berisi ketentuan /aturan khusus yang harus dipenuhi oleh isi database. Aturan terpenting adalah MAPPING CARDINALITIES, yang menentukan jumlah entity yang dapat dikaitkan dengan entity lainnya melalui relationship-set.
Simbol yang digunakan :


: Menunjukan object dasar

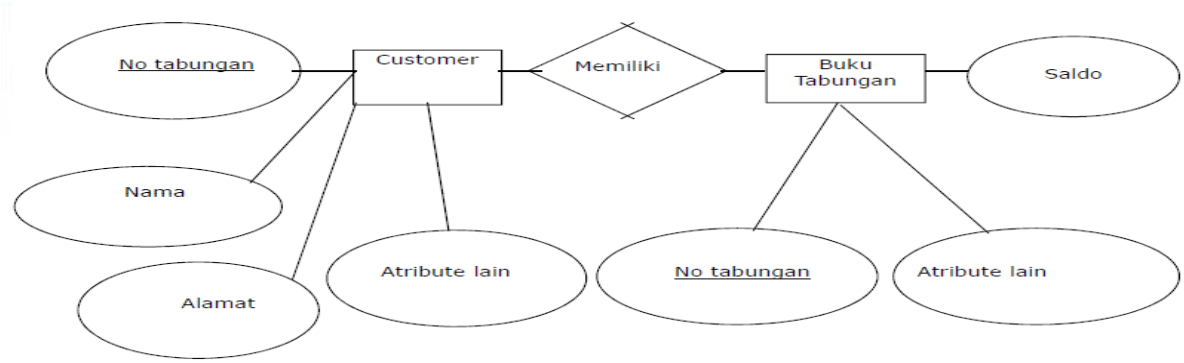

: Menunjukan relasi



: Menunjukan atribut dari objek dasar

: Menunjukan adanya relasi

Contoh Kasus ER - Model



2. BINARY MODEL

Pemetaan data dengan menggunakan 0 dan 1, atau true dan false dengan kondisi tertentu atau hanya dalam alternatif.

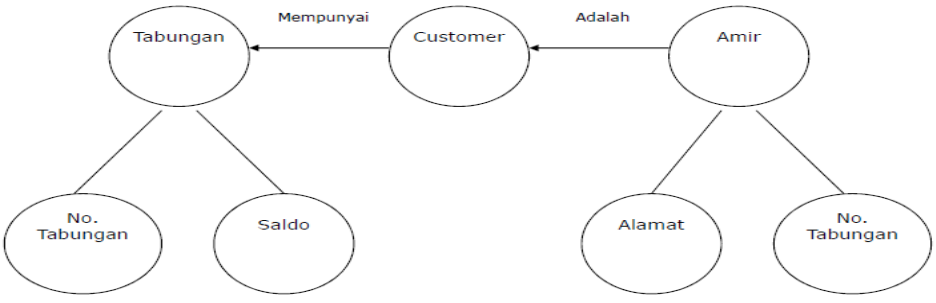
3. ***SEMANTIC MODEL***

Hampir sama dengan Entity Relationship model dimana relasi antara objek dasar tidak dinyatakan dengan symbol 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 sebagai berikut :

: Menunjukkan adanya relasi

: menunjukkan atribut

Contoh kasus Semantic model



B. RECORD BASED DATA MODEL

Model ini berdasarkan pada record untuk menjelaskan kepada user tentang hubungan logic antar data dalam basis data

PERBEDAAN DENGAN OBJECT BASED DATA MODEL

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)

Terdapat 3 data model pada record based data model :

1. Model Relational,

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

Contoh : data base penjual barang terdiri dari 3 tabel :

– Supllier
– Suku_cadang
– Pengiriman

SUPPLIER

No_supl	Nama_pen	Status	KOTA
S01	PT. OGAH-RUGI	03	MEDAN
s02	PT. SANTAI-DULU	03	SURABAYA
s03	PT. MALU-MALU	02	BANDUNG
s04	CV. ASAL JADI	01	MEDAN

NO_SUPL	NO_PART	JUML
S01	P01	200
S01	P02	300
S01	P04	250
S02	P01	300
S02	P02	400
S03	P03	400
S04	P04	300

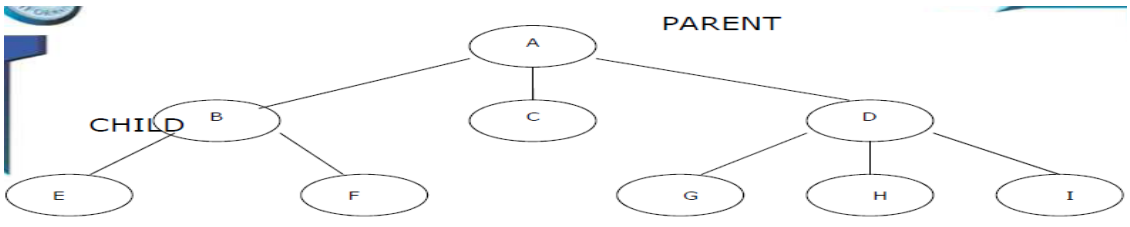
PENGIRIMAN

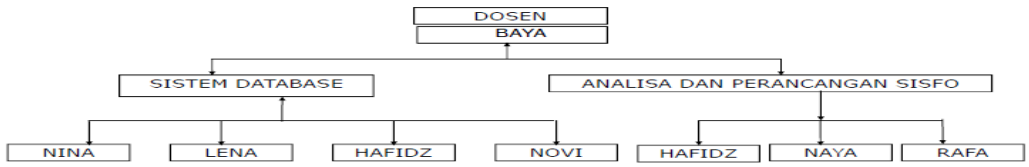
SUKU CADANG

NO_PART	NAMA_PART	BAHAN BAKU	BERAT	KOTA
P01	BAUT-3 cm	BESI	10	JAKARTA
P02	MUR - 3 cm	BESI	8	JAKARTA
P03	BAUT -10 cm	ALUMUNIUM	45	SURABAYA
P04	PACKING	KARET	6	MEDAN
P05	RING-MM	ALUMUNIUM	2	JAKARTA

2. Model Hirarki

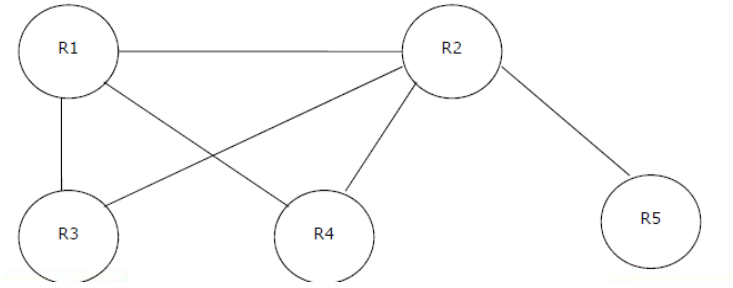
Dimana data serta hubungan antar data direpresentasikan dengan record dan link (pointer), dimana record-record tersebut disusun dalam bentuk tree (pohon), dan masing-masing node pada tree tersebut merupakan record/grup data elemen dan memiliki hubungan cardinalitas 1:1 dan 1:M.

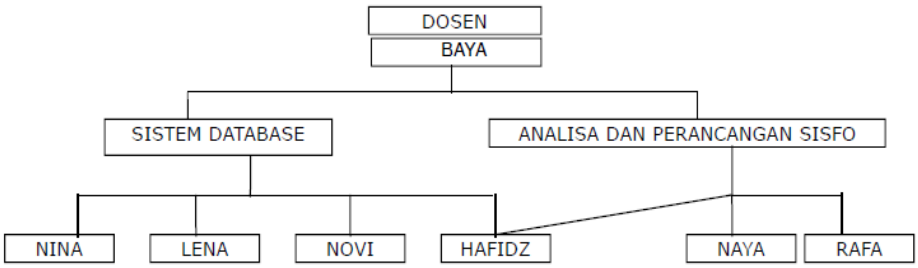




3. Model Jaringan

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





C.PHYSICAL DATA MODEL

Digunakan untuk menguraikan data pada internal level Beberapa model yang umum digunakan :

- ✓ Unifying model

  Model ini menggabungkan memori dan transaksi database dalam satu kesatuan model.

- ✓ Frame memory

  Frame Memory adalah sebuah virtual view dari tempat penyimpanan sekunder yang digunakan untuk mendukung penyimpanan record database

D. MODEL DATA KONSEPTUAL

Model yang dibuat berdasarkan anggapan bahwa dunia nyata terdiri dari koleksi obyek-obyek dasar yang dinamakan entitas (entity) serta hubungan (relationship) antara entitasentitas itu. Biasanya direpresentasikan dalam bentuk ***Entity Relationship Diagram***.

Manfaat Penggunaan CDM dalam perancangan database :

- ◈ Memberikan gambaran yang lengkap dari struktur basis data yaitu arti, hubungan, dan batasan-batasan
- ◈ Alat komunikasi antar pemakai basis data, designer, dan analis.

Analisa Kasus

(Perpustakaan Smart Lanjutan

Slide 1 & 2)

- Buat Model data berbasis objek (Semantik Model)
- Buat Model data berbasis record

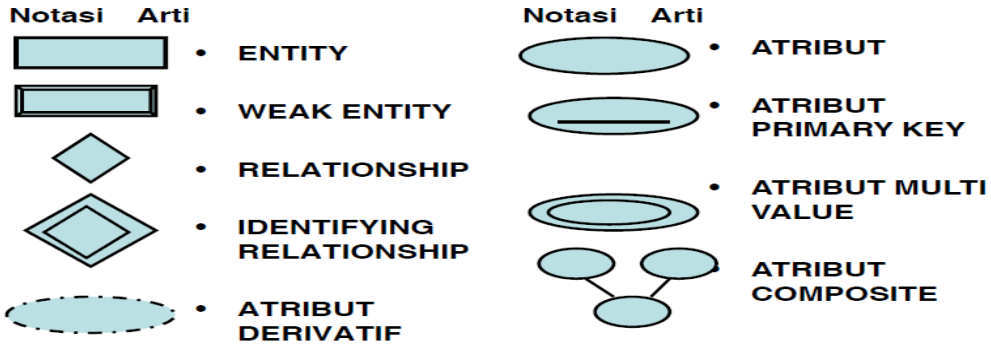
– Model Relational **– Model Jaringan** **– Model hirarki**

ENTITY RELATIONSHIP

PENGERTIAN

Entity relationship Adalah jaringan yang menggunakan susunan data yang disimpan dari sistem secara abstrak. Entity-relationship dari model terdiri dari unsur-unsur entity dan relationship antara entity-entitiy tersebut.

SIMBOL-SIMBOL ER-DIAGRAM

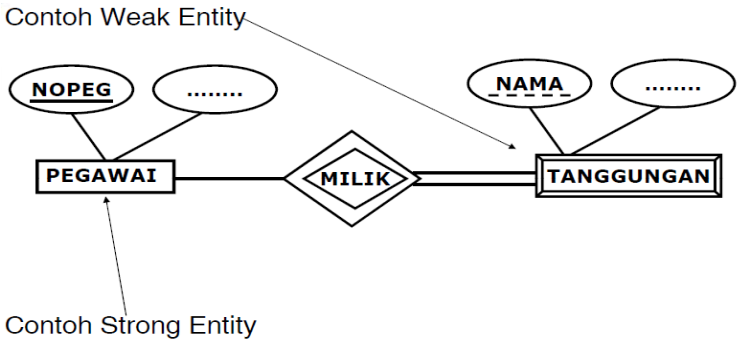


KOMPONEN ENTITY RELATIONSHIP

1. Entitas yaitu suatu kumpulan object atau sesuatu yang dapat dibedakan atau dapat diidentifikasikan secara unik. Dan kumpulan entitas yang sejenis disebut dengan entity set.
2. Relationship yaitu hubungan yang terjadi antara satu entitas atau lebih
3. Atribut, kumpulan elemen data yang membentuk suatu entitas.
4. Indicator tipe terbagi 2 yaitu :
   a. Indicator tipe asosiatif object
   b. Indicator tipe super tipe

ENTITY SET TERBAGI ATAS :

1. 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.
2. Weak Entity set, 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 : Strong entity set

NOPEG	NAMA
200107340	BILLY
200307569	FUAD
200107341	NINING
200107486	FINTRI

Weak entity set transaction

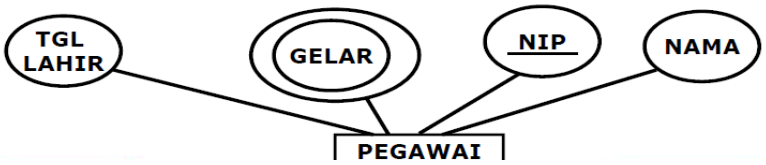
NOPEG	TANGGUNGAN	TANGGAL LAHIR	JENIS KELAMIN
200107340	HAFIDZ	22-03-2006	LAKI-LAKI
200307569	RENI	13-05-1999	PEREMPUAN
200107341	RAFFA	21-06-2006	LAKI-LAKI
200107486	NAIA	25-10-2006	PEREMPUAN

JENIS –JENIS ATRIBUT

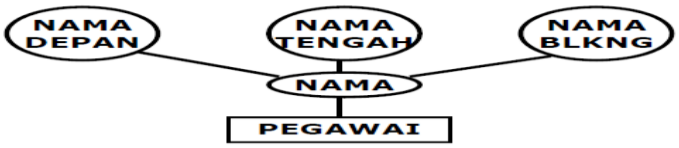
1. KEY →atribut yang digunakan untuk menentukan suatu entity secara unik
2. ATRIBUT SIMPLE →atribut yang bernilai tunggal

3. ATRIBUT MULTI VALUE →atribut yang memiliki sekelompok nilai untuk setiap instan entity

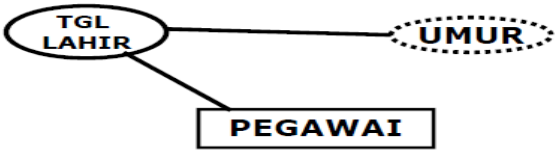
Pada gambar dibawah ini, yang menjadi atribut key adalah NIP. Tgl Lahir dan Nama adalah atribut simple. Sedangkan Gelar merupakan contoh atribut multivalue.



4. ATRIBUT COMPOSIT →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.

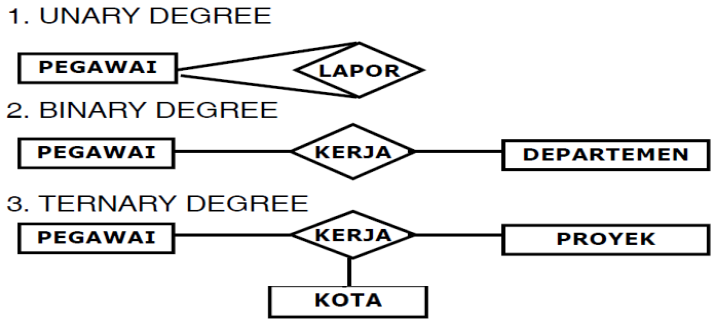


5. ATRIBUT DERIVATIF →Suatu atribut yg dihasilkan dari atribut yang lain. Sehingga umur yang merupakan hasil kalkulasi antara Tgl Lahir dan tanggal hari ini. Sehingga keberadaan atribut umur bergantung pada keberadaan atribut Tgl Lahir.



DERAJAT RELATIONSHIP

menjelaskan jumlah entity yang berpartisipasi dalam suatu relationship



MAPPING CARDINALITY

Banyaknya entity yang bersesuaian dengan entity yang lain melalui relationship

JENIS-JENIS MAPPING :

1. One to one

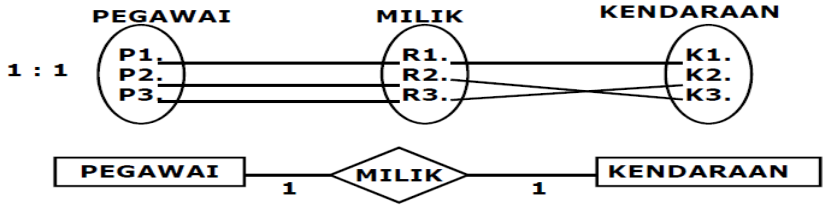
2. Many to One atau One to many

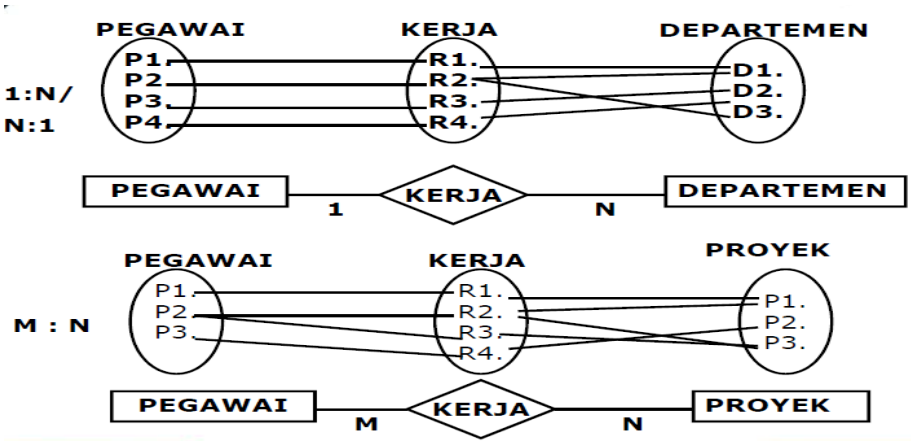
3. Many to many

REPRESENTASI DARI ENTITY SET

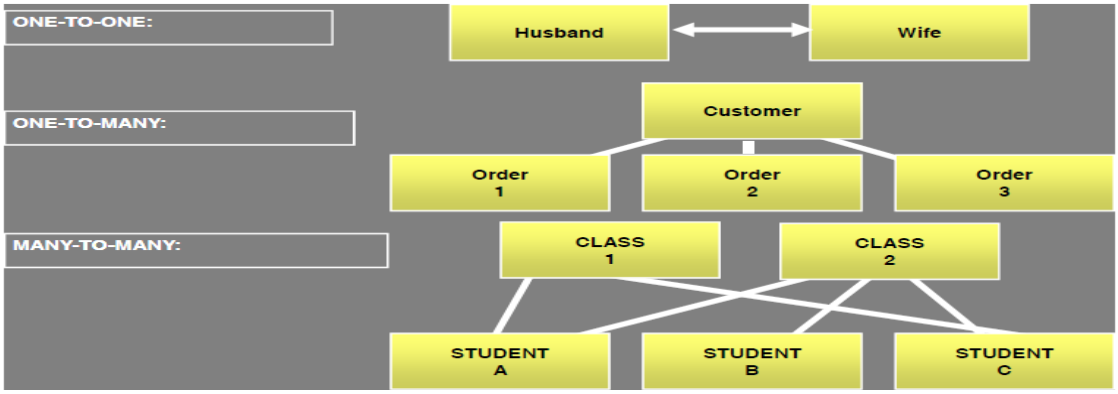
Entity set direpresentasikan dalam bentuk tabel dan nama yang unique. Setiap tabel terdiri dari sejumlah kolom, dimana masing-masing kolom diberi nama yang unique pula

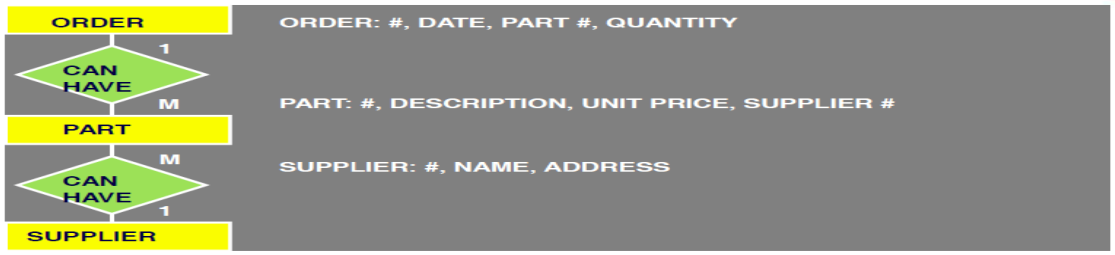
- **CARDINALITY RATIO CONSTRAINT**, Menjelaskan batasan jml keterhubungan satu entity dgn entity lainnya Jenis Cardinality Ratio = 1:1 1:N/ N:1 M : N

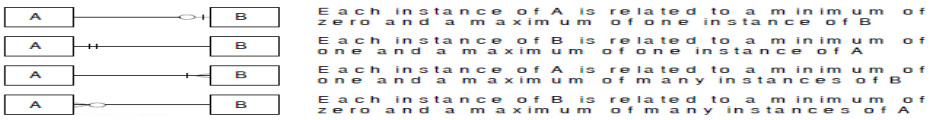




Cardinality 1:1,1:M,M:N

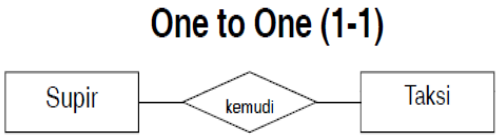






Logical Record Structured (LRS)

LRS → representasi dari struktur record-record pada tabel-tabel yang terbentuk dari hasil relasi antar himpunan entitas. **Menentukan Kardinalitas, Jumlah Tabel dan Foreign Key (FK).**



Gambar di atas menunujukan relasi dengan kardinalitas 1-1, karena:

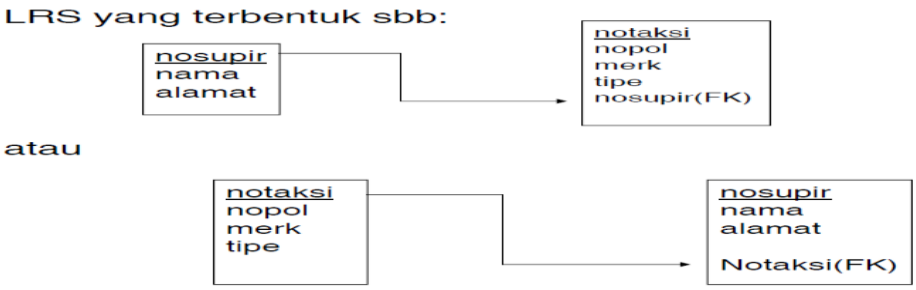
1 supir hanya bisa mengemudikan 1taksi, dan

1 taksi hanya bisa dikemudikan oleh 1 supir.

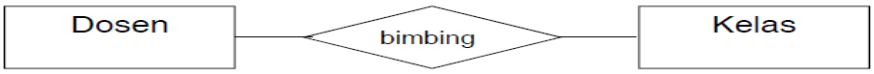
Relasi 1-1 akan membentuk 2 tabel:

Tabel Supir (nosupir, nama, alamat)

Tabel Taksi (notaksi, nopol, merk, tipe)



One to Many (1-M)



Gambar di atas menunujukan relasi dengan kardinalitas 1-M, karena:

1 Dosen bisa membimbing banyak Kelas, dan

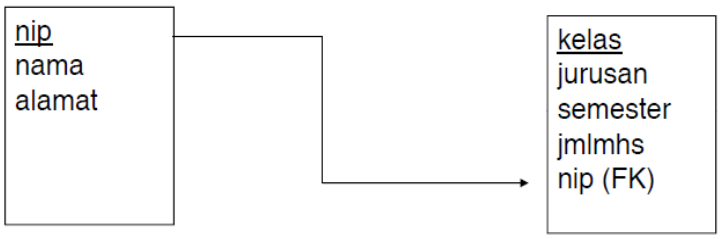
1 Kelas hanya dibimbing oleh 1 Dosen.

Relasi 1-M akan membentuk 2 tabel:

Tabel Dosen (**nip**, nama, alamat)

Tabel Kelas (**kelas**, jurusan, semester, jmlmhs)

LRS yang terbentuk sbb:



Many to Many (M-M)

Gambar di atas menunujukan 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:

Tabel Mahasiswa (**nim**, nama, alamat)

Tabel Mtkuliah (**kdmk**, nmmk, sks)

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

Participation Constraint

Menjelaskan apakah keberadaan suatu entity tergantung pada hubungannya dengan entity lain. Terdapat dua macam participation constrain yaitu:

1. Total participation constrain yaitu:

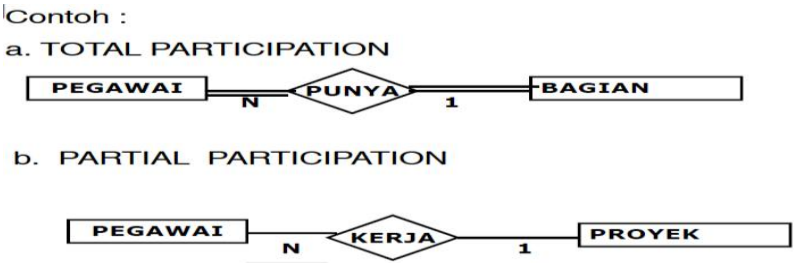
Keberadaan suatu entity tergantung pada hubungannya dengan entity lain.

Didalam diagram ER digambarkan dengan dua garis penghubung antar entity dan relationship.

2. Partial participation, yaitu

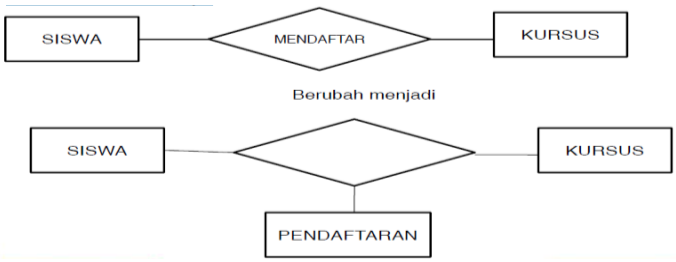
   Keberadaan suatu entity tidak tergantung pada hubungan dengan entity lain.

   Didalam diagram ER digambarkan dengan satu garis penghubung.

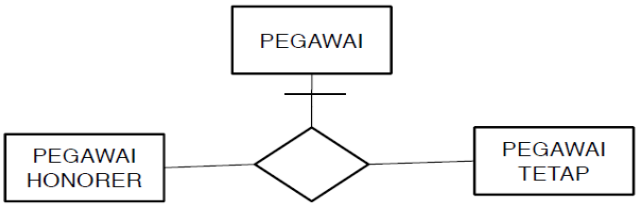


INDICATOR TIPE

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



Indicator tipe super tipe, terdiri dari suatu object dan satu subkategori atau lebih yang dihubungkan dengan satu relationship yang tidak bernama.



Analisa Kasus ERD Perpustakaan Smart

(Lanjutan dari Slide 1,2 & 3)

1. Pembuatan gambar ERD dari Perpustkaan Smart

   Langkah –langkah pembuatan ER diagram

   Tentukan entity – entity yang diperlukan

   Tentukan relationship antar entity – entity.

   Tentukan cardinality ratio dan participation constraint

   Tentukan attribute – attribute yang diperlukan dari tiap entity

   Tentukan key diantara attribute – attribute.

   Tentukan LRS dari masing-masing relasi

   Hindari penamaan entity, relationship dan atribute yang sama

TEHNIK NORMALISASI

BEBERAPA PENGERTIAN NORMALISASI :

Normalisasi merupakan proses pengelompokan elemen data menjadi tabel–tabel yang menunjuk-kan entity dan relasinya. **Normalisasi** adalah proses pengelompokan atributeatribute dari suatu relasi sehingga membentuk WELL STRUCTURE RELATION.

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

WELL STRUCTURE 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.

Contoh :

Terdapat sebuah relation Course, dengan ketentuan sbb:

1. Setiap mahasiswa hanya boleh mengambil satu matakuliah saja.
2. Setiap matakuliah mempunyai uang kuliah yang standar (tidak tergantung pada mahasiswa yang mengambil matakuliah tsb).

RELASI KURSUS

STUDENT-ID	KODE-MTK	BIAYA
92130	CS-200	75
92200	CS-300	100
92250	CS-200	75
92425	CS-400	150
92500	CS-300	100
92575	CD-500	50

- Relasi di atas merupakan sebuah relation yang sederhana dan terdiri dari 3 kolom/attribute
- Bila diteliti secara seksama, maka akan ditemukan redundancy pada datanya, dimana biaya kuliah selalu berulang pada setiap mhs. Akibatnya besar kemungkinan terjadi Error atau inkonsistensi data, bila dilakukan update terhadap relation tsb yang disebut dengan Anomali

ANOMALY merupakan penyimpangan-penyimpangan atau Error atau inkonsistensi data yang terjadi pada saat dilakukan proses insert, delete maupun update.

Terdapat 3 jenis Anomali :

1. Insertion Anomali

   Error yang terjadi sebagai akibat operasi insert record/tuple pada sebuah relation

   contoh :

   Ada matakuliah baru (CS-600) yang akan diajarkan, maka matakuliah tsb tidak bisa di insert ke dalam relation tsb sampai ada mhs yang mengambil matakuliah tsb.

2. Deletion Anomali

   Error yang terjadi sebagai akibat operasi delete record/tuple pada sebuah relation

   Contoh :

   Mhs dengan student-id 92-425, memutuskan untuk batal ikut kuliah CS-400, karena dia merupakan satu-satunya peserta matakuliah tsb, maka bila record/tuple tsb didelete akan berakibat hilangnya informasi bahwa mata- kuliah CS-400, biayanya 150.

3. Update Anomali

   Error 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 dinaikan dari 75 menjadi 100, maka harus dilakukan beberapa kali modifikasi terhadap record-record, tuple-tuple mhs yang mengambil matakuliah CS-200, agar data tetap konsisten.

Berdasarkan teori normalisasi, relation course dipecah menjadi 2 relation terpisah , sebagai berikut :

STUDENT-ID	KODE-MTK
92130	CS-200
92200	CS-300
92250	CS-200
92425	CS-400
92500	CS-300
92575	CD-500

KODE-MTK	BIAYA
CS-200	75
CS-300	100
CS-400	150
CS-500	50

PROBLEM-PROBLEM PADA RELATION YANG SUDAH DINORMALISASI

- Performance problem

  Masalah terhadap performa database

- Referential Integrity Problem

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

BEBERAPA KONSEP YANG HARUS DIKETAHUI:

a. Field/ Atribut Kunci
b. Kebergantungan Fungsi

1. Key Field / atribute kunci dalam database:

1. *Super key*

Yaitu himpunan dari satu atau lebih entitas yang digunakan untuk mengidentifikasikan secara unik sebuah entitas dalam entitas set.

2. *Candidate key*

Yaitu satu attribute atau satu set minimal attribute yang mengidentifikasikan secara unik suatu kejadian yang spesifik dari entity.

3. *Primary key*

Yaitu satu atribute atau satu set minimal atribute yang tidak hanya mengidentifikasikan secara unik suatu kejadian yang spesifik tapi juga dapat mewakili setiap kejadian dari suatu entity

4. *Alternate key*

Yaitu kunci kandidat yang tidak dipakai sebagai primary key

5. *Foreign key*

yaitu satu atribute (atau satu set atribute) yang melengkapi satu relationship (hubungan yang menunjukkan ke induknya.

SALES

S#	SNAME	KODE
S1	ADI	1002
S2	RAFI	1001
S3	HANY	1003

PESANAN

KODE	P#
1002	2648
1001	2649
1003	2641

- Super key = S#, SNAME, KODE
- Candidat key = S#, SNAME
- Primary key = S#
- Altenative key = SNAME
- Foreign key = KODE

b. Ketergantungan Kunci

1. **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 Aatau 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 tepatsatu harga Y dalam R.

2. **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

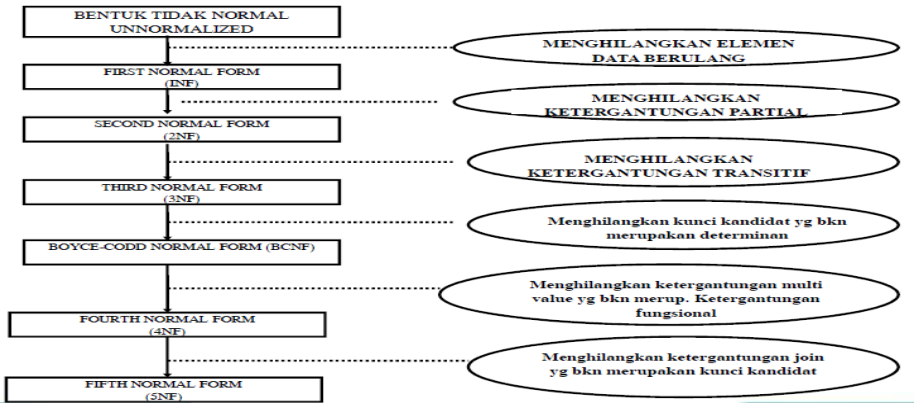
Contoh:

PersonID, Project, Project_budget ➔ time_spent_byperson_onProject (bukan FFD)

PersonID, Project ➔ time_spent_byperson_onProject (FDD)

3. Ketergantungan Partial
   Sebagian dari kunci dapat digunakan sebagai kunci utama
4. Ketergantungan Transitif
   Menjadi atribute biasa pada suatu relasi tetapi menjadi kunci pada relasi lain
5. Determinan
   Suatu atribute (field) atau gabungan atribute dimana beberapa atribute lain bergantung sepenuhnya pada atribute tersebut.

LANGKAH - LANGKAH PEMBENTUKAN NORMALISASI:



1. Bentuk tidak normal (*Unnormalized Form*):

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan saat menginput.

Contoh data :

no_siswa	Nama	PA	kelas1	kelas2	kelas3
22890100 22890101	Rafi Thoriq	Rachmat Adi	1234 1234	1543 1775	1543

Ket : PA = Penasehat Akademik

Siswa yg punya nomor siswa, nama, dan PA mengikuti 3 mata pelajaran/kelas. Disini ada perulangan kelas 3 kali ini bukan bentuk 1 NF.

1. Bentuk Normal Ke Satu (*1 NF/First Normal Form*)

Suatu relasi 1NF jika dan hanya jika sifat dari setiap relasi atributnya bersifat atomik.

Atom adalah zat terkecil yang masih memiliki sifat induknya, bila dipecah lagi maka ia tidak memiliki sifat induknya.

Ciri-ciri 1 NF :

- Setiap data dibentuk dalam flat file, data dibentuk dalam satu record demi satu record nilai dari field berupa "atomic value"
- Tidak ada set atribute yang berulang atau bernilai ganda
- Tiap field hanya satu pengertian

no_siswa	Nama	Pa	kode_kelas
22890100	Rafi	Rachmat	1234
22890100	Rafi	Rachmat	1543
22890101	Thoriq	Adi	1234
22890101	Thoriq	Adi	1775
22890101	Thoriq	Adi	1543

3. Bentuk Normal Ke Dua (*2 NF /Second Normal Form*)

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu. Atribute bukan kunci haruslah bergantung secara fungsi pada kunci utama/primary key. Sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci field. Kunci field haruslah unik dan dapat mewakili atribute lain yg menjadi anggotanya.

Misal :

Dari contoh relasi Siswa pada I NF terlihat bahwa kunci utama/primary key adalah nomor siswa. Nama siswa dan PA bergantung fungsi pada no_siswa, tetapi kode_kelas bukanlah fungsi dari siswa, maka file siswa dipecah menjadi 2 relasi.

Relasi Siswa

No-siswa	Nama	Pa
22890100 22890101	Rafi Thoriq	Rachmat Adi

dan

Relasi ambil_kelas

No-siswa	Kode_kelas
22890100	1234
22890100	1543
22890101	1234
22890101	1775
22890101	1543

4. Bentuk Normal Ke Tiga (*3 NF* / *Third Normal Form*)

Untuk menjadi bentuk normal ketiga maka relasi haruslah dalam bentuk normal kedua dan semua atribute bukan primer tidak punya hubungan yang transitif. Dengan kata lain, setiap atribute bukan kunci haruslah bergantung hanya pada primary key dan pada primary key secara menyeluruh.

Contoh pada bentuk normal kedua di atas termasuk juga bentuk normal ke tiga karena seluruh atribute yang ada disitu bergantung penuh pada kunci primernya.

5. *Boyce-Codd Normal Form* (BCNF)

BCNF mempunyai paksaan yg lebih kuat dari bentuk normal ketiga. Untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap atribute harus bergantung fungsi pada atribute superkey Pada contoh di bawah ini terdapat relasi seminar dengan

ketentuan sbb :

a. kunci primer adalah no_siswa+seminar.

b. Siswa boleh mengambil satu atau dua seminar.

c. Setiap siswa dibimbing oleh salah satu diantara 2 instruktur seminar tsb.

d. Setiap instruktur boleh hanya mengambil satu seminar saja.

Pada contoh ini no_siswa dan seminar menunjuk seorang instruktur :

Relasi seminar

no_siswa	Seminar	Instruktur
22890100	2281	Si doel
22890101	2281	Pak tile
22890102	2291	Mandra
22890101	2291	Basuki
22890109	2291	Basuki

Bentuk relasi seminar adalah bentuk normal ketiga, tetapi tidak BCNF karena nomor seminar masih bergantung fungsi pada instruktur, jika setiap instruktur dapat mengajar hanya pada satu seminar. Seminar bergantung fungsi pada satu atribute bukan superkey seperti yg disyaratkan oleh BCNF. Maka relasi seminar haruslah dipecah menjadi dua yaitu :

Relasi pengajar

Instruktur	Seminar
Si doel	2281
Pak tile	2281
Mandra	2291
Basuki	2291

no_siswa	Instruktur
22890100	Si doel
22890101	Pak tile
22890102	Mandra
22890101	Basuki
22890109	Basuki

6. **Bentuk Normal Ke Empat (4 NF)**
   Relasi R adalah bentuk 4 NF jika dan hanya jika relasi tersebut juga termasuk BCNF dan semua ketergantungan multivalue adalah juga ketergantungan fungsional
7. **Bentuk Normal Ke Lima (5 NF)**
   Disebut juga PJNF (Projection Join Normal Form) dari 4 NF dilakukan dengan menghilangkan ketergantungan join yang bukan merupakan kunci kandidat.

KASUS PENERAPAN NORMALISASI

PT. SANTA PURI — FAKTUR PEMBELIAN BARANG
Jalan senopati 11
Yogyakarta

Kode Suplier : G01 — Tanggal : 05/09/2000
Nama Suplier : Gobel Nustra — Nomor : 998

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

Jatuh tempo faktur : 09/09/2000

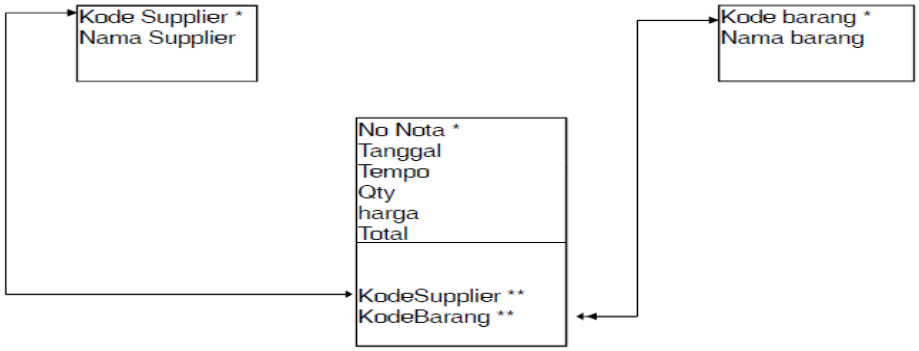
1.Step 1 bentuk unnormalized

no fac	kode supp	nama supp	kode brg	nama barang	tanggal	jatuh tempo	qty	harga	jumlah	Total
779 998	S02 G01	Hitachi Gobel N	R02 A01 A02	RICE COOKER AC SPLIT ½ PK AC SPLIT 1 PK	02/09/00 05/09/00	08/09/00 09/09/00	10 10 10	15000 135000 200000	150000 1350000 2000000	150000 3350000

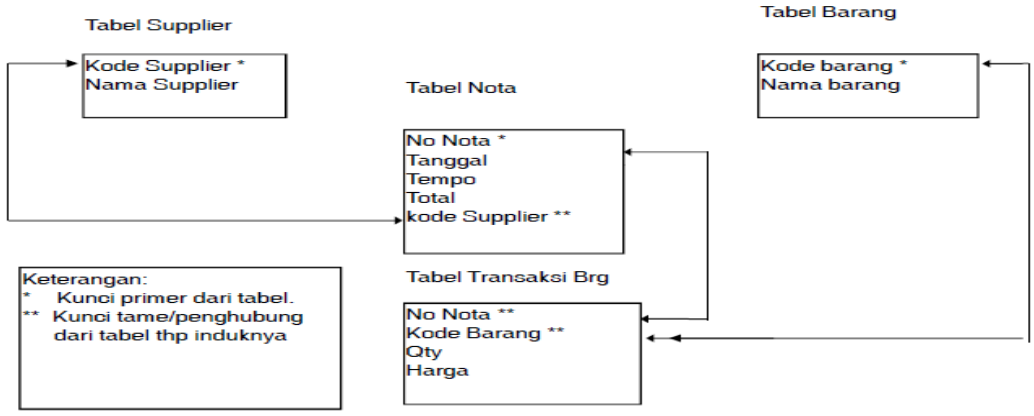
2. Step 2 bentuk 1 NF

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

3. Step 3 bentuk 2 NF



4. Step IV Bentuk 3 NF



Latihan

Buatlah bentuk Normalisasi dari dokumen berikut ini :

Kartu pengobatan masyarakat

No Pasien : 1234/PO/IV/99

Data Pasien dari, **NOPEN** : 1000019999

Alamat Pasien, **Jalan** : Kebon Jeruk No. 27

Kecamatan : Kemanggisan

Kode Pos :11530

Tanggal Pendaftaran : 1 Mei 1999

Nama Pasien : Bachtiar Jose

Kelurahan : Palmerah

Wilayah : Jakarta Barat

Telepon : 5350999

NoRM	Tgl periksa	Kode Dokter	Nama Dokter	KodeSakit	Diskripsi sakit	Kode obat	Nama obat	Dosis
RM001	1/5/99	D01	Dr Zurmaini	S11	Tropicana	B01 B02	Sulfa Anymiem	3dd1 4dd1
RM002	4/7/99	D01	Dr Zurmaini	S12	Ulcer Triombis	B01 B03	Sulfa Supralin	3dd2 3dd1
RM003	4/4/99	D02	Dr Harjono	S12	Ulcer Triombis	B04	Adrenalin	4dd2
RM004	7/8/99	D04	Dr Mahendra	S12	Ulcer Triombis	B01 B02 B03	Sulfa Anymiem Supralin	3dd2 4dd2 3dd1

BAHASA QUERY FORMAL

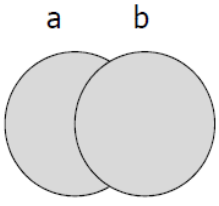
ALJABAR RELATIONAL

Adalah kumpulan operasi terhadap relasi, dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru

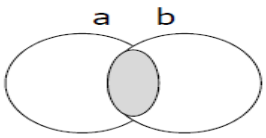
OPERATOR YANG DIGUNAKAN

A. OPERATOR HIMPUNAN

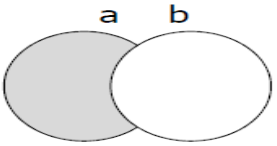
1. Union atau gabungan (È)
   Union dari relasi A dan B dinyatakan sebagai A È B



2. Intersection atau irisan (Ç)
   Intersection dari relasi A dan B dinyatakan sebagai A Ç B



3. Difference
   Difference dari relasi A dan B dinyatakan dengan A - B



4. Cartesian product

Product cartesian dari relasi A dan B dinyatakan dengan A X B

contoh :

A = { 1,2,3}

B = { 5,7 }

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

OPERATOR RELATIONAL

1. Restrict (σ) adalah Pemilihan tupel atau record
2. Project (π) adalah pemilihan attribute atau field
3. Divide ($\div$) adalah membagi
4. Join (θ) adalah menggabungkan

ALJABAR RELASIONAL

Operator pada aljabar relationaldibagi menjadi 2 kelompok :

1. Operator dasar untuk fundamental operational
2. Operator tambahan untuk additional operasional

Tabel dibawah ini adalah contoh untuk mengerjakan perintah – perintah Relation Algebra:

RELASI : MATA KULIAH

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

OPERATOR DASAR

a. Selection (s) Lower Case Omega

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

Contoh :

1. Mencari tuple-tuple dari MAHASISWA yang memiliki jenis kelamin laki-laki, Ekspresi aljabar relational : σ J_KEL = "LAKI-LAKI" (MAHASISWA
2. Tampilkan data mata kuliah yang memiliki kode 360 atau yang memilki sks 4
   σ KD_MK="306" V SKS = 4 (MATAKULIAH)

b. Projection (π)

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

Contoh :

Tampilkan nama beserta gaji dari dosen

πnama_dos,gaji (DOSEN)

c. Cartesian product (X)

Operator dengan dua relasi untuk menghasilkan tabel hasil perkalian kartesian.

Contoh :

Tampilkan nid,nama_d (dari relasi Dosen), nama_mk (dari relasi Matakuliah), thn_akademik,smt,hari,jam_ke,waktu,kelas (dari relasi Mengajar) dimana semester mengajar adalah pada semester '1'.

π nid, nama_d, nama_mk, thn_akademik,smt, hari,jam_ke, waktu, kelas (σ smt=1 ^Dosen.nid = Mengajar.nid ^mengajar.kdmk = Matakuliah.kdmk (DosenxMatakuliahxMengajar))

d. Union (È)

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 :

Penggabungan berdasarkan kolom kota dari table mahasiswa dengan tabel dosen

$$\pi \text{ kota } (\text{mahasiswa}) \cup \pi \text{kota } (\text{Dosen})$$

e. Set diference (-)

Operasi untuk mendapatkan tabel dis uatu relasi tapi tidak ada di relasi lainnya.

R – S = { X I X E R dan X E S }

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

Query I : tampilkan nama yang tinggal di depok

πnama_mhs(σ alamat="DEPOK" (MAHASISWA))

Query II : tampilkan nama yang berjenis kelamin perempuan

πnama_mhs(σ j_kel ="PEREMPUAN" (MAHASISWA))

Tampilkan query I minus query II :

πnama_mhs(salamat="DEPOK"(MAHASISWA))-
πnama_mhs(sj_kel="PEREMPUAN" (MAHASISWA))

OPERATOR TAMBAHAN

1. SET INTERSECTION (Ç)

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

2. THETA JOIN

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

3. NATURAL JOIN

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

4. DIVISION

Merupakan operasi pembagian atas tuple-tuple dari 2 relation

Contoh:

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

A

B

Pno
P2

A/B

Sno
S1
S2

Analisa Perpustakaan Smart

1. Diharapkan dosen untuk membuat tambahan kasus terbaru dari Perpustakaan Smart yang diimplementasikan dengan solusi Query bahasa Formal.

BAHASA QUERY KOMERSIAL

STRUKTUR QUERY LANGUAGE (SQL)

SQL dipublikasikan oleh **E.F. CODD** (1970) mengenai model relational. Kemudian pada tahun 1974, **D. Chamberlin** dan **R.F. Boyce** mengembangkan bahasa query untuk memanipulasi dan mengekstraksi data dari basisdata relational.

Sasaran SQL

1. Menciptakan basis data dan struktur relasi
2. Melakukan menajemen data tingkat dasar
3. Membentuk query sederhana dan kompleks
4. Melakukan tugas-tugas dengan seminimal mungkin memakai struktur dan sintaks perintah relatif mudah dipelajari
5. Harus portable

Jenis SQL :

1. Interactive SQL
2. Static SQL
3. Dynamic SQL

Subdivisi SQL

1. **DDL (*Data Definition Language*)**

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

2. **DML (*Data Manipulation Language*)**

   Query-query ini digunakan untuk manajemen data dalam basisdata

3. **DCL (*Data Control Language*)**

   Query-query ini berhubungan dengan pengaturan hak akses danwewenang.

PENGELOMPOKAN STATEMEN SQL

1. Data Definition Language (DDL)

CREATE DATABASE	DROP DATABASE
CREATE TABEL	DROP TABEL

CREATE INDEX DROP INDEX
CREATE VIEW DROP VIEW
ALTER TABLE

2. Data Manipulation Language
   INSERT, SELECT, UPDATE, DELETE
3. Data Access
   GRANT , REVOKE
4. Data Integrity
   RECOVER TABLE
5. Auxiliary
   SELECT INTO OUTFILE, LOAD, RENAME TABLE

KASUS DATA DEFINITION LANGUAGE (DDL)

A. CREATE

1. **Pembuatan Database**

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

   Sintaks : CREATE DATABASE nama_database

   Contoh : Buat database dengan nama KAMPUS

   CREATE DATABASE KAMPUS

2. **Pembuatan Tabel**

   Sintaks : CREATE TABLE nama_table

   (nama_kolom1 tipe_data_kolom1, nama_kolom2,tipe_data_kolom2,….)

   Contoh :

   Buat struktur tabel dengan nama tabel MHS dengan data NIM char(8),

   NAMA char(25), ALAMAT char(30)

   CREATE TABLE MHS (NIM char(8) not null,

   NAMA char(25) notnull, ALAMAT char(30) notnull)

3. Pembuatan Index

   Sintaks : CREATE [UNIQUE] INDEX nama_index

   ON nama_table (nama_kolom) ;

Contoh :

   Buat index data mahasiswa berdasarkan NIM dengan nama MHSIDX

   Dimana NIM tidak boleh sama

   CREATE UNIQUE INDEX MHSIDX ON MHS(NIM)

4. Pembuatan View

   Sintaks :

   CREATE VIEW nama_view [(nama_kolom1,….)]

   AS SELECT statement

   [WITH CHECK OPTION] ;

Contoh :

Buat view dengan nama MHSVIEW yang berisi semua data mahasiswa

CREATE VIEW MHSVIEW

AS SELECT * FROM MHS

B. DROP (MENGHAPUS)

1. **Menghapus Database**

Sintaks : DROP DATABASE nama_db ;

2. **Menghapus Tabel**

Sintaks : DROP TABLE nama_table ;

3. **Menghapus Index**

Sintaks : DROP INDEX nama_index ;

4. **Menhapus View**

Sintaks : DROP VIEW nama_view ;

Contoh :

DROP DATABASE **KAMPUS**;

DROP TABLE MHS;

DROP INDEX MHSIDX;

DROP VIEW MHSVIEW;

C. ALTER TABLE (MERUBAH STRUKTUR TABEL)

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

Contoh :

1. Tambahkan kolom JKEL dengan panjang 1 char pada tabel MHS

   ALTER TABLE MHS ADD JKEL char(1);

2. Ubah panjang kolom JKEL menjadi 15 char

   ALTER TABLE MHS MODIFY COLUMN JKEL char(15);

3. Hapus kolom JKEL dari data table MHS

   ALTER TABLE MHS DROP JKEL;

DATA MANIPULATION LANGUAGE (DML)

1. INSERT

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

Contoh :

Masukan data mhs dengan NIM 10296832 Nurhayati beralamat di Jakarta INSERT INTO MHS VALUES(“10296832”,”Nurhayati”,“Jakarta”); tambahkan record baru seperti dibawah ini.

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

2. UPDATE

Sintaks : UPDATE nama_tabel

SET nama_kolom = value_1

WHERE kondisi ;

Contoh :

Ubah alamat menjadi "Depok" untuk mahasiswa yang memiliki NIM "10296832"

UPDATE MHS

SET ALAMAT="Depok"

WHERE NIM=" 10296832";

3. DELETE

Sintaks : DELETE FROM nama_table

WHERE kondisi

Contoh :

Hapus data mahasiswa yang mempunyai NIM "21198002"

DELETE FROM MHS

WHERE NIM=" 21198002"

Tabel dibawah ini untuk mengerjakan Select (tampilan) dari SQL

Tabel MHS

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 Mata Kuliah

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

4. SELECT

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

Contoh :

a. Tampilkan semua data mahasiswa

SELECT NIM,NAMA,ALAMAT FROM MHS;

Atau

SELECT * FROM MHS;

Maka hasilnya adalah :

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

b. Tampilkan Mata Kuliah yang SKSnya 2

Select NAMA_MK from matakuliah Where sks = 2

Maka Hasilnya :

NAMA_MK
Sistem Basis Data
Pancasila

c. Tampilkan semua data nilai dimana nilai MID lebih besar sama dengan 60 atau nilai finalnya lebih besar 75. maka penulisannya :

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

JOIN

1. JOIN atau INNER JOIN
   Menggabungkan dua tabel dimana diantara dua table datanya bersesuaian.
2. LEFT JOIN atau LEFT OUTER JOIN
   Menggabungkan dua tabel dimana diantara dua table datanya bersesuaian dan juga semua record pada table sebelah kiri.
3. RIGHT JOIN atau RIGHT OUTER JOIN
   Menggabungkan dua tabel dimana diantara dua table datanya bersesuaian dan juga semua record pada table sebelah kanan.

```
SELECT Nilai.NIM, MHS.NAMA, Nilai.KD_MK, Nilai.MID
        FROM Nilai INNER JOIN MHS
        ON Nilai.NIM = MHS.NIM
```

Hasil :

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

```
SELECT MHS.NIM, MHS.NAMA, Nilai.KD_MK, Nilai.MID
        FROM MHS LEFT OUTER JOIN Nilai
        ON Nilai.NIM = MHS.NIM
```

Hasil :

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
10296001	Fintri	-	-
21198002	Julizar	-	-

```
SELECT MHS.NIM, MHS.NAMA, Nilai.KD_MK, Nilai.MID
        FROM Nilai RIGHT OUTER JOIN MHS
        ON Nilai.NIM = MHS.NIM
```

Hasil :

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
10296001	Fintri	-	-
21198002	Julizar	-	-

DATA ACCESS

1. GRANT

Sintaks :
```
GRANT hak_akses ON nama_db
TO nama_pemakai
[IDENTIFIED BY] [PASSWORD] 'Password'
[WITH GRANT OPTION] ;
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.

```
GRANT SELECT (FINAL) ON NILAI TO ADI
```

2. REVOKE

Sintaks :
```
REVOKE hak_akses ON nama_db
FROM nama_pemakai ;
REVOKE hak_akses ON nama_tabel
FROM nama_pemakai ;
```

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

```
REVOKE SELECT (FINAL) ON NILAI FROM ADI
```

DATA INTEGRITY

RECOVER TABLE

Sintaks : RECOVER TABLE nama_tabel

Contoh :

Kembalikan keadaan data mahasiswa seperti pada saat sebelum terjadi kerusakan

```
RECOVER TABLE MHS ;
```

AUXILIARY

1. SELECT … INTO OUTFILE 'filename'

Sintaks ini 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 '|'

```
SELECT * FROM MHS
INTO OUTFILE "/home/adi/teks"
FIELDS TERMINATED BY " ½";
```

2. LOAD

Sintaks query ini digunakan untuk mengimpor data dari file lainke 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) :

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

3. RENAME TABLE

Sintaks :

```
RENAME TABLE OldnamaTabel
TO NewNamaTabel
```

Contoh :

```
RENAME TABLE MHS
TO MAHASISWA
```

MENGGUNAKAN FUNGSI AGGREGATE :

1. **COUNT** digunakan untuk menghitung jumlah.
   Menghitung jumlah record mahasiswa dari tabel MAHASISWA
   SELECT COUNT(*) FROM MAHASISWA
2. **SUM** digunakan untuk menghitung total dari kolom yang mempunyai tipe data numerik.
   SELECT SUM(SKS) AS 'TOTAL SKS' FROM MATAKULIAH
3. **AVG** digunakan untuk menghitung rata-rata dari data-data dalam sebuah kolom.
   SELECT AVG(FINAL) AS 'FINAL' FROM Nilai
4. **MIN** digunakan untuk menghitung nilai minimal dalam sebuah kolom.
   SELECT MIN(FINAL) FROM Nilai
5. **MAX** diguankan untuk menghitung nilai maksimum dalam sebuah kolom
   SELECT MAX(MID) FROM Nilai.

SUBQUERY

Adalah subselect yang dapat digunakan di klausa **WHERE** dan **HAVING** dipernyataan select luar untuk menghasilkan tabel akhir. 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 disisi kanan pembandingan

Penggunanaan ANY dan ALL

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

Jika subquery diawali kata kunci **ANY**, syaratnya akan bernilai **TRUE** jika dipenuhi sedikitnya satu nilai yang dihasilkan subquery tersebut.

Penggunanaan EXIST DAN NOT 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**.

(Masing-masing dosen membuat contoh untuk subquery)

CONTOH SUBQUERY :

1. Coba ambil nilai mid dan final dari mahasiswa yang bernama Astuti.
   SELECT MID, FINAL FROM NILAI WHERE NIM=(SELECT
   NIM FROM MAHASISWAWHERE 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).

Aplikasi yang digunakan sebagai contoh adalah phptriad-mysql front

Dari Address ketik : http://localhost/phpmyadmin

Tampilan password ketik **root** dan untuk password ketik **password.**

FRAGMENTASI DATA

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.

Alasan-alasan diperlukannya fragmentasi, yaitu :

1. Penggunaan
2. Efisiensi
3. Paralleslisme
4. Keamanan

BEBERAPA PERATURAN YANG HARUS DIDEFINISIKAN
KETIKA MENDEFINISIKAN FRAGMENT :

1. 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.
2. 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.

3. Disjointness

data didalam fragmen tidak boleh diikutkan dalam fragmen lain agar tidak terjadi redundancy data, kecuali untuk atribut primary key dalam fragmentasi vertical

Kerugian

TIGA JENIS FRAGMENTASI :

1. Fragmentasi horisontal

terdiri dari tuple dari fragment global yang kemudian dipecah-pecah atau disekat menjadi beberapa sub-sets

2. Fragmentasi vertikal

Membagi atribut-atribut dari fragment global yang tersedia menjadi beberapa grup.

3. Fragmentasi campuran

Cara yang sederhana untuk membangun fragmentasi campuran sbb :

a. Menggunakan fragmentasi horisontal pada fragmentasi vertikal

b. Menggunakan fragmentasi vertical pada fragmentasi horizontal

CONTOH 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 **Horisontal** terbagi menjadi 3 fragment yang berbeda berdasarkan Mt_Kuliah

1. Relasi Mt_Kuliah="Sistem Basis Data"

σMt_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

2. 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

3. 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

Fragment di atas memenuhi kondisi jika Nama_Mhs dan Mt_Kuliah adalah 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 Data	78	B	1
124	Farah	102	Peranc. Sistem	60	C	2
125	Sarah	101	Sistem Basis Data	40	D	3
126	Salsabila	101	Sistem Basis Data	90	A	4
127	Azizah	103	Visual Basic	70	B	5
128	Farhan	103	Visual Basic	40	D	6
129	Faiz	102	Peranc. Sistem	80	A	7

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

Fragmentasi Campuran

Terdapat relasi berdasarkan Mata Kuliah yang sama
Relasi 1a.
π NIM,Nama_Mhs,Mt_Kuliah,Nil_Akhir,Grade,Tuple_ID(s 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.
πNIM,Nama_Mhs,Mt_Kuliah,Nil_Akhir,Grade,Tuple_ID(s 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
p NIM,Nama_Mhs,Mt_Kuliah,Nil_Akhir,Grade,Tuple_ID(s 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

Bagaimana bentuk database untuk data yang telah mengalami proses fragmentasi? Fragmentasi data merupakan langkah yang diambil untuk menyebarkan data dalam database terdistribusi . Selanjutnya akan dibahas apa yang dimaksud Database terdistribusi.

DATABASE TERDISTRIBUSI

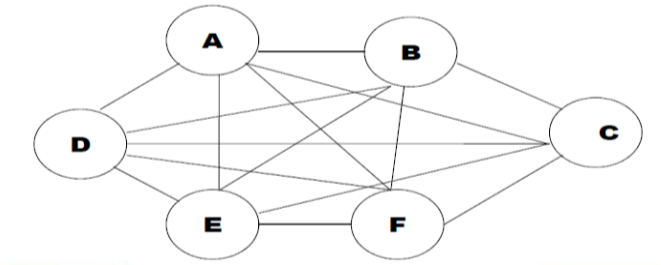
Yaitu kumpulan data yang digunakan bersama yang saling terhubung secara logik tetapi tersebar secara fisik pada suatu jaringan komputer.

Karakteristik Database terdistribusi, yaitu :

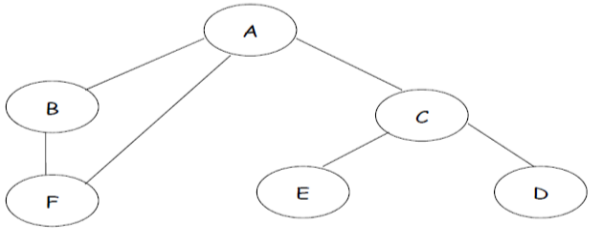
1. Kumpulan data yang digunakan bersama secara logik tersebar pada sejumlah komputer yang berbeda
2. Komputer yang dihubungkan menggunakan jaringan komunikasi
3. Data pada masing-masing situs dapat menangani aplikasiaplikasi lokal secara otonom
4. Data pada masing situs dibawah kendali satu DBMS
5. Masing-masing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

BENTUK-BENTUK TOPOLOGI DISTRIBUSI DATA :

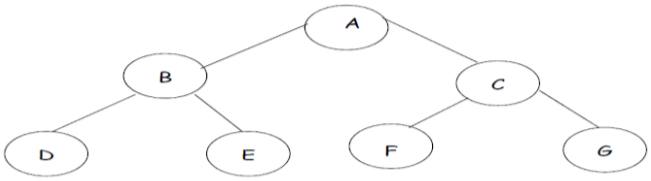
a. Fully Connected network



b. Partialy conneted network

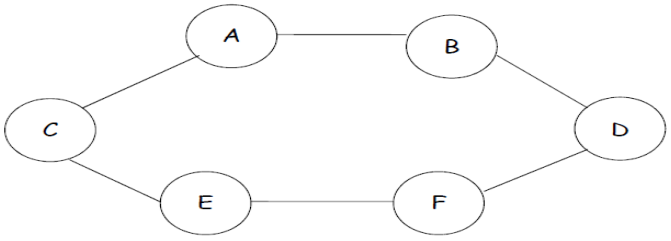


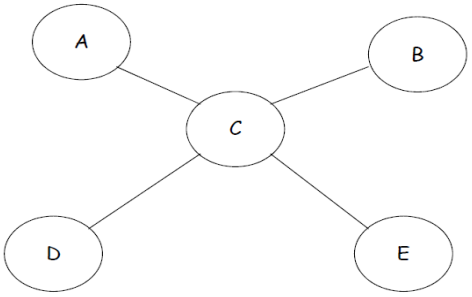
c. Tree Strutured Network



d. Ring network

e. Star network





KEUNTUNGAN DAN KERUGIAN DATABASE TERDISTRIBUSI

KEUNTUNGAN :

1. Secara alami mengikuti struktur organisasi
2. Adanya otonomi lokal
3. Sifatnya dapat dipakai secara bersama
4. Peningkatan ketersediaan
5. Peningkatan kehandalan
6. Peningkatan kinerja
7. Ekonomis
8. Pertumbuhan yang modular

KERUGIAN :

1. Harga software mahal (Biaya)
2. Kompleksitas
3. Kelemahan dalam keamanan
4. Sulitnya menjaga keutuhan data
5. Kurangnya standar
6. Kurangnya pengalaman
7. Perancangan basisdata lebih kompleks

Analisa kasus Perpustakaan Smart

- Pembuatan Salah satu Topologi Jaringan Database Terdistribusi dari Perpustakaan Smart.
- Implementasikan Perpustakaan Smart Fragmentasikan dengan 3 kondisi :***F.Horizontal,F.Vertikal & F.Campuran***

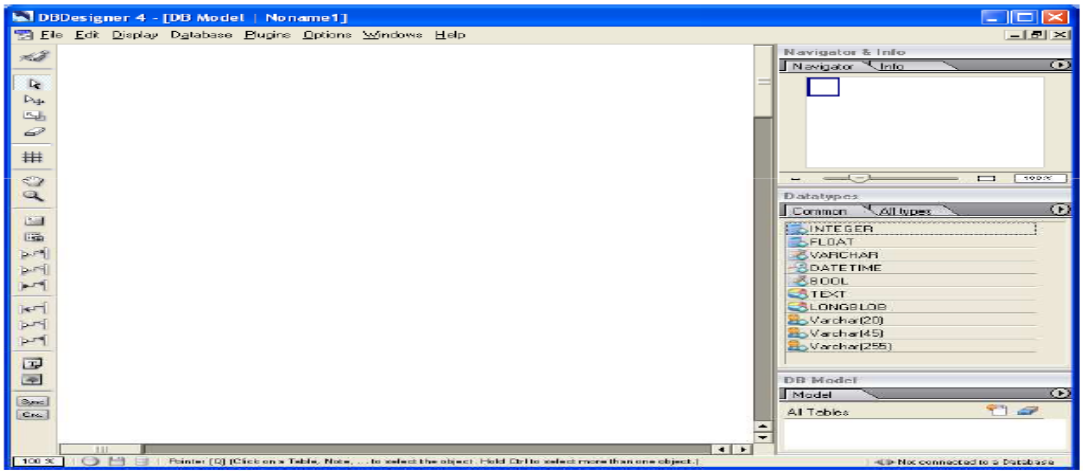
Diharapkan Dosen untuk memberikan Analisa kasus Perpustakaan Smart dan Solusi dari permasalahannya.

PERANCANGAN DAN IMPLEMENTASI BASIS DATA
MENGGUNAKAN MYSQL

Perangkat Lunak Bantu untuk Perancangan Basis Data

Pada perangkat lunak bantu telah tersedia komponenkomponen (notasi-notasi) perancangan basis data. Salah satu perangkat lunak bantu untuk keperluan semacam itu adalah DBDesigner yang dioptimalkan untuk MySQL Database.

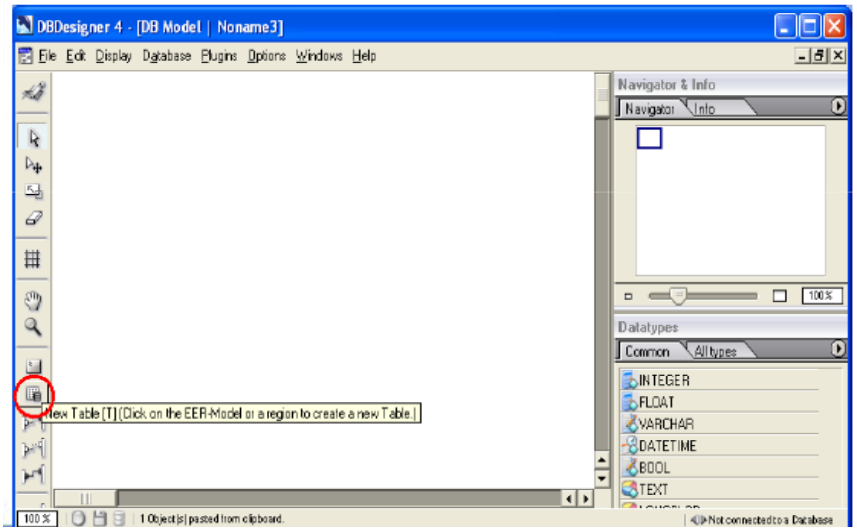
Tampilan jendela DBDesigner.



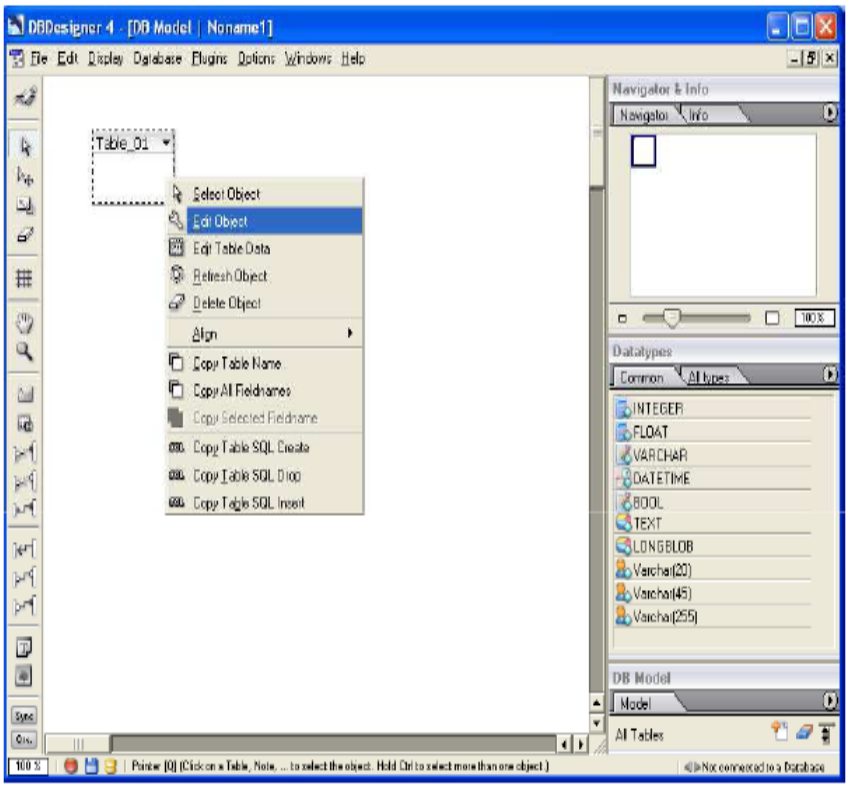
Contoh penggunaan DBDesigner

Menggunakan Komponen TABEL dan RELASI

Klik komponen **Tabel** pada toolbar seperti di gambar berikut.



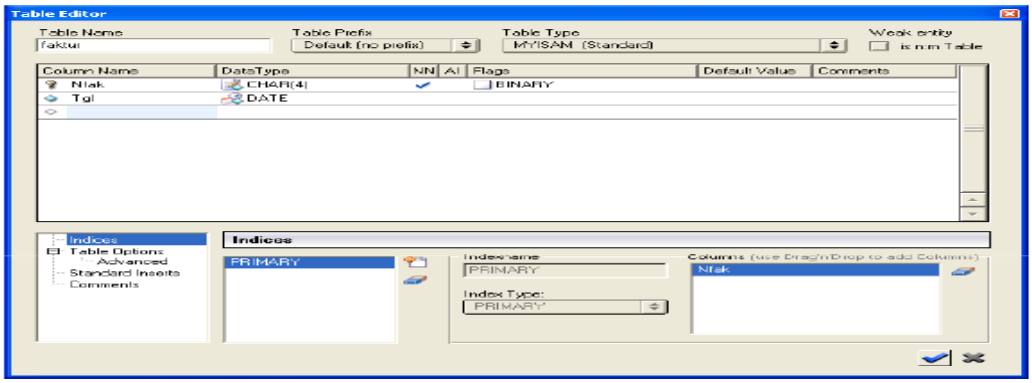
Letakan komponen tsb. pada page area sehingga muncul komponen **Tabel** (Table_01) pada page area, kemudian klik kanan komponen tsb sehingga muncul menu dan pilihlah **Edit Object** seperti berikut.



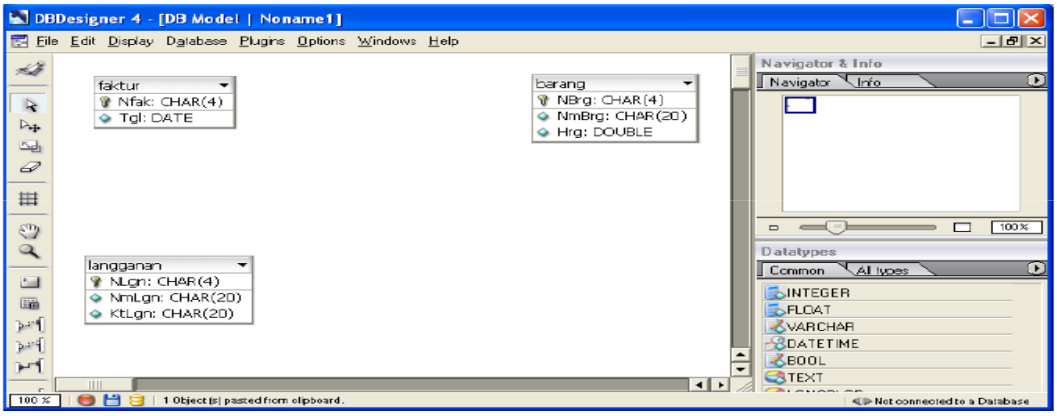
Menu Edit Object akan menampilkan jendela **Table Editor**.

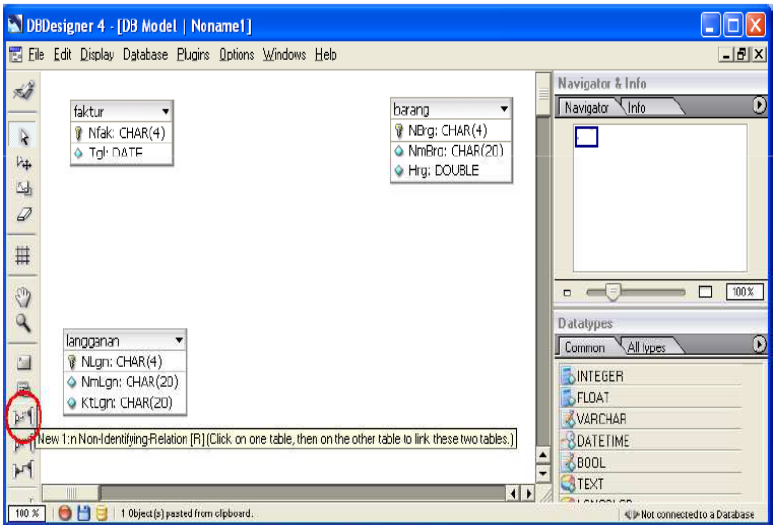
Pada **Table Editor** kita bisa menentukan properties dari tabel seperti nama tabel, tipe data, primary key dsb.

Ubah dan simpanlah properties tabel (Table _01) menjadi tabel **faktur** (struktur tabel seperti pada pembahasan LRS tanpa ada FK) seperti berikut



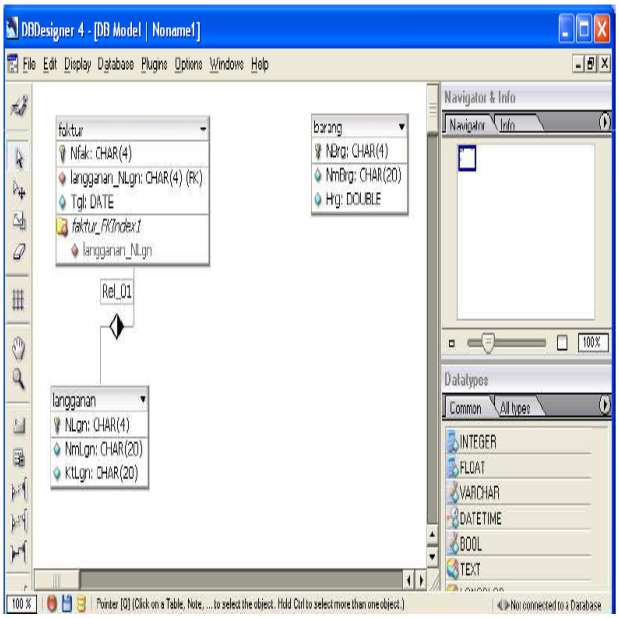
Ulangi langkah-langkah menggunakan komponen **Table** di atas (**tabel faktur**) untuk tabel **barang** dan **langganan** (struktur tabel seperti pada pembahasan LRS tanpa ada FK). Sehingga ada 3 komponen Table seperti gambar berikut



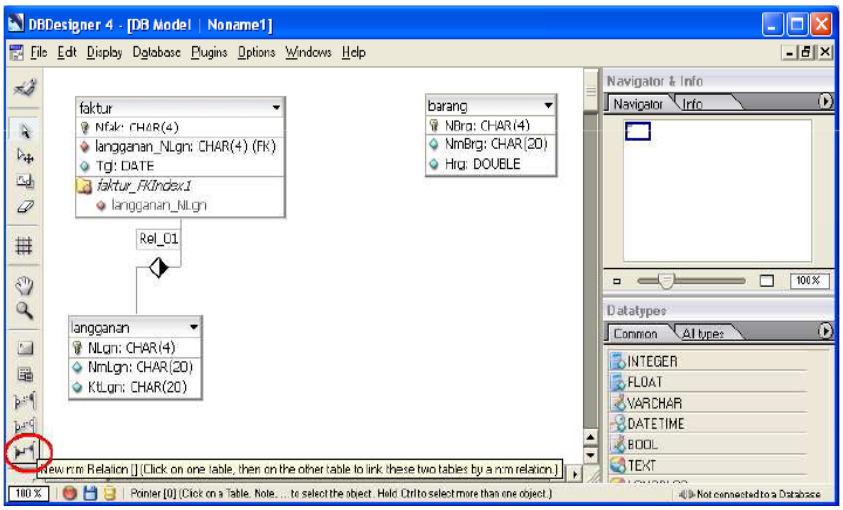


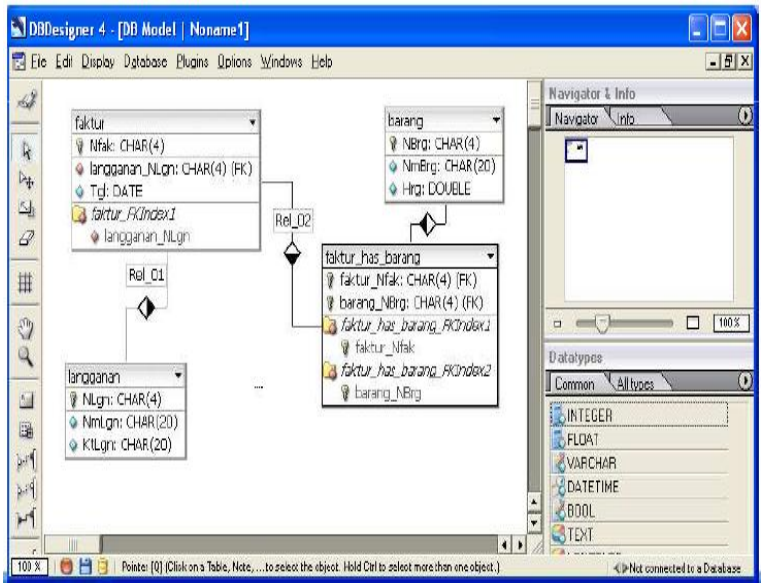
Langkah berikutnya membuat relasi **1-M** antara **langganan** dengan **faktur** dengan cara klik komponen **1-n Relation** pada toolbar seperti di gambar berikut.

Klik di tabel **langganan** kemudian klik di tabel **faktur**, sehingga muncul komponen relasi yang menghubungkan kedua tabel tsb. dan FK (NLgn) berada pada tabel faktur, seperti gambar berikut.

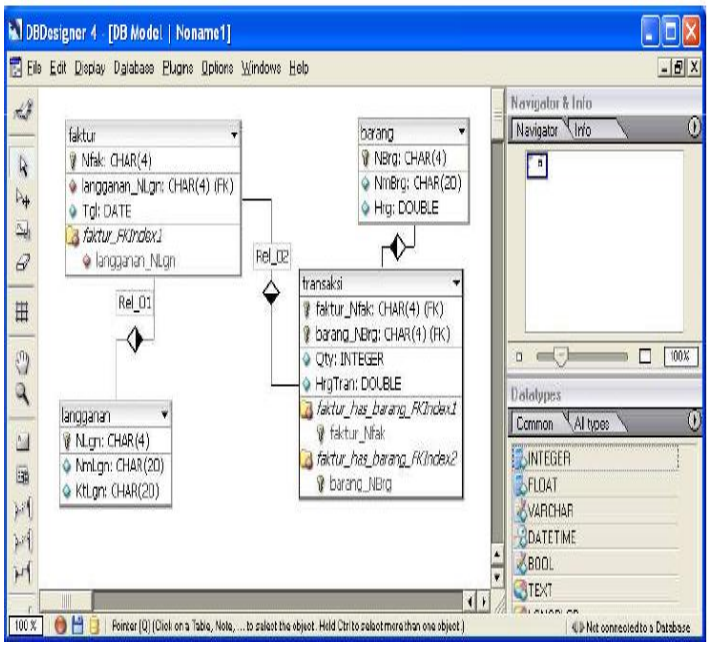


Langkah berikutnya membuat relasi **M-M** antara **faktur** dengan **barang** dengan cara klik komponen **n-m Relation** pada toolbar seperti di gambar berikut.





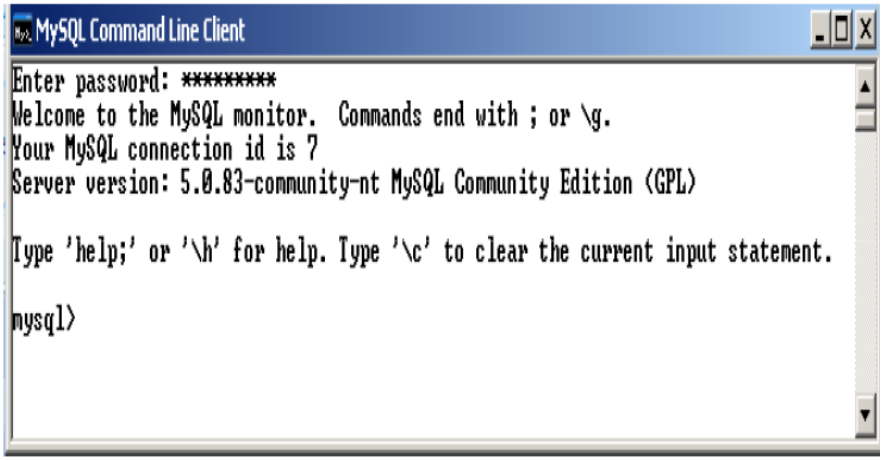
Klik di tabel **faktur** kemudian klik di tabel **barang**, sehingga muncul komponen relasi yang disertai munculnya tabel baru (**faktur_has_barang**) dan FK (Nfak & NBrg) berada pada table tsb, seperti gambar berikut.



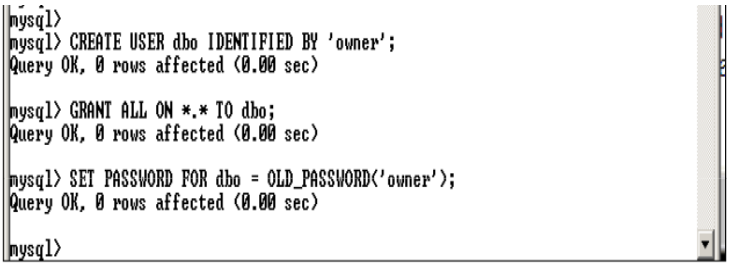
Edit properties tabel faktur_has_barang yaitu dengan mengganti nama menjadi tabel transaksi dan menambahkan field Qty dan HrgTran. Sehingga menjadi seperti gambar berikut.

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.

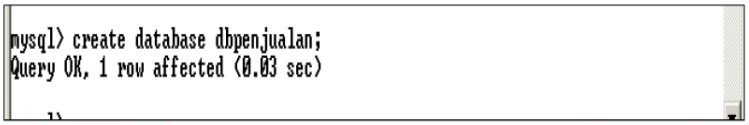
Lakukan login terlebih dahulu ke MySQL dengan memasukkan password root.

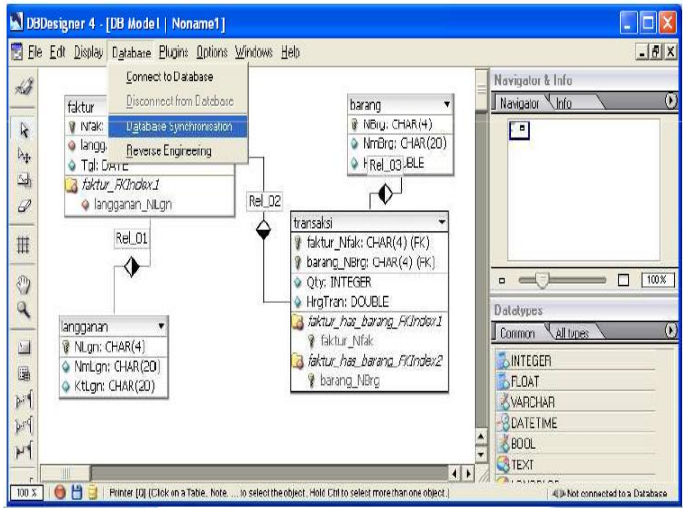


Buat user baru bernama dbo dengan password ”owner”. Ketikkan 3 perintah dibawah ini.



Buat Database baru yaitu dbpenjualan



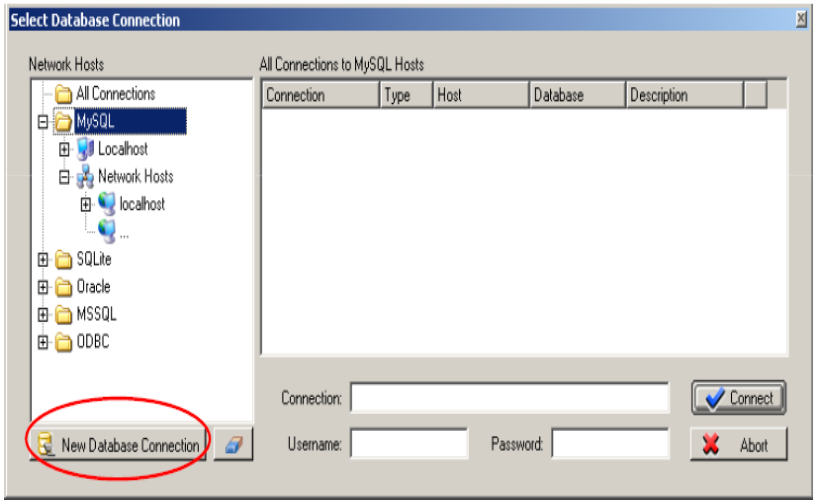


Mengekspor Tabel Hasil Rancangan Ke Server Database

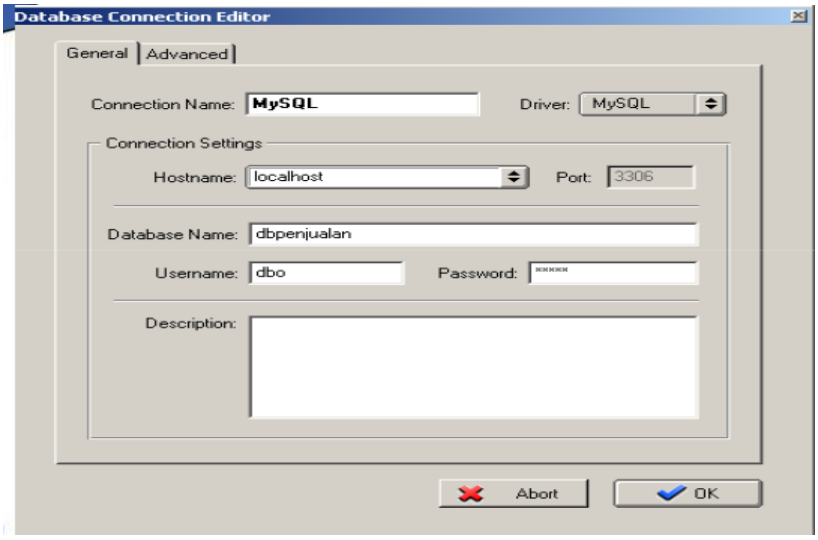
Mengekspor tabel ke server database bisa dilakukan dari menu

Database → Database Sychronisastion seperti gambar berikut.

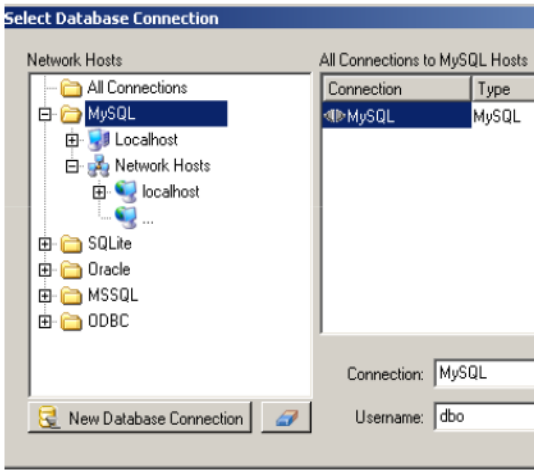
Lalu pilih MySQL sebagai database dan kemudian klik **New Database Connection**

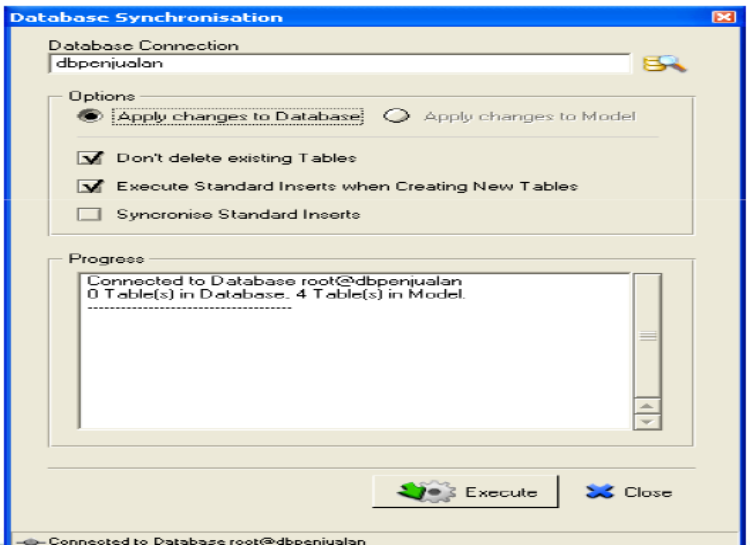


Masukkan Nilai berikut:
Connection Name : *MySQL*
Hostname : *localhost*
Database Name : *dbpenjualan*
UserName : *dbo*
Password : *owner*
Lalu klik OK



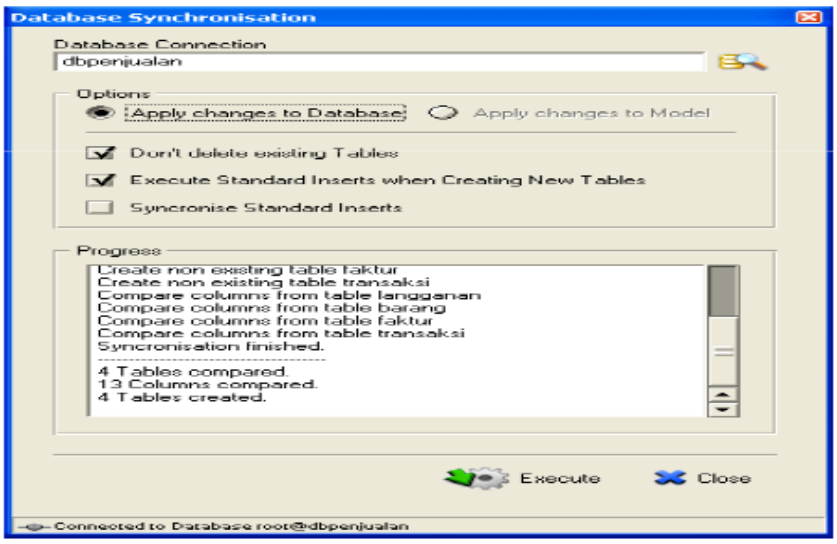
Klik Connect untuk terkoneksi ke MySQL

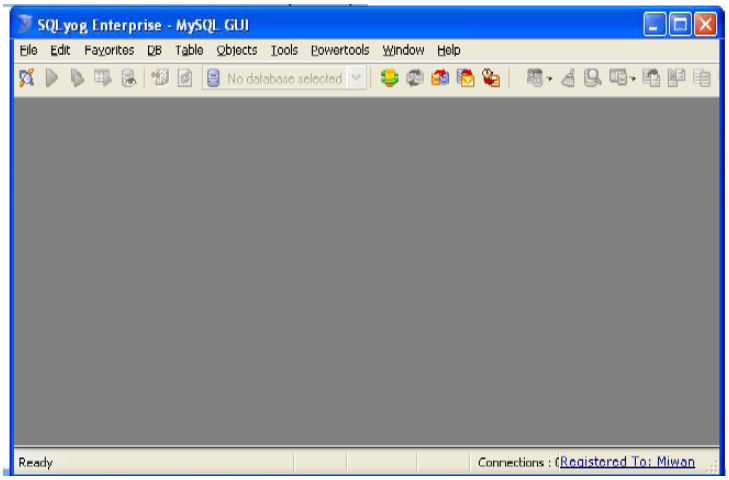




Klik Execute untuk mengeksekusi sinkronisasi

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

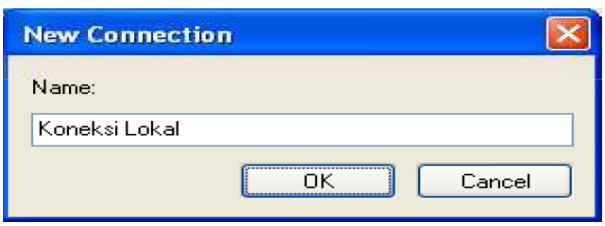




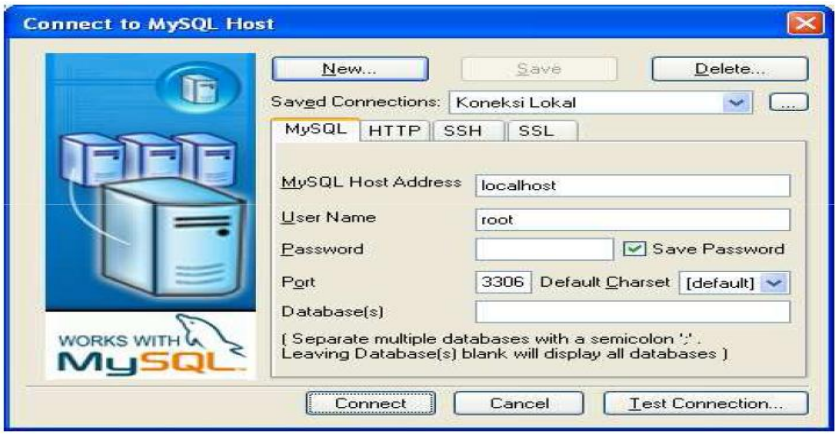
Implementasi Basis Data Menggunakan SQLYog (MySQL GUI)

SQLyog merupakan salah satu perangkat lunak yang berfungsi untuk mengelola database MySQL dengan menggunakan *Graphical User interface* (GUI).

Berikut ini beberapa fungsi yang bisa digunakan pada SQLYog:
Membuat koneksi ke server MySQL
Klik menu **File > New Connection** akan tampil jendela koneksi berikut.



Klik tombol **New** dan akan tampil jendela **New Connection**, isilah nama koneksi kemudian klik **OK** seperti gambar berikut.



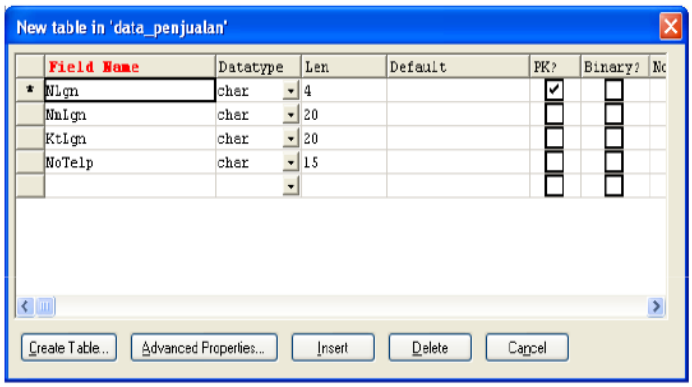
Pastikan Host Address, User Name dan yang lainya diisi dengan benar, kemudian klik **Connect.**

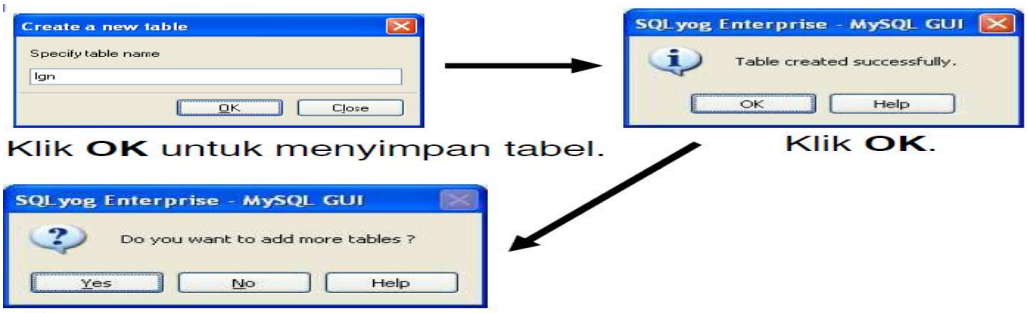
Membuat database

Pilih menu **DB** > **Create Database**, kemudian tentukan nama database (data_penjualan).

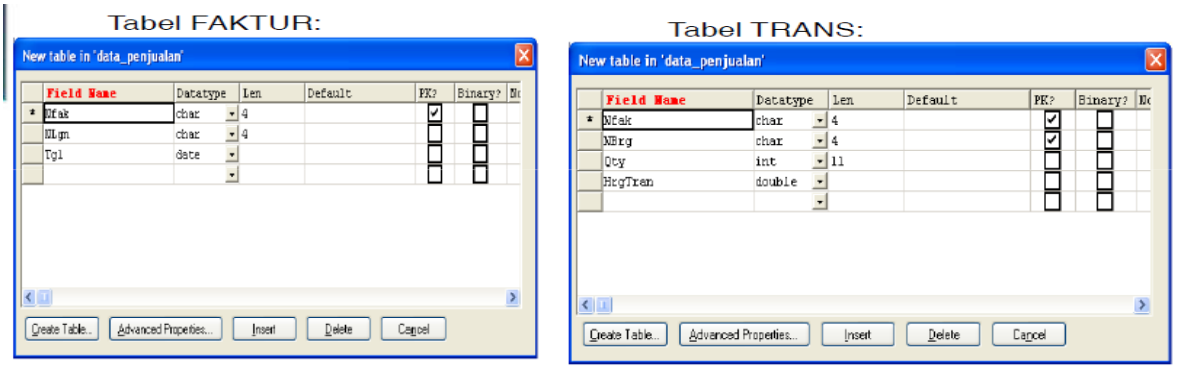
Membuat table

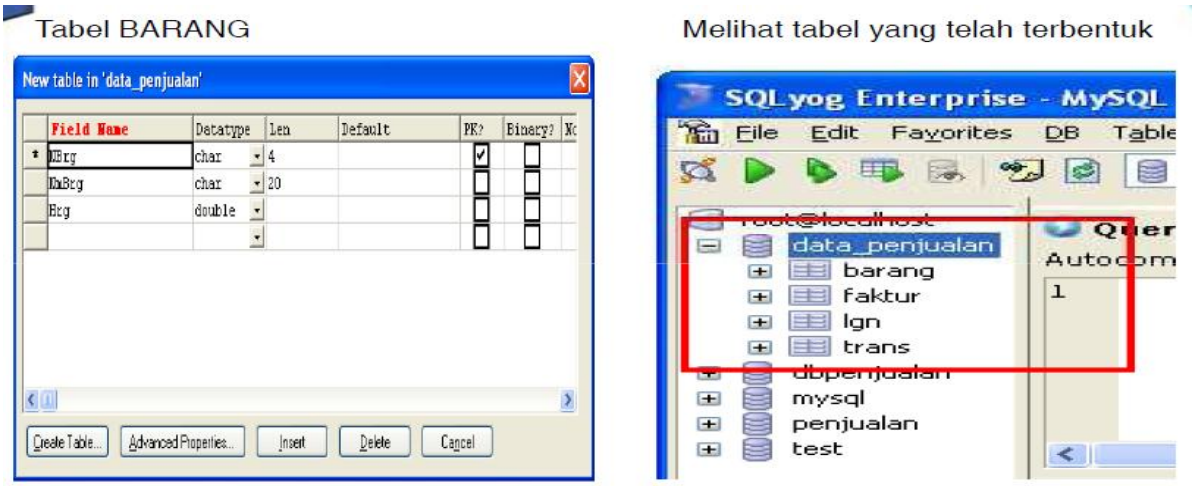
Klik pada database untuk mengaktifkannya. Berikut ini untuk membuat tabel langganan (lgn), pilih menu **DB** > **Create Table**, kemudian tentukan field- fieldnya, kemudian klik **Create Table** dan tentukan nama tabel.

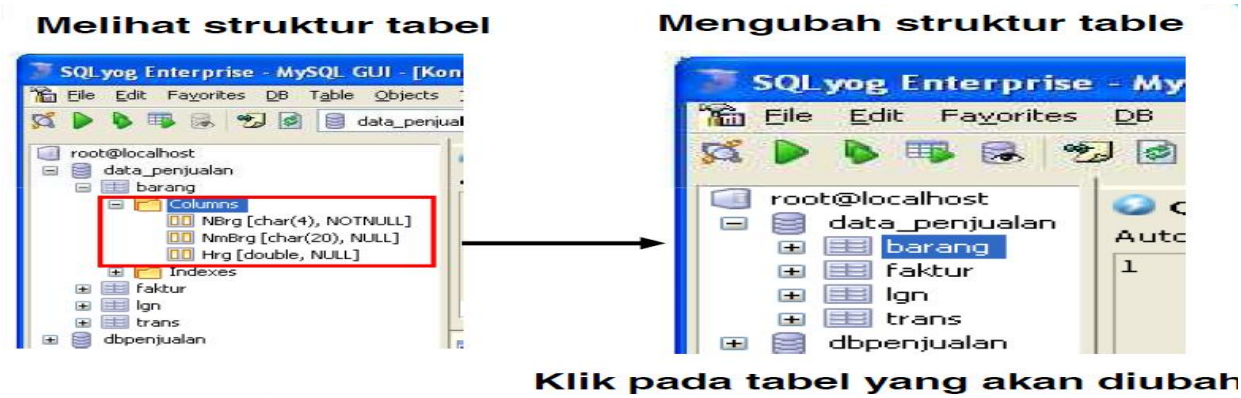




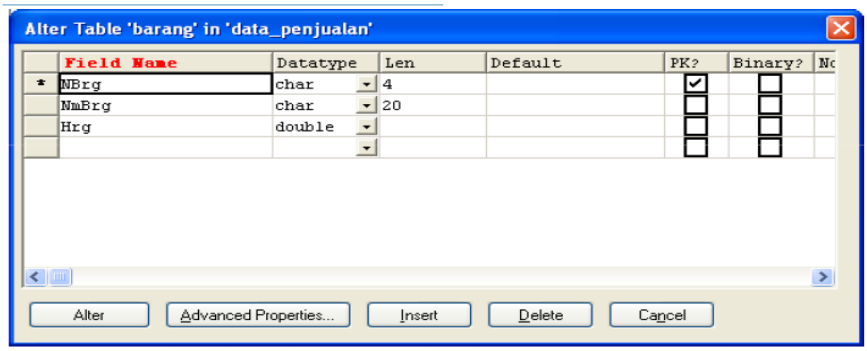
Tambahkan beberapa tabel-tabel yang lain sbb





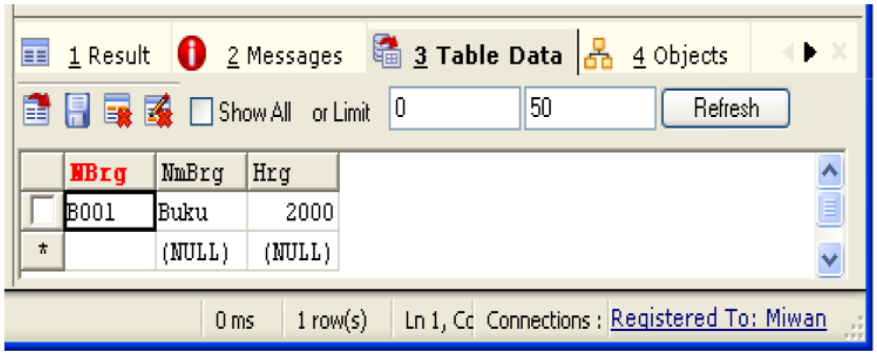


Pilih menu Table > Alter Table, kemudian ubahlah. Jika telah selesai klik Alter



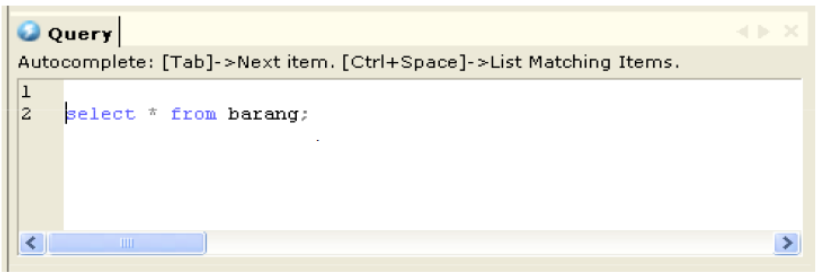
Manipulasi data pada tabel

Untuk menambah, mungubah, menghapus dan melihat data pada tabel bisa dilakukan langsung pada tab Table Data seperti halnya pada MS-Access.



Mengeksekusi perintah SQL

Perintah-perintah SQL bisa dieksekusi dengan cara menuliskannya terlebih dahulu pada tab Query,



kemudian menjalankanya dengan menekan toolbar 

LATIHAN

1. Sebuah perusahaan yang melayani pemesanan barang/produk umum memerlukan sebuah program aplikasi yang berfungsi untuk menyimpan data produk beserta suppliernya dan juga berfungsi untuk mencatat transaksi pemesananan produk dari customer. Setiap produk yang dipesan akan dikirim ke customer yang memesannya. Rancanglah database untuk program aplikasi tersebut dengan menggunakan DBDesigner dan ekspor hasilnya ke server MySQL, untuk memenuhi keinginan perusahaantersebut.
2. Seorang kolektor mobil ingin mendata seluruh mobil miliknya dan memerlukan program aplikasi yang bisa berfungsi untuk menyimpan data koleksi mobilnya. Rancanglah database untuk program aplikasi tersebut dengan menggunakan DBdesigner dan ekspor hasilnya ke server MySQL, sehingga program yang dikembangkan bisa memenuhi keinginan kolektor tersebut.

LINGKUNGAN DATABASE

CONCURRENCY (KONKURENSI)

Ada 3 masalah yang disebabkan oleh Concurrency :

1. Masalah kehilangan modifikasi (*Lost Update Problem*) Masalah ini timbul jika dua transaksi mengakses item database yang sama yang mengakibatkan nilai dari database tersebut menjadi tidak benar.

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

Contoh Lost Update problem

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*

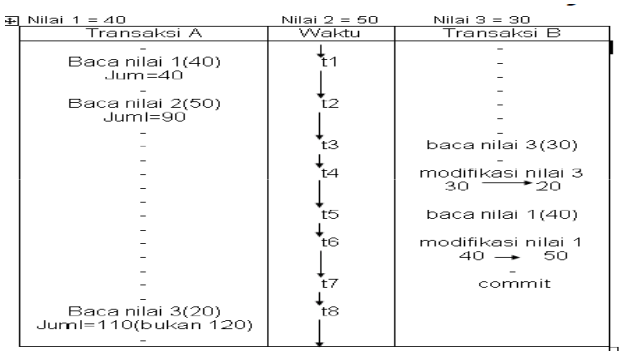
Waktu	Transaksi Simpanan	Transaksi Bunga	Saldo
T1	Read Saldo	.....	1.000.000
T2	Saldo:=saldo+1.000.0000	.....	1.000.000
T3	Write Saldo	.....	2.000.000
T4	.....	Read Saldo	2.000.000
T5	.....	Saldo:= saldo*0.15	2.000.000
T6	.....	Write Saldo	2.300.000
T7	.....	RollBack	2.300.000

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.

Contoh ***Problem of inconsistency Analysis***



Transaksi A menjumlahkan nilai 1, nilai 2 dan nilai 3
Transaksi B _ nilai 1 + 10, nilai 3 –10.

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.

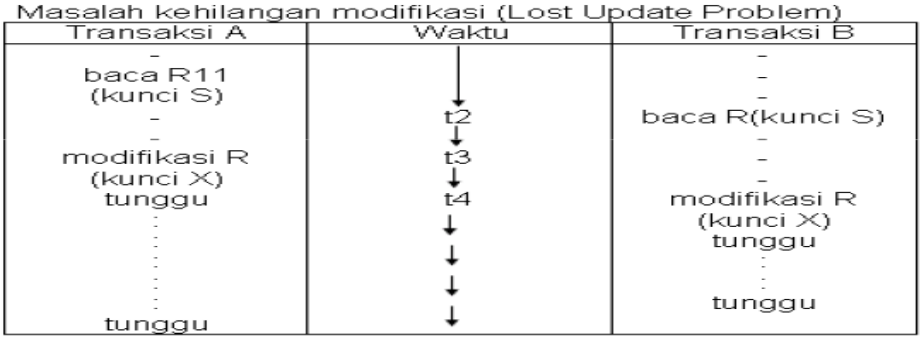
2. Exclusive (X)

   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

•KASUS CARA KERJA LOCKING



Masalah Modifikasi Sementara (uncommited Update Problem)

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

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

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) (kunci S) Jum1=40	t1	-
Baca nilai 2(50) (kunci S) Jum1=90	t2	-
-	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 (kunci S) tunggu	t7	tunggu

TIMESTAMPING

Adalah salah satu alternatif mekanisme control 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.

Pertemuan 14
LINGKUNGAN DATABASE
LANJUTAN

CRASS DAN RECOVERY

PENGERTIAN :

Crash adalah suatu failure atau kegagalam 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
4. sendiri akan memasuki suatu kondisi tidak konsisten.

KLASIFIKASI FAILURE

Berdasarkan Jenis storage

1. Volatile storage, biasanya informasi yang terdapat pada volatile akan hilang, jika terjadi kerusakan system (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 kegagalan :

1. Logical Error, program tidak dapat lagi dilaksanaka 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

SECURITY dan INTEGRITY

SECURITY adalah suatu proteksi data terhadap perusakan data dan pemakaian oleh pemakai yangtidak 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).
4.

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 system operasi yang digunakan dalam jaringan
5. Sistem database, sistem dapat mengatur hak akses user

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 :

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

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