DAFTAR KONTROL MAHASISWA

NPM : _____
NAMA : _____
KELAS : _____

Pertemuan	Tanggal	Hari	Tugas	Keterangan
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

KATA PENGANTAR

Puji syukur saya panjatkan kepada Allah SWT atas limpahan rahmat dan karunia-Nya sehingga buku Panduan Teori dan Praktikum Basis Data dengan menggunakan Microsoft SQL Server 2008 ini dapat terselesaikan tepat waktu.

Buku pedoman praktikum ini merupakan penyempurnaan dari modul praktikum sebelumnya dan diharapkan dengan adanya modul praktikum ini dapat meningkatkan pemahaman lebih lanjut tentang pemrograman serta sebagai pedoman bagi mahasiswa dalam berkreasi untuk mengerjakan studi kasus lainnya. Selain itu, buku pedoman ini juga dapat digunakan sebagai pedomandidalam mengerjakan/membuat aplikasi, khususnya masalah-masalah yang ada di dunia nyata dan diimplementasikan dalam bentuk aplikasi.

Dengan penuh kesadaran, bahwa buku panduan teori dan praktikum ini masih perlu disempurnakan lagi, sehingga saran dan kritik untuk penyajian serta isinya sangat diperlukan.

Akhir kata, saya ucapkan terimakasih kepada seluruh dosen jurusan Sistem Informasi yang turut berpartisipasi dalam penulisan buku panduan praktikum ini. Ucapan terimakasih juga kami sampaikan kepada seluruh pihak yang berpartisipasi sehingga pelaksanaan praktikum ini dapat berjalan dengan lancar.

Lampung, Agustus 2021
Ttd,

Penulis

DAFTAR ISI

INTRODUCTIONS OF DATABASES | CHAPTER 1

Manajemen sistem basis data atau kerennya *Database Management System* (DBMS) adalah kumpulan data yang berhubungan dan memerlukan suatu program untuk mengakses datanya. Kumpulan data umumnya dikenal dengan istilah basis data. Jadi, basis data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, dan dengan *software* untuk melakukan manipulasi data untuk kegunaan tertentu. Basis data biasanya berisi informasi yang relevan dengan perusahaan. Tujuan utama dari DBMS adalah menyediakan suatu cara untuk menyimpan dan mendapatkan informasi kembali dari basis data secara efektif dan efisien.

Sistem basis data didesain untuk mengelola informasi yang besar. Manajemen data menjelaskan dua struktur, yaitu tentang penyimpanan informasi dan menyediakan mekanisme untuk memanipulasi informasi. Sebagai tambahan, sistem basis data harus memastikan bahwa informasi tersimpan, terhindar dari sistem *crash,* dan terlindungi dari akses ilegal. Jika data akan dibagikan ke beberapa pemakai, sistem harus terhindar dari hasil akses yang tidak tepat.

Informasi adalah bagian yang sangat penting bagi suatu organisasi, maka perlu konsep desain dan teknik untuk mengelola data dalam volume yang besar atau bahkan sangat besar. Basis data menyediakan fasilitas atau memudahkan dalam memproduksi informasi yang digunakan oleh pemakai untuk mendukung pengambilan keputusan. Hal inilah yang menjadikan alasan dari penggunaan tehnologi basis data pada saat sekarang (dunia bisnis). Berikut ini contoh penggunaan Aplikasi database dalam dunia bisnis :

1. Bank: Pengelolaan data nasabah, akunting, semua transaksi perbankan
2. Bandara: Pengelolaan data reservasi, penjadualan
3. Universitas: Pengelolaan pendaftaran, alumni
4. Penjualan: Pengelolaan data customer, produk, penjualan
5. Pabrik: Pengelolaan data produksi, persediaan barang, pemesanan, agen
6. Kepegawaian: Pengelolaan data karyawan, gaji, pajak
7. Telekomunikasi : Pengelolaan data tagihan, jumlah pulsa

SISTEM BASIS DATA VS *FILE SYSTEM*

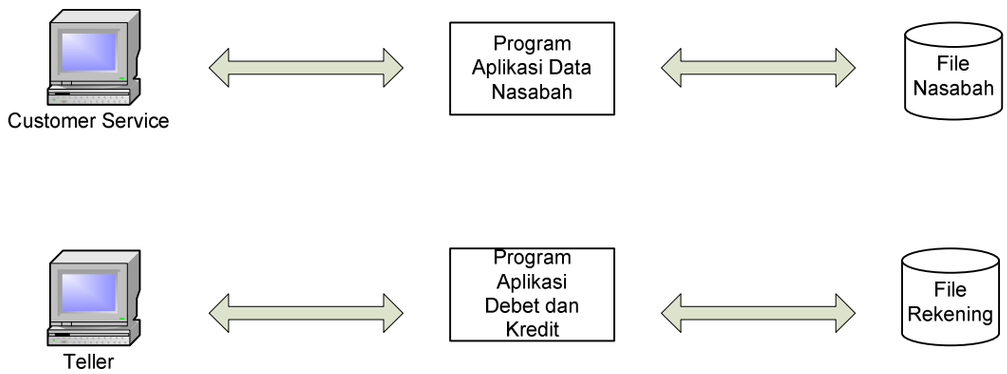
Sebagai bahan pertimbangan, mari kita pelajari bagian dari perusahaan yang memiliki produk seperti tabungan, bagaimana bank menjaga seluruh informasi tentang konsumen dan akun simpanannya. Salah satu cara yang untuk menjaga informasi dalam komputer adalah *file* sistem operasi. Untuk mengizinkan pemakai memanipulasi informasi, sistem memiliki jumlah program aplikasi yang memanipulasi *file*, termasuk :

1. Sebuah program untuk mendebet dan mengkredit akun
2. Sebuah program untuk menambah akun baru
3. Sebuah program untuk menemukan keseimbangan akun
4. Sebuah program untuk menghasilkan laporan bulanan

Programmer sistem menulis program aplikasi ini untuk memenuhi kebutuhan bank. program aplikasi baru ditambahkan ke sistem sebagai kebutuhan. Sebagai contoh, anggaplah bahwa tabungan bank memutuskan untuk menawarkan rekening cek . Hasil dari, bank menciptakan *file* permanen baru yang berisi informasi tentang semua pengecekan rekening dipertahankan di bank,

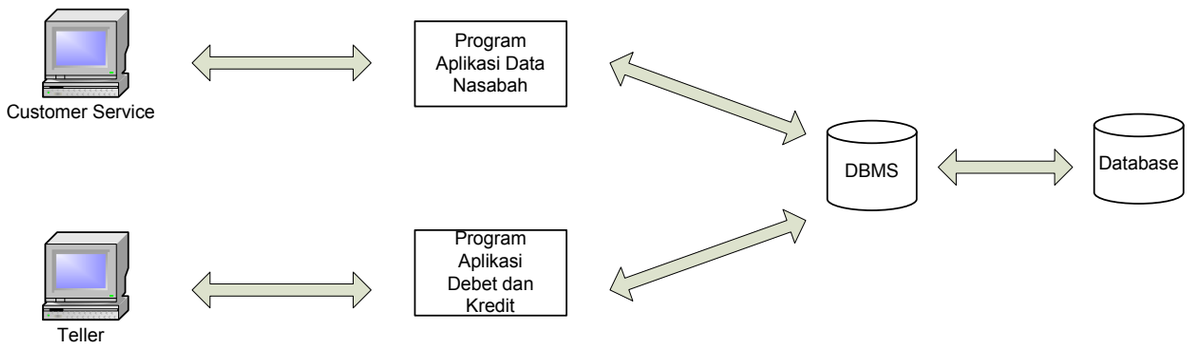
dan mungkin harus menulis program aplikasi baru untuk menghadapi situasi yang tidak muncul di rekening tabungan, seperti cerukan. Demikian, seiring berjalannya waktu, sistem memperoleh lebih banyak *file* dan program aplikasi yang lebih. sistem *file* - pengolahan khas ini didukung oleh sistem operasi konvensional . Sistem ini menyimpan catatan permanen di berbagai *file* , dan perlu berbeda program aplikasi untuk mengambil catatan dari, dan menambahkan catatan untuk, *file* yang sesuai. Sebelum sistem manajemen basis data (DBMS) datang , organisasi biasanya menggunakan informasi yang tersimpan dalam sistem tersebut. Menjaga informasi organisasi dalam sistem *file - processing* memiliki sejumlah kelemahan utama :

1. Redudansi dan inkonsistensi data;
2. Kesulitan dalam mengakses data;
3. Data terisolasi;
4. Masalah integritas data;
5. Masalah atomic data;
6. Anomali data ketika akses bersama-sama;
7. Masalah keamanan;



Gambar 1.1 Pemrosesan secara *File*

Kesulitan-kesulitan ini, antara lain, mendorong pengembangan sistem basis data. Artinya basis data akan memberikan solusi dalam penyelesaian masalah-masalah yang ada.

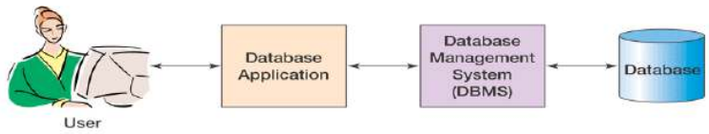


Gambar 1.2 Pemrosesan Secara *Database System*

SISTEM BASIS DATA

Seiring dengan berjalannya waktu lambat laun sistem pemrosesan *file* mulai ditinggalkan karena masih bersifat manual, yang kemudian dikembangkanlah sistem pemrosesan dengan pendekatan *database*.

Sistem Basis Data terdiri dari basis data dan DBMS.



Gambar 1.3 Sistem Basis Data

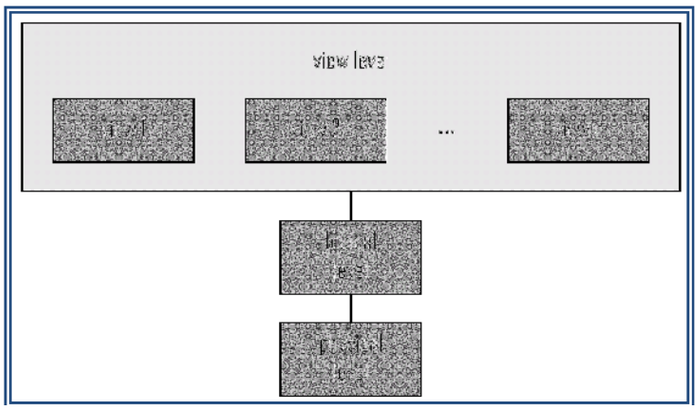
KOMPONEN SISTEM BASIS DATA

Komponen-komponen utama penyusun sistem basis data adalah :

a. Perangkat keras
b. Sistem operasi
c. Basis data
d. Sistem pengelola basis data (DBMS)
e. Pemakai (*Programmer*, *User* mahir, *User* Umum, *User* Khusus)

ABSTRAKSI DATA

1. Sistem basis data biasanya menyembunyikan detil tentang bagaimana data disimpan dan diperlihara. Oleh karena itu, seringkali data yang terlihat oleh pemakai sebenarnya berbeda dengan yang tersimpan secara fisik;
2. Abstraksi data merupakan level dalam bagaimana melihat data dalam sebuah sistem basis data.



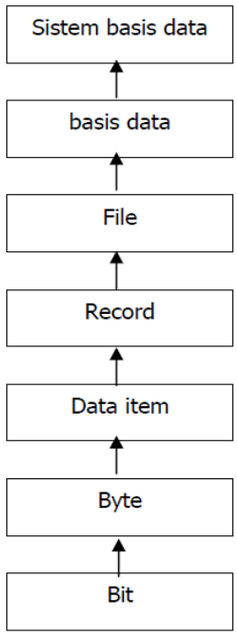
Gambar 1.4 Arsitektur Sistem Basis Data

Penjelasan :

- ***Logical Level*** merupakan pandangan yang berkaitan dengan permasalahan data-data apa saja yang diperlukan untuk disimpan dalam basis data dan penjelasan mengenai hubungan antar data yang satu dengan lainnya. *Conceptual view* dapat disetarakan dengan *schema*, dilakukan *database administrator.*
- ***Physical Level*** merupakan bentuk implementasi dari *conceptual view*, yaitu pandangan tentang bagaimana data disimpan dalam media penyimpan data.
- ***User View*** dapat disejajarkan dengan *sub-schema*.

PENYUSUN SISTEM BASIS DATA

Sistem basis data merupakan lingkup terbesar dalam organisasi data. Sistem basis data mencakup semua bentuk komponen data yang ada dalam suatu sistem. Sedangkan basis data merupakan komponen utama yang menyusun sistem basis data.



Keterangan

- ***Bit***, merupakan sistem angka biner yang terdiri atas angka 0 dan 1
- ***Byte***, merupakan bagian terkecil, dapat berupa karakter numerik, huruf, ataupun karakter khusus yang membentuk suatu item data / field. 1 Byte digunakan untuk mengkodekan 1 karakter
- ***Data Item* (*field*),** merepresentasikan suatu atribut dari suatu record yang menunjukkan suatu item dari data, misalnya nama, alamat. Kumpulan dari field membentuk suatu record
- ***Record***, menggambarkan suatu unit data individu yang tertentu. Kumpulan dari record membentuk suatu *file*.
- ***File***, terdiri dari record-record yang menggambarkan satu kesatuan data yang sejenis
- **Basis data**, sekumpulan dari berbagai macam tipe *record* yang mempunyai hubungan terhadap suatu objek tertentu
- **Sistem basis data**, merupakan sekumpulan basis data, yang tersusun dari beberapa *file*

TIPE *FILE*

Tipe *file* yang digunakan dalam sistem basis data:

a. ***File* induk (*master file*)**
   Ada 2 *file* induk :
   - ***File* induk acuan (*Reference Master File*)**
     - Recordnya relatif statis, jarang berubah nilainya
     - Contoh : *file* daftar gaji, matakuliah
   - ***File* induk dinamik (*Dynamic Master File*)**
     - Nilai dari recordnya sering berubah atau diupdate sebagai hasil suatu transaksi
     - Contoh : *file* stok barang

b. ***File* transaksi (*Transaction file*)**
   Disebut juga *file* input. Digunakan untuk merekam data hasil transaksi.
   Contoh *file* penjualan barang

c. ***File* laporan (*Report File*)**
   Disebut juga *file* output. Berisi informasi sementara yang akan ditampilkan sebagai laporan

d. ***File* Sejarah (*History File*)**
   Disebut juga *file* arsip (*archieval file*). Merupakan *file* yang berisi data masa lalu yang sudah tidk aktif lagi, tapi masih disimpan sebagai arsip

e. ***File* Pelindung (*Backup File*)**
   Merupakan salinan dari *file-file* yang masih aktif di dalam basis data pada saat tertentu. Digunakan sebagai cadangan apabila *file* basis data yang aktf mengalami kerusakan atau hilang

BAHASA BASIS DATA

Bahasa basis data merupakan perantara bagi pemakai dengan basis data dalam berinteraksi, yang telah ditetapkan oleh pembuat DBMS. Dapat dibedakan menjadi 2, yaitu :

1. ***Data Definition Language* (DDL)**
   Dengan bahasa ini kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur tabel, dll. Hasil dari kompilasi perintah DDL menjadi Kamus Data,

yaitu data yang menjelaskan data sesungguhnya. Contoh : *Create, Modify Report, Modify Structure*.

2. ***Data Manipulation Language*** **(DML)**
   Berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data, yang berupa *insert, update, delete*, dll. Ada dua jenis, yaitu prosedural (ditentukan data yang diinginkan dan cara mendapatkannya) dan non-prosedural (tanpa menyebutkan cara mendapatkannya. Contoh : Dbase 3+, FoxBase, SQL, QBE.

KONSEP DASAR BASIS DATA

Data adalah Representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, mahasiswa, pembeli), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentu angka, huruf, symbol,teks, gambar, bunyi atau kombinasinya.

ISTILAH - ISTILAH DASAR BASIS DATA

Enterprise
Suatu bentuk organisasi seperti: bank, universitas, rumah sakit, pabrik, dsb. Data yang disimpan dalai basis data merupakan data operasional dari suatu enterprise.
Contoh data operasional : data keuangan, data mahasiswa, data pasien

Entitas
Suatu obyek yang dapat dibedakan dari lainnya yang dapat diwujudkan dalai basis data. Contoh **Entitas** dalam **lingkungan bank** terdiri dari : **Nasabah**, **Simpanan**, **Hipotik**. Contoh **Entitas** dalam **lingkungan universitas** terdiri dari : Mahasiswa, mata kuliah. Kumpulan dari entitas disebut **Himpunan Entitas.**
Contoh: semua nasabah, semua mahasiswa

Atribut (Elemen Data)
Karakteristik dari suatu entitas.
Contoh: Entitas Mahasiswa **atribut**nya terdiri dari **Npm, Nama, Alamat, Tanggal lahir.**

Nilai Data (*Data Value*)
Isi data / informasi yang tercakup dalai setiap elemen data.
Contoh Atribut Nama Mahasiswa dapat berisi Nilai Data : **Hendra, Kiki, Kirana, Keola**

Kunci Elemen Data (*Key Data Element*)
Tanda pengenal yang secara unik mengidentifikasikan entitas dari suatu kumpulan entitas.
Contoh Entitas Mahasiswa yang mempunyai atribut-atribut npm, nama, alamat, tanggal lahir menggunakan Kunci Elemen Data npm.

Record Data
Kumpulan isi elemen data yang saling berhubungan.
Contoh : kumpulan atribut npm, nama, alamat, tanggal lahir dari Entitas Mahasiswa berisikan : "03050006", "Hendra Kurniawan", "Kota Metro - Lampung", "25 Agustus 1983".

KEUNTUNGAN SISTEM BASIS DATA

1. **Terkontrolnya kerangkapan data**
   Dalam basis data hanya mencantumkan satu kali saja field yang sama yang dapat dipakaioleh semua aplikasi yang memerlukannya.
2. **Terpeliharanya keselarasan (ke-konsistenan) data**
   Apabila ada perubahan data pada aplikasi yang berbeda maka secara otomatis perubahan itu berlaku untuk keseluruhan
3. **Data dapat dipakai secara bersama (shared)**

Data dapat dipakai secara bersama-sama oleh beberapa program aplikasi (secara *batch* maupun *on-line*) pada saat bersamaan.

4. **Dapat diterapkan standarisasi**
   Dengan adanya pengontrolan yang terpusat maka DBA dapat menerapkan standarisasi data yang disimpan sehingga memudahkan pemakaian, pengiriman maupun pertukaran data.
5. **Keamanan data terjamin**
   DBA dapat memberikan batasan-batasan pengaksesan data, misalnya dengan memberikan *password* dan pemberian hak akses bagi user (misalnya: *modify, delete, insert, retrieve*).
6. **Terpeliharanya integritas data**
   Jika kerangkapan data dikontrol dan ke konsistenan data dapat dijaga maka data menjadi Akurat.
7. **Terpeliharanya keseimbangan (keselarasan) antara kebutuhan data yang berbeda dalam setiap aplikasi**
   Struktur basis data diatur sedemikian rupa sehingga dapat melayani pengaksesan data dengan cepat.
8. ***Data Independence* (kemandirian data)**
   Dapat digunakan untuk bermacam-macam program aplikasi tanpa harus merubah format data yang sudah ada.

KELEMAHAN SISTEM BASIS DATA

1. Memerlukan tenaga spesialis
2. Kompleks
3. Memerlukan tempat yang besar
4. Mahal

PENGGUNA BASIS DATA

System Engineer

Tenaga ahli yang bertanggung jawab atas pemasangan Sistem Basis Data, dan juga mengadakan peningkatan dan melaporkan kesalahan dari sistem tersebut kepada pihak penjual.

Database Administrator (DBA)

Tenaga ahli yang mempunyai tugas untuk mengontrol sistem basis data secara keseluruhan, meramalkan kebutuhan akan sistem basis data, merencanakannya dan mengaturnya. Tugas DBA:

- ✓ Mengontrol DBMS dan *software-software*
- ✓ Memonitor siapa yang mengakses basis data
- ✓ Mengatur pemakaian basis data
- ✓ Memeriksa *security, integrity*, *recovery* dan *concurency*

Program Utility yang digunakan oleh DBA:

- *Loading Routines*
  Membangun versi utama dari *database*
- Reorganization *Routines*
  Mengatur / mengorganisasikan kembali *database*
- *Journaling Routines*
  Mencatat semua operasi pemakaian *database*
- *Recovery Routines*
  Menempatkan kembali data, sebelum terjadinya kerusakan
- *Statistical Analysis Routines*
  Membantu memonitor kehandalan sistem

PEMAKAI AKHIR

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

Programmer aplikasi

Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML), yang disertakan (*embedded*) dalam program yang ditulis dalam bahasa pemrograman induk (seperti C, pascal, cobol, dll).

User Mahir (*Casual User*)

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

User Umum (*End User/Naïve User*)

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

User Khusus (*Specialized/Sophisticated User*)

Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus seperti aplikasi AI, Sistem Pakar, Pengolahan Citra,dll, yang bisa saja mengakses basis data dengan/tanpa DBMS yang bersangkutan.

LATIHAN SOAL

a. Apa perbedaan DBMS vs Basis Data ? **versi anda** bukan **google**
b. Jelaskan arti data dan informasi **versi anda** bukan **google** !
c. Jelaskan apa perbedaan antara data vs informasi **versi anda** bukan **google** !
d. Jelaskan pengertian dasar dari basis data **versi anda** bukan **google** !
e. Jelaskan perbedaan antara basis data dan sistem basis data **versi anda** bukan **google** !
f. Sebutkan faktor yang melatar belakangi peralihan penggunaan sistem *file* menuju sistem basis data ? **versi anda** bukan **google**
g. Apa kelebihan dan kelemahan penggunaan sistem basis data **versi anda** bukan **google** ?
h. Kapan basis data diperlukan ? Jelaskan ! **versi anda** bukan **google**
i. Jelaskan tiga level dalam abstraksi data **versi anda** bukan **google** !

***) DIKUMPUL SATU MINGGU SETELAH PERTEMUAN INI !!!**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA MODEL

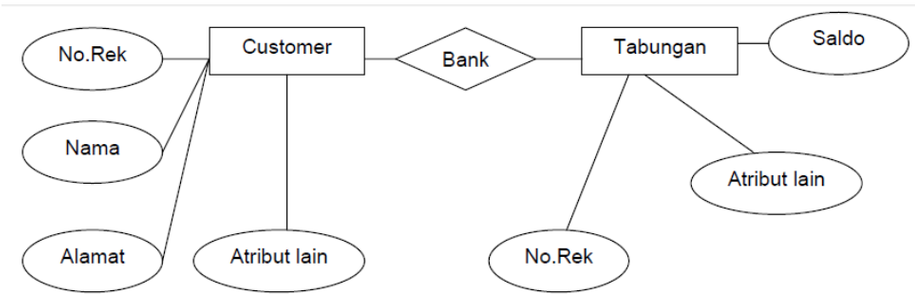
CHAPTER 2

MODEL DATA

Model data merupakan suatu cara untuk menjelaskan bagaimana pemakai dapat melihat data secara logik. Ada 3 (tiga) jenis model data, yaitu :

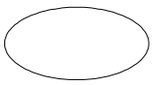
a. Model Data Berbasis Objek
Merupakan himpunan data dan relasi yang menjelaskan hubungan logik antar data dalam suatu basis data berdasarkan objek datanya. Terdiri dari 2 (dua) jenis, yaitu :

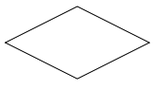
- **Entity Relational Model**
  Merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan (dunia nyata) terdiri dari objek-objek dasar yang mempunyai hubungan/relasi antar objek tersebut. Contoh :

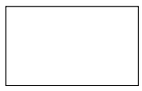


Gambar 2.1 Entity Relational Model

Arti Simbol :

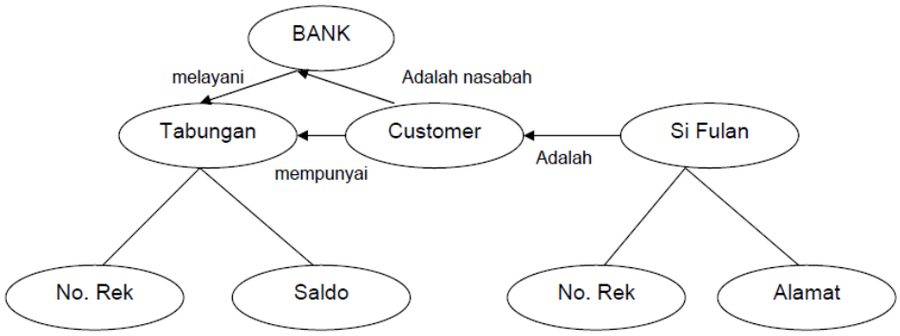
 ➔ Objek Dasar

 ➔ Objek Dasar

 ➔ Objek Dasar

 ➔ Objek Dasar

- **Semantik Model**
  Relasi antar objek dinyatakan dengan kata-kata (semantic). Contoh :



Gambar 2.2 Semantic Model

Arti Simbol:

➔ Menunjuk pada Atribut
➔ Menunjuk pada adanya relasi/hubungan

b. Model Data Berbasis Record

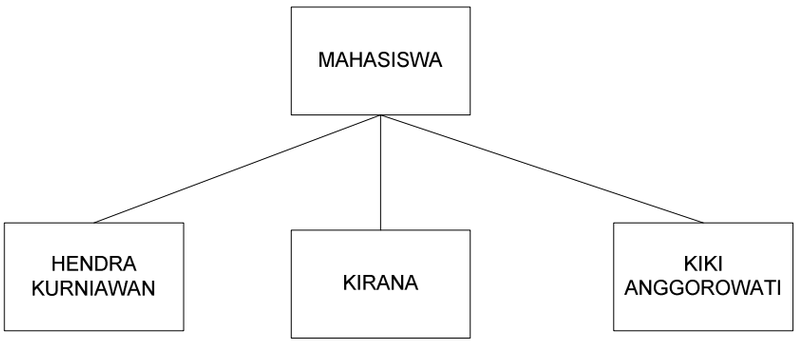
Model ini mendasarkan pada record untuk menjelaskan kepada *user* tentang hubungan logik antar data dalam basis data. Ada 3 (tiga) jenis, yaitu :

- **Relational Model**
  Menjelaskan hubungan secara logik antar data dalam basis data dengan memvisualisasikan kedalam bentuk tabel-tabel yang terdiri dari baris dan kolom yang menunjuk pada atribut dan nilai data. Kelebihan dari model ini adalah lebih mudah untuk dipahami daripada model-model lainnya.
  Contoh :

NPM	**Nama**
03050006	Hendra Kurniawan
03050036	Kiki Anggorowati
03050023	Kirana

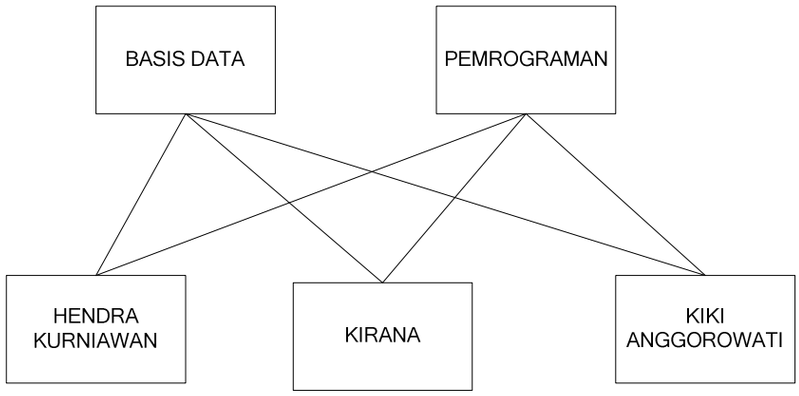
Penjelasan :
- Jumlah kolom disebut ***degree***/derajat
- Baris disebut sebagai **atribut**
- Tiap baris disebut sebagai ***record/tuple***
- Banyaknya baris dalam suatu tabel disebut ***cardinality***

- **Hierarchy Model (Model Hirarki)**
  Menjelaskan hubungan secara logik antar data dalam basis data dalam bentuk hubungan bertingkat (hirarki). Elemen penyusunnya disebut sebagai ***node,*** yang berupa rinci data, agregate data, atau *record*. Contoh:



Gambar 2.3 Hirarki Model

- ***Network Model (Plex Structure)***
  Model ini bisa dikatakan hampir sama dengan hirarki model, dan digambarkan seperti hirarki model namun posisi dari ***Child*** lebih rendah dari ***Parent*****,** dan ***Child*** bisa memiliki lebih dari **Satu *Parent*.**



Gambar 2.4 *Network Model*

c. Model Data Berbasis Fisik
Digunakan untuk menjelaskan kepada pemakai bagaidalam basis data disimpan dalam media penyimpanan secara fisik, yang lebih berorientasi pada mesin. Ada 2 (dua) model, yaitu:

- *Unifying Model*
- *Frame Memory*

LATIHAN SOAL

a. Sebutkan model-model data ! Berikan contoh satu persatu !
b. Apa kelebihan dari model berbasis objek, fisik ? **versi anda** bukan **google**
c. Apa perbedaan dari model berbasis objek, fisik ? **versi anda** bukan **google**
d. Berikan contoh yang lain dan gambarkan contoh tersebut untuk model data berbasis objek dan model data berbasis record ! **versi anda** bukan **google**
e. Berikan contoh kasus dan gambarkan dengan network model ! **versi anda** bukan **google**

***) DIKUMPUL SATU MINGGU SETELAH PERTEMUAN INI !!!**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

RELATIONAL DATABASE MODEL CHAPTER 3

Model ini menjelaskan tentang hubungan logik antar data dala basis data dengan cara memvisualisasikan kedalam tabel bentuk dua dimensi yang terdiri dari baris dan kolom yang menunjukkan atribut. Istilah-istilah dalam model basis data relasional:

a. ***Record*** : Sebuah baris dalam suatu relasi. Disebut juga *tuple*
b. ***Cardinality*** : Banyaknya record dalam sebuah relasi
c. ***Atribut*** : Suatu kolom dalam sebuah relasi
d. ***Domain*** : Batasan nilai dalam atribut dan tipe datanya
e. ***Derajat / degree*** : Banyaknya kolom dalam relasi
f. ***Candidate Key*** : Atribut atau sekumpulan atribut yang unik yang dapatdigunakan untuk membedakan suatu record
g. Primary Key : Salah satu CK yang dipilih dan dijadikan key untuk menbedakan record satu dan yang lainnya.
h. Foreign Key : PK pada tabel induk yang berelasi pada tabel lain sebagai referensial.
i. ***Alternate Key*** : CK yang dipilih menjadi PK
j. ***Unary Relation*** : Suatu relasi yang hanya mempunyai satu kolom
k. ***Binary Relation*** : Suatu relasi yang hanya mempunyai dua kolom
l. ***Ternary Relation*** : Suatu relasi yang hanya mempunyai tiga kolom

KARAKTERISITIK MODEL BASIS DATA RELASIONAL

Relasi dalam model basis data relasional memiliki karakteristik sebagai berikut:

a. Semua ***entry /*** **elemen data** pada suatu baris dan kolom tertentu harus mempunyai nilai tunggal (***single value***), atau suatu nilai yang tidak dapat dibagi lagi (***atomic value)***, bukan suatu kelompok pengulangan.
b. Semua ***entry /*** **elemen data** pada suatu baris dan kolom tertentu dalam relasi yang sama harus mempunyai jenis yang sama.
c. Masing-masing kolom dalam suatu relasi mempunyai nama yang unik.
d. Pada suatu relasi / tabel yang sama tidak ada dua baris yang identik.

KOMPONEN RELASI

Tabel relasional mempunyai 2 (dua) komponen, yaitu :

1. **Intention**
   - Terdiri dari dua bagian, yaitu struktur penamaan (***naming structure***) dan batasan integritas (***integrity constraint).***
   - Struktur penamaan menunjukkan nama tabel dan nama atribut n nilai dan tipe datanya
   - Batasan integritas dipengaruhi oleh integritas referential yang meliputi key constraint dan referensial constraint.
   - Key contstraint tidak mengizinkan adanya nilai null pada atribut yang digunakan sebagai PK
   - Refential constraint memberikan aturan bahwa nilai-nilai dalam atribut kunci yang digunakan untuk menghubungkan ke basis data lain tidak diijinkan memiliki nilai null.

2. **Extention**
   Menunjukkan isi dari tabel-tabel pada suatu waktu, cenderung berubah sewaktu-waktu.

KUNCI RELASI

Dasar penentuan PK adalah bahwa nilai-nilai data dari atribut yang digunakan sebagai PK harus unik, tidak mungkin ada nilai rinci data yang sama pada semua record dalam basis data. Aturan lainnya:

- **Integritas Entity**
  - Nilai atribut yang dipilih sebagai PK tidakboleh null untuk setiap record yang ada dalam relasi;
  - Aturan ini menjamin bahwa semua record yang ada dalam basis data akan dapat diakses karena semua record dapat diidentifikasi berdasarkan kunci yang unik.
  - Contoh :

NPM *)	**Nama**	**JenisKelamin**
03050006	Hendra Kurniawan	Male
03050036	Kiki Anggorowati	Female
03050026	Kirana	Female

Ket : *) Menunjukkan ***Primary Key (PK)***

- **Integritas Referensial**
  - Jika dua buah tabel direlasikan maka PK harus menjamin bahwa untuk setiap nilai PK tertentu dalam tabel A, harus pula ada record dengan nilai PK yang sama pada tabel B.
  - Contoh:

TblMahasiswa

NPM *)	**Nama**	**JenisKelamin**
03050006	Hendra Kurniawan	Male
03050036	Kiki Anggorowati	Female
03050026	Kirana	Female

TblKRS

NPM *)	**JmlMK**	**JmlSKS**
03050006	7	21
03050036	8	24
03050046	4	16

03050046 → tidak terdapat pada **TblMahasiswa ::** Integritas Referensial tidak terpenuhi

RELASI ANTAR ENTITY

Ada dua jenis relasi antar *entity*, yaitu :

1. **Relasi antar *entity* dalam satu tabel**
   Berupa relasi antar entity yang berupa record untuk menyediakan data dan informasi dari atribut-atribut dalam satu tabel. Contoh: NPM → **03050006** bernama **Hendra Kurniawan** dan ber jenis kelamin ***Male.***

2. **Relasi antar *entity* dalam banyak tabel**
   Tipe ini memiliki tingkat relasi yang lebih rumit. Ada tiga jenis relasi, yaitu, ***Tree, Simple Network, Complex Network.***

BAHASA RELATIONAL MODEL

Menggunakan bahasa *query* → pernyataan yang diajukan untuk mengambil informasi. Bahasa *Query* (*Query Language*) lebih ditekankan pada aspek pencarian data dari dalam tabel. Aspek pencarian ini sedemikian penting karena merupakan inti dari upaya untuk pengelolaan data. Bahasa *query* terbagi 2:

1. **Bahasa Formal**
   Bahasa *query* yang diterjemahkan dengan menggunakan simbol-simbol matematis. Contoh:
   - **Aljabar Relasional**
     Bahasa query prosedural → pemakai menspesifikasikan data apa yang dibutuhkan dan bagaimana untuk mendapatkannya.
   - **Kalkulus Relasional**
     Bahasa query non-prosedural pemakai menspesifikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana untuk mendapatkannya. Terbagi 2:
     1. Kalkulus Relasional Tupel
     2. Kalkulus Relasional Domain

2. **Bahasa Komersil**
   Bahasa Query yang dirancang sendiri oleh programmer menjadi suatu program aplikasi agar pemakai lebih mudah menggunakannya (*user friendly*). Contoh:

   - **QUEL**
     Berbasis pada bahasa kalkulus relasional;
   - **QBE**
     Berbasis pada bahasa kalkulus relasional;
   - **SQL**
     Berbasis pada bahasa kalkulus relasional dan aljabar relasional;

Contoh-contoh Basis Data Relasional:

- DB2 → IBM
- ORACLE → Oracle
- SYBASE → Powersoft
- INFORMIX → Informix
- Microsoft Access → Microsoft

Pada chapter ini, kita akan mempelajari bahasa **aljabar relasional model.**

ALJABAR RELASIONAL

Operasi – Operasi Dasar pada aljabar relasional, yaitu

1. Select
2. Project
3. Cartesian Product
4. Union
5. Set Defference

Operasi – Operasi Tambahan

1. Natural Join
2. Theta Join
3. Intersection
4. Division

SELECT OPERATION

Untuk memilih baris tertentu dari sebuah himpunan baris data (record) yang memenuhi kondisi dan membuang baris yang lain. Notasi:

$$\sigma_{<\text{kondisi pilihan}>}(<\text{nama relasi}>)$$

CONTOH :

- σ**dep_nomor=4**(PEGAWAI)
  untuk memilih sub himpunan pegawai yang bekerja departemen nomor 4
- σ**gaji>30000 AND dep_nomor=3** (PEGAWAI)
  Untuk memilih sub himpunan pegawai yang memiliki gaji lebih dari 30000 yang bekerja di departemen 3

PROJECT OPERATION

Untuk memilih attribut (kolom) tertentu dari himpunan / subhimpunan dan membuang yang lain.
Notasi:

$$\pi_{<\text{daftar attribut}>}(<\text{nama relasi}>)$$

Contoh:
Untuk memilih atribut JenisKel dan Gaji dari tabel Pegawai:

$$\pi_{\text{JenisKel, Gaji}}(\text{Pegawai})$$

URUTAN OPERASI ALJABAR RELASIONAL

Sebuah operasi bisa dituliskan dalam bentuk beberapa ekspresi aljabar relasional dengan mengelompokkan untuk tiap-tiap operasi dan memberi nama.

$$\pi_{\text{nmDepan, nmBlk, gaji}}(\sigma\text{dep\_nomor=5(PEGAWAI)})$$

Penjelasan:
Mengambil informasi nama depan, nama belakang dan gaji dari pegawai-pegawai yang bekerja di departemen nomor 5.

UNION OPERATION

Untuk menghimpun relasi dari dua/lebih himpunan relasi. Notasi :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$$R \cup S$$

A	B
α	1
α	2
β	1
β	3

SET DIFFERENCE OPERATION

Untuk menghimpun dari dua/lebih himpunan relasi dimana nilai himpunan A tidak sama dengan himpunan B. Notasi:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

R – S

A	B
α	1
β	1

CARTESIAN PRODUCT OPERATION

Membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi tupelo-tupel yang mungkin. Notasi :

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

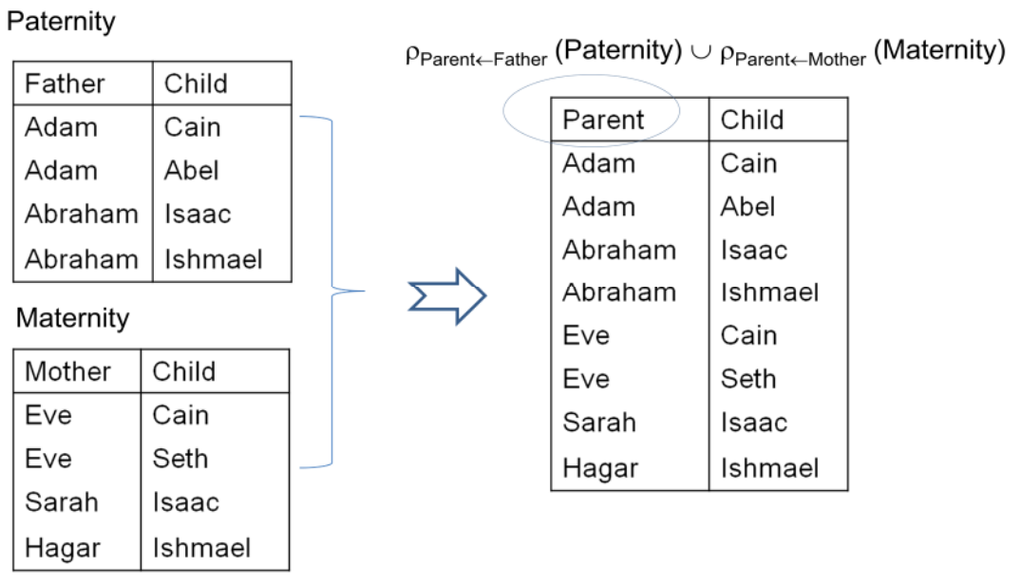
s

R X S

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

RENAME OPERATION

Untuk merubah atribut pada suatu relasi. Notasi : ←

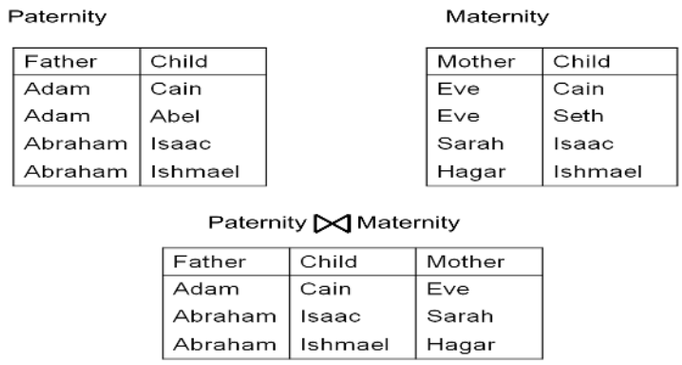


JOIN

Membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi yang mungkin dari relasi–relasi. Beberapa operasi dasar pada JOIN.

1. Set Intersection ∩
2. Natural Join ⋈
3. Aggregation *g*
4. Outer Join (Left Outer Join(⟕), Right Outer Join (⟖), Full Outer Join (⟗)
5. Division ÷

NATURAL JOIN



OUTER JOIN

Employees

Employee	Department
Smith	sales
Black	production
White	production

Departments

Department	Head
production	Mori
purchasing	Brown

Employees ⟕ LEFT Departments

Employee	Department	Head
Smith	sales	NULL
Black	production	Mori
White	production	Mori

Employees ⟖ RIGHT Departments

Employee	Department	Head
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

Employees ⟗ FULL Departments

Employee	Department	Head
Smith	sales	NULL
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

DIVISION

Untuk memndapatkan nilai yang ada pada salah satu atribut dari relasi ‘ pembilang ‘ yang nilai atributnya sama dengan nilai atribut relasi ‘ penyebut ‘

ACCOUNT

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

BRANCH

branch-name	*branch-city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

DEPOSITOR

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Dari 3 tabel diatas, kita akan melakukan operasi division, sebagai contoh seperti berikut :

$$\Pi_{\text{customer-name,branch-name}}(\text{DEPOSITOR} \bowtie \text{ACCOUNT}) \div \Pi_{\text{branch-name}}(\sigma_{\text{branch-city=”Brooklyn”}}(\text{BRANCH}))$$

Hasilnya adalah : **JOHNSON**

LATIHAN SOAL

a. Bagaimana cara menjelaskan istilah-istilah ini, agar anda pribadi memahami dengan baik tidak secara **text book.**
   - *Record* - Key - FK - AK
   - Atribut - P K - CK - Relasi

b. Tentukan atribut-atribut yang diperlukan untuk file-file pembentuk basis data berikut:
   - Perpustakaan (Anggota, Buku, Peminjaman, Pengembalian)
   - Koperasi Simpan Pinjam (Angota, Simpanan, Pinjaman)
   - Penjualan (Customer, Barang, Penjualan)

c. Berdasarkan penentuan atribut yang telah dibuat, tentukan **Primary Key**

d. Buatlah operasi aljabar relasional berdasarkan kasus dibawah ini:

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

- *Set Projection number, age from Graduates;*
- *Set Selection which Age<=35 from managers;*
- *Set Union between Graduates and Manager which Age >35;*
- *Use left outer join to finish Graduates and Managers;*
- *Use Right outer join to finish Graduates and Managers;*
- *Use full outer join to finish Graduates and Managers;*
- *Set Intersection by two operand (Graduates and Managers);*
- *Set difference by two operand;*

***) DIKUMPUL SATU MINGGU SETELAH PERTEMUAN INI !!!**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu : Nilai

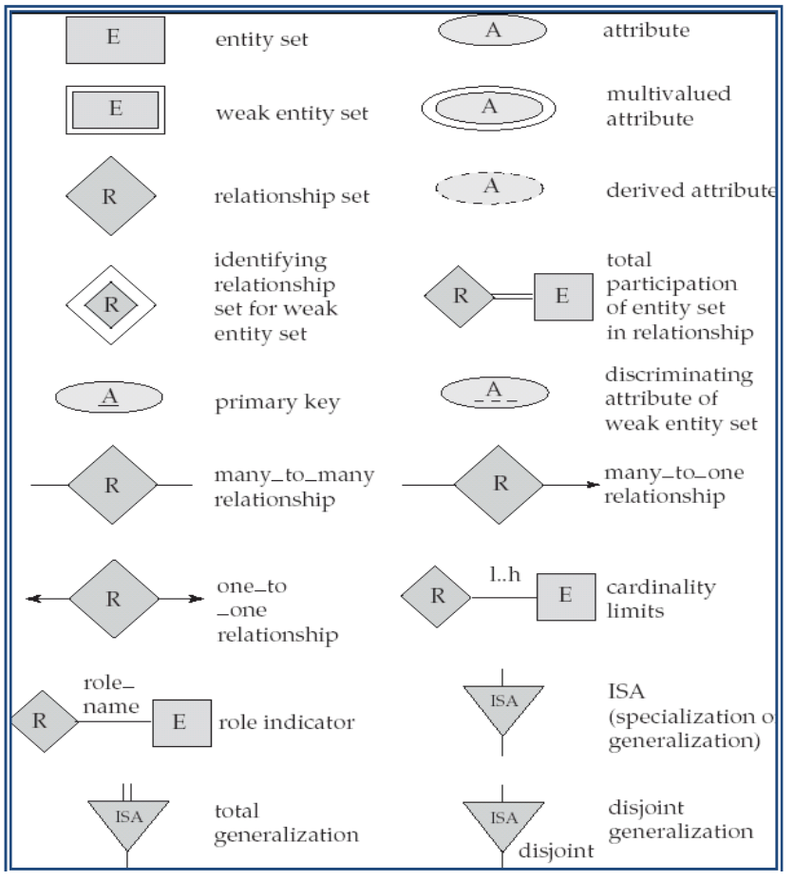
(……..)
NIK. _____

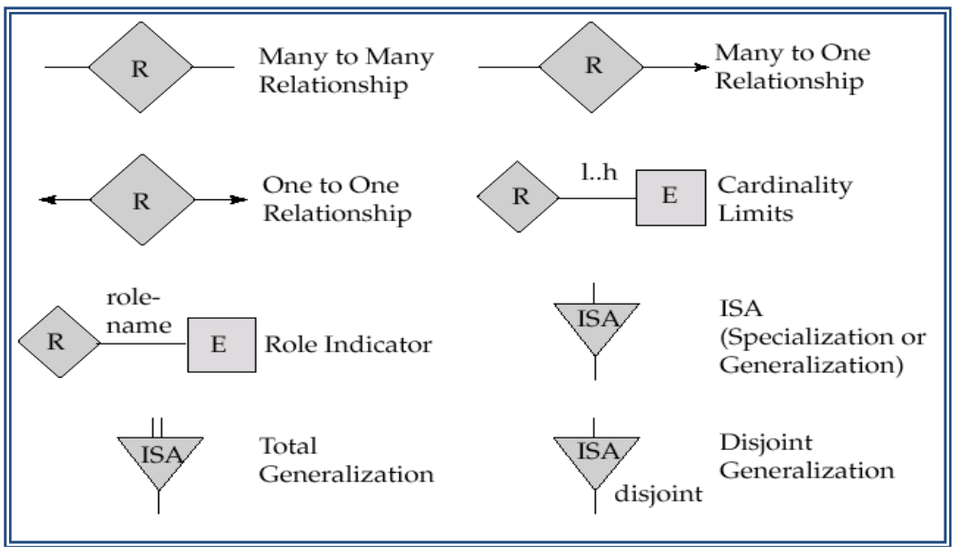
ENTITY RELATIONSHIP DIAGRAM (ERD)

CHAPTER 4

Entity Relationship Diagram (ER-D) merupakan notasi grafis dalam pemodelan data konseptuan yang menggambarkan hubungan / relasi antar entitas dalam penyimpanan data. ER-D digunakan untuk memodelkan struktur data dan hubungan antar data, karena hubungan ini relatif kompleks. ER-D kebanyakan digunakan untuk menjawab bentuk model hubungan data pada kasus-kasus pengembangan sistem.

SIMBOL / NOTASI

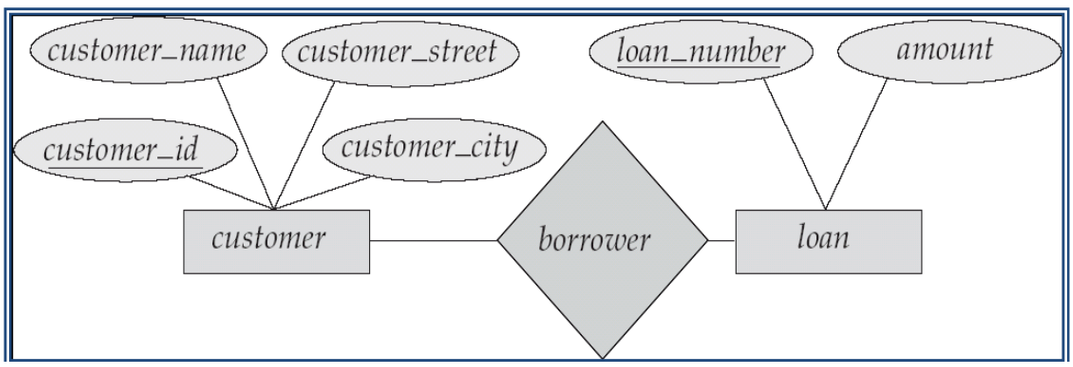




Gambar 4.1 Notasi ER-D

NOTASI DAN ARTINYA

1. ***Entity*****,** objek atau benda dalam dunia nyata yang dapat diidentifikasi atau ditemukan di lingkungan pemakai. Contoh: mahasiswa, pelanggan, rekening. *Entity* digambarkan dengan **PERSEGI PANJANG**.
2. ***Entity Set*****,** kumpulan dari *entity* yang sejenis. Contoh: orang/manusia pada lingkungan perguruan tinggi adalah ***entity* set** dari **Mahasiswa**.
3. **Atribut,** karakterisitik dari suatu *entity*/entitas. Contoh: **atribut NPM** dari ***entity* Mahasiswa.** Atribut digambarkan dengan **ELIPS.**
4. ***Relationship*****,** hubungan yang terjadi antara satu *entity* atau lebih.
5. ***Relationship Set*****,** kumpulan relationship yang sejenis. Atribut digambarkan dengan **GARIS.**



Gambar 4.1 ER-Diagram

ATRIBUT

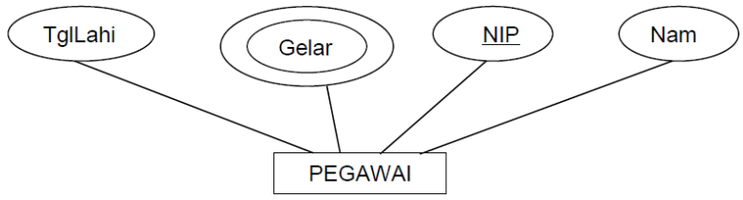
- Atribut adalah karakteristik dari *entity* atau relationship, yang menyediakan penjelasan detail tentang *entity* atau relationship tersebut.
- Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu *entity* atau *relationship*.

Jenis-Jenis Atribut:

1. ***Key***
   Atribut yang digunakan untuk menentukan suatu *entity* secara unik. Atribut *key* ditandai dengan ***Underscore.***

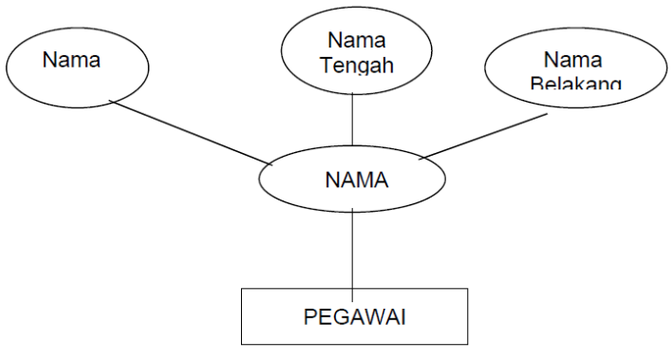
2. ***Simple Atributte***
   Atribut yang bernilai tunggal.

3. ***Multivalue Atributte***
   Atribut yang memiliki sekelompok nilai untuk setiap instan *entity*. Atribut multivalue digambarkan dengan **Elips Ganda.**



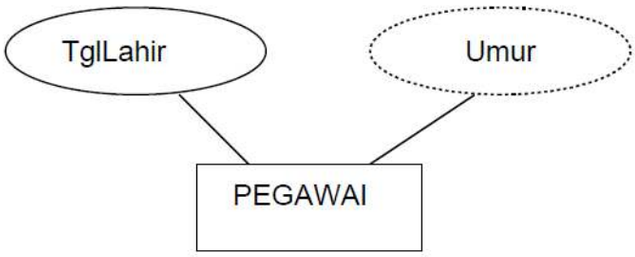
Gambar 4.2 *Multivalue Atribute*

4. ***Composite Atributte***
   Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.



Gambar 4.3 *Composite Atribut*

5. ***Derivatif Atributte***
   Suatu atribut yang dihasilkan dari atribut yang lain. Untuk atribut ini biasanya pada impelentasinya tidak digunakan karena sifatnya yang dinamis. Namun pada model konseptual dibuat untuk memberikan arti lebih pada suatu entitas.



Gambar 4.4 *Derivatif Attribute*

DERAJAT DARI *RELATIONSHIP*

Derajat dari *relationship* menjelaskan jumlah *entity* yang berpartisipasi pada relasi. Ada tiga jenis derajat dari *relationship*, yaitu:

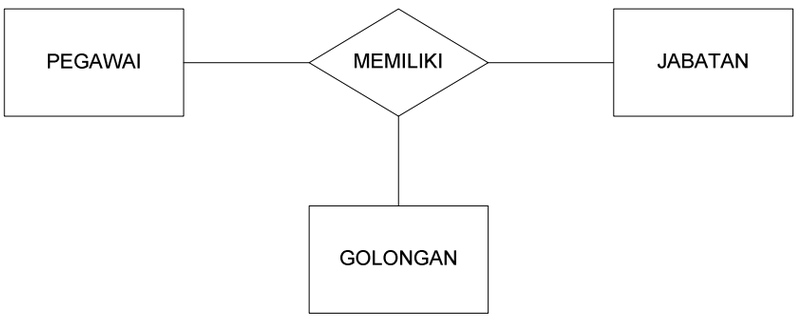
1. **Derajat Satu (*Unary Degree*)**
   Satu entitas terhubung dengan relasi, dan kembali pada entitas lagi.

Gambar 4.5 *Unary Degree*

2. **Derajat Dua (*Binary Degree*)**
   Dua entitas yang terhubung pada relasi.

Gambar 4.6 *Binary Degree*

3. **Derajat Tiga (*Ternary Degree*)**
   Banyak entitas tehubung pada relasi.

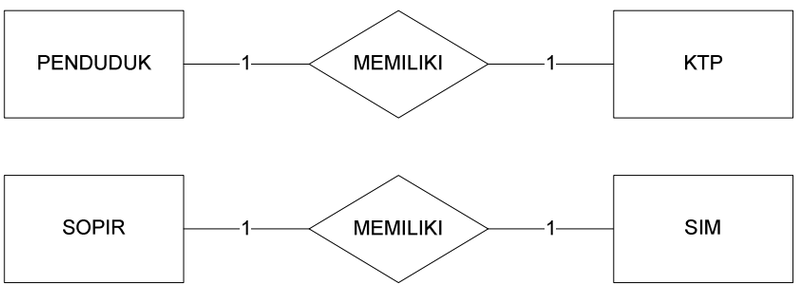


Gambar 4.7 *Ternary Degree*

CARDINALITY CONSTRAINT

Menjelaskan batasan relasi/keterhubungan antara entitas pada himpunan satu dengan himpunan yang lain. Jenis-jenis *cardinality constraint* sebagai berikut :

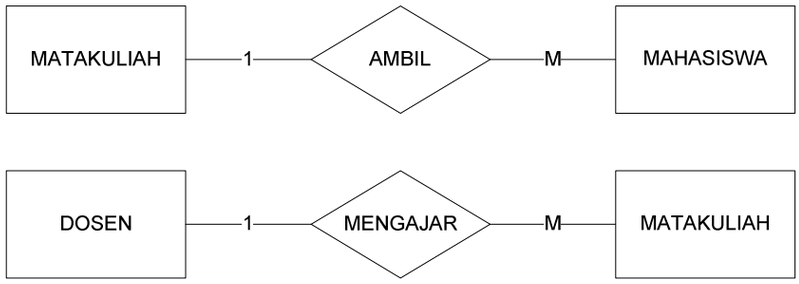
1. **1 to 1 / 1 : 1 / satu ke satu**
   Hubungan kardinalitas ini menjelaskan bahwa setiap entitas pada himpunan X memilik keterhubungan dengan entitas pada himpunan Z maksimal satu entitas. Dan begitu sebaliknya setiap entitas pada himpunan Z memiliki hubungan paling banyak satu entitas pada himpunan X.



Gambar 4.8 Kardinalitas 1 : 1

2. 1 to M / 1 : M / satu ke banyak

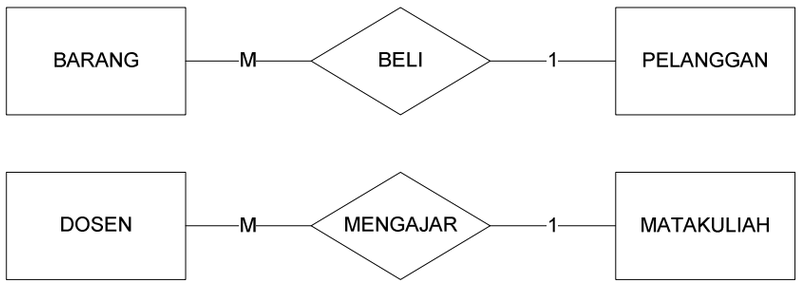
Hubungan kardinalitas ini menjelaskan bahwa setiap entitas pada himpunan X memilik keterhubungan dengan banyak entitas pada himpunan Z. Dan tidak sebaliknya setiap entitas pada himpunan Z memiliki hubungan paling banyak satu entitas pada himpunan X.



Gambar 4.9 Kardinalitas 1 : M

3. M to 1 / M : 1 / banyak ke satu

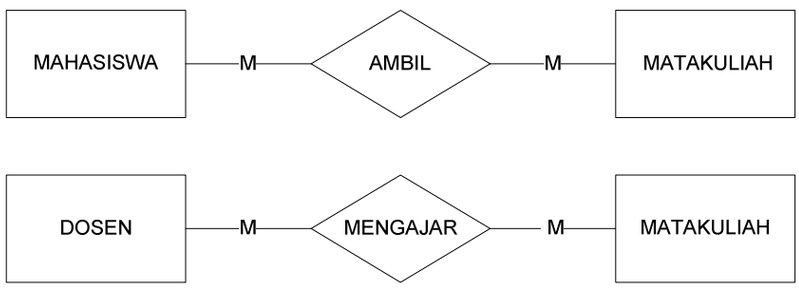
Hubungan kardinalitas ini menjelaskan bahwa setiap entitas pada himpunan X memilik keterhubungan dengan paling banyak satu entitas pada himpunan Z. Dan tidak sebaliknya setiap entitas pada himpunan Z memiliki hubungan banyak entitas pada himpunan X.



Gambar 4.10 Kardinalitas M : 1

4. M to M / M : M / banyak ke banyak

Hubungan kardinalitas ini menjelaskan bahwa setiap entitas pada himpunan X memilik keterhubungan dengan banyak entitas pada himpunan Z. Dan sebaliknya setiap entitas pada himpunan Z memiliki hubungan banyak entitas pada himpunan X.

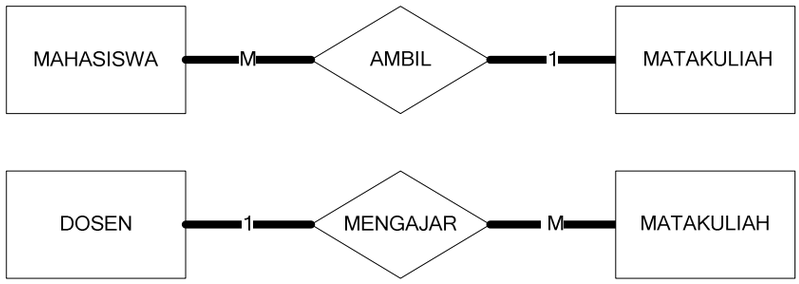


Gambar 4.11 Kardinalitas M : M

PARTICIPANT CONSTRAINT

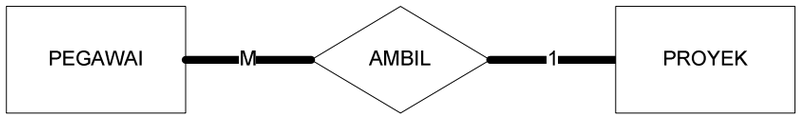
Menjelaskan apakah kebenaran suatu *entity* tergantung hubungannya *entity* lain. Terdapat 2 (dua) macam, yaitu:

1. ***Total Participant***
   Keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain.



Gambar 4.12 *Total Participant*

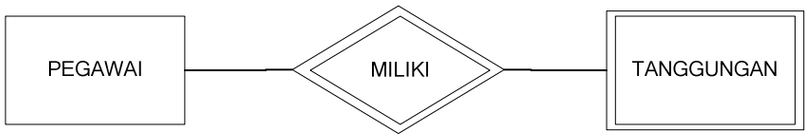
2. ***Partial Participant***
   Keberadaan suatu *entity* tidak tergantung pada hubungannya dengan *entity* lain.



Gambar 4.12 *Total Participant*

WEAK *ENTITY*

Weak *Entity* adalah suatu *Entity* dimana keberadaan dari *entity* tersebut tergantung dari keberadaan *entity* lain. *Entity* yang merupakan induknya disebut *Identifying Owner* dan relationshipnya disebut *Identifying Relationship.* Weak *Entity* selalu mempunyai Total Participation constraint dengan Identifying Owner. Weak *entity* pada relasi disimbolkan dengan persegi panjang ganda.



Gambar 4.13 Weak *Entity*

TAHAPAN PEMBUATAN ER-DIAGRAM

KASUS:

Membuat *database* yang sederhana untuk suatu sistem informasi akademis.

TAHAP 1: PENENTUAN ENTITIES

- **Mahasiswa** : menyimpan semua informasi pribadi mengenai semua mahasiswa
- **Dosen** : menyimpan semua informasi pribadi mengenai semua dosen
- **Mata_kuliah** : menyimpan semua informasi mengenai semua mata kuliah yang ditawarkan
- **Ruang** : menyimpan semua informasi mengenai ruang kelas yang digunakan

TAHAP 2: PENENTUAN *ATTRIBUTES*

- **Mahasiswa**:
  - NPM : nomor induk mahasiswa (integer) PK
  - nama_mhs : nama lengkap mahasiswa (string)
  - alamat_mhs : alamat lengkap mahasiswa (string)

- **Dosen**:
  - Nip : nomor induk pegawai (integer) PK
  - Nama_dosen : nama lengkap dosen (string)
  - Alamat_dosen : alamat lengkap dosen (string)
- **Mata_kuliah**:
  - kode_mk: kode untuk mata kuliah (integer) PK
  - nama_mk: nama lengkap mata kuliah (string)
  - deskripsi_mk: deskripsi singkat mengenai mata kuliah (string)
- **Ruang**:
  - kode_ruang: kode untuk ruang kelas (string) PK
  - lokasi_ruang: deskripsi singkat mengenai lokasi ruang kelas (string)
  - kapasitas_ruang: banyaknya mahasiswa yang dapat ditampung (integer)

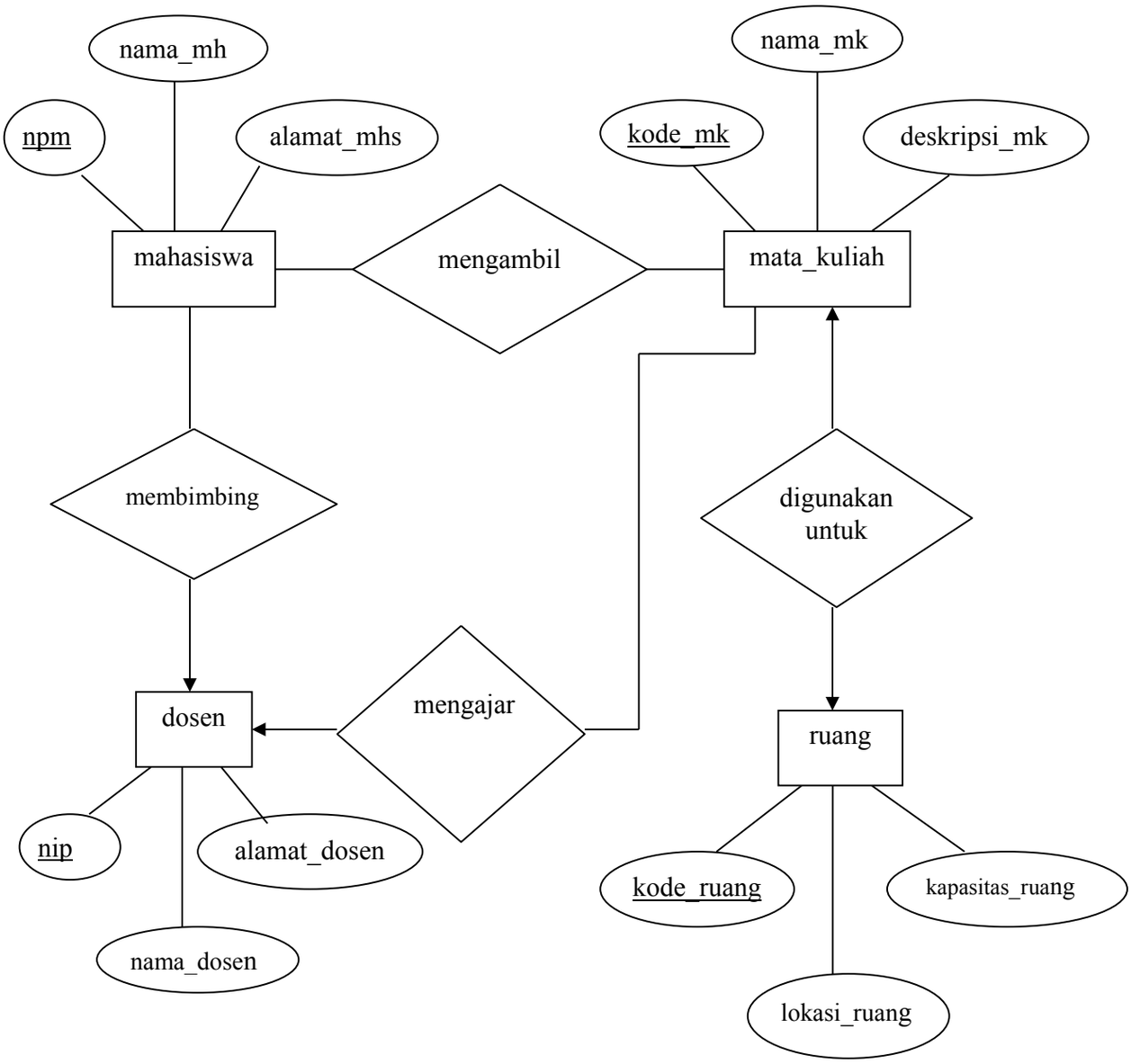
TAHAP 3: PENENTUAN RELATIONSHIPS

	Mahasiswa	**Dosen**	**Mata_kuliah**	**Ruang**
Mahasiswa	-	n:1	m:n	-
Dosen		-	1:n	-
Mata_kuliah			-	1:1
Ruang				-

HUBUNGAN:

- **ruang** digunakan untuk **mata_kuliah**:
  - Tabel utama: **ruang**
  - Tabel kedua: **mata_kuliah**
  - Relationship: One-to-one (1:1)
  - Attribute penghubung: **kode_ruang** (FK **kode_ruang** di **mata_kuliah**)
- **dosen** mengajar **mata_kuliah**:
  - Tabel utama: **dosen**
  - Tabel kedua: **mata_kuliah**
  - *Relationship*: One-to-many (1:n)
  - *Attribute* penghubung: **nip** (FK **nip** di **mata_kuliah**)
- **mahasiswa** mengambil **mata_kuliah**:
  - Tabel utama: **mahasiswa, mata_kuliah**
  - Tabel kedua: **mhs_ambil_mk**
  - Relationship: Many-to-many (m:n)
  - Attribute penghubung: **nim**, **kode_mk** (FK **nim**, **kode_mk** di **mhs_ambil_mk**)
- **dosen** membimbing **mahasiswa**:
  - Tabel utama: **dosen**
  - Tabel kedua: **mahasiswa**
  - Relationship: One-to-many (1:n)
  - Attribute penghubung: **nip** (FK **nip** di **mahasiswa**)

Tahap 4: Pembuatan ERD



Gambar 4.14 ER-Diagram Final

LATIHAN SOAL

a. Buatlah ER-Diagram dengan tema "**Bimbingan Skripsi Online**" (Individu (NPM Akhiran Ganjil)) ***Learn By Your Self, Please !***
b. Buatlah ER-Diagram dengan tema "**Nilai Mahasiswa**" (Individu (NPM Akhiran Genap)) ***Learn By Your Self, Please !***
c. Buatlah ER-Diagram dengan tema "**Pengisian Kartu Rencana Studi Online**" (Kelompok (Max 5 Orang)) ***Learn By Your Self, Please !***

***) DIKUMPUL SATU MINGGU SETELAH PERTEMUAN INI !!!**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

NORMALISASI CHAPTER 5

Proses normalisasi pertama kali diperkenalkan oleh E.F.Codd pada tahun 1972. Normalisasi sering dilakukan sebagai suatu uji coba pada suatu relasi secara berkelanjutan untuk menentukan apakah relasi tersebut sudah baik atau masih melanggar aturan-aturan standar yang diperlakukan pada suatu relasi yang normal (sudah dapat dilakukan proses *insert, update, delete*, dan *modify* pada satu atau beberapa atribut tanpa mempengaruhi integritas data dalam relasi tersebut). Proses normalisasi merupakan metode yang formal/standar dalam mengidentifikasi dasar relasi bagi primary keynya (atau *candidate key* dalam kasus BCNF), dan dependensi fungsional diantara atribut-atribut dari relasi tersebut. Normalisasi akan membantu perancang basis data dengan menyediakan suatu uji coba yang berurut yang dapat diimplementasikan pada hubungnan individualshingga skema relasi dapat di normalisasi ke dalam bentuk yang lebih spesifik untuk menghindari terjadinya *error* atau inkonsistansi data, bila dilakuan update tehadap relasi tersebut dengan *anomaly*.

DEFINISI NORMALISASI

- Normalisasi adalah suatu proses memperbaiki / membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika.
- Normalisasi adalah proses pengelompokan data ke dalam bentuk tabel atau relasi atau file untuk menyatakan entitas dan hubungan mereka sehingga terwujud satu bentuk database yang mudah untuk dimodifikasi.
- Normalisasi dapat berguna dalam menjawab 2 pertanyaan mendasar yaitu: “apa yang dimaksud dengan desain database logical?” dan “apa yang dimaksud dengan desain *database* fisikal yang baik? *What is phisical good logical database design*?”.
- Normalisasi adalah suatu proses untuk mengidentifikasi “tabel” kelompok atribut yang memiliki ketergantungan yang sangat tinggi antara satu atribut dengan atrubut lainnya.

TUJUAN NORMALISASI

Normalisasi perlu dilakukan agar kerelasian dalam basis data menjadi mudah dimengerti, mudah dipelihara, mudah diprosesnya, mudah untuk dikembangkan sesuai kebutuhan baru.

MACAM-MACAM PENYIMPANGAN

INSERTION ANOMALY

Insertion Anomaly, merupakan error atau kesalahan yang terjadi sebagai akibat dari operasi menyisipkan (insert) tuple / record pada sebuah relasi.
Contoh: ada matakuliah baru (CS-600) yang akan diajarkan, maka matakuliah tersebut tidak bisa di *insert* / disisipkan ke dalam Relasi Kuliah di atas sampai ada mahasiswa yang mengambil matakuliah tersebut.

DELETE ANOMALY

Delete Anomaly merupakan *error* atau kesalahan yang terjadi sebagai akibat operasi penghapusan (*delete*) terhadap tupe / record dari sebuah relasi.
Contoh: mahasiswa dengan NIM □92425 (pada Relasi Kuliah di atas), memutuskan untuk batal ikut kuliah CS-400, karena ia merupakan satu-satunya peserta matakuliah tersebut, maka bila record / tuple tersebut dihapus / delete akan berakibat hilangnya informasi bahwa mata kuliah CS400, biayanya 150.

UPDATE ANOMALY

Update Anomaly merupakan error atau kesalahan yang terjadi sebagai akibat operasi perubahan (update) tuple / record dari sebuah relasi.
Contoh: biaya kuliah untuk matakuliah CS-200 (pada relasi kuliah di atas) akan dinaikkan dari 75 menjadi 100, maka harus dilakukan beberapa kali modifikasi terhadap record-record atau tuple-tuple mahasiswa yang mengambil matakuliah CS-200 tersebut, agar data teap konsisten.

FUNCTIONAL DEPENDENCIES

Salah satu konsep utama yang berhubungan dengan normalisasi adalah *functional dependency* (ketergantungan fungsional). Suatu ketergantungan fungsional menggambarkan *relationship*/hubungan di antara atribut-atribut.

FUNCTIONAL DEPENDENCY

Functional dependency (ketergantungan fungsional) menggambarkan relationship/hubungan antara atribut-atribut dengan relasi. Sebagai contoh: Jika A dan B adalah atribut-atribut dari relasi R. B dikatakan *functionally* dependent (bergantungan fungsional) terhadap A (dinotasikan dengan A → B), jika masing-masing nilai dari A dalam relasi R berpasang-an secara tepat dengan satu nilai dari B dalam relasi R. Ketergantungan antara atribut-atribut A dengan B dapat dilihat pada gambar di bawah ini.

Gambar 5.1 *Functional Dependency*

FULL FUNCTIONAL DEPENDENCY

Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sepenuhnya, jika perpindahan beberapa atribut dari A menghasilkan tepat satu pasangan pada atribut B. Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sebagian, jika ada beberapa atribut yang dapat dihilangkan dari A sementara ketergantungan tersebut tetap berlaku /berfungsi. Sebagai contoh dapat dilihat pada suatu ketergantungan fungsional di bawah ini.

NIK, Nama → Kd_Cabang.

Dari kasus di atas dapat dikatakan bahwa masing-masing nilai dari NK, Nama berasosiasi/berelasi dengan nilai tunggal dari Kd_Cabang. Dengan demikian, relasi di atas tidak memiliki ketergantungan fungsional sepenuhnya (*full funcional dependency*), karena Kd_Cabang juga masih memiliki ketergantungan fungsional pada himpunan bagian dari (NIK, Nama). Dengan kata lain, Kd_Cabang adalah *Full functional dependecy* hanya pada NIK.

TRANSITIVE DEPENDENCY

Suatu kondisi dimana A, B, dan C adalah atribut-atribut dari suatu relasi sedemikian sehingga A → B dan B → C, maka A → C (C memiliki ketergantungan transitif terhadap A melalui B), dan harus dipastikan bahwa A tidak memiliki ketergantungan fungsional (functional dependent) terhadap B atau C). Sebagai contoh dapat dilihat ketergantungan fungsional yang ditunjukkan pada relasi sebagai berikut.

NIK → Kd_Cabang dan Kd_ Cabang → Alamat_Cabang

Dapat dilihat ketergantungan transitif NIK → Alamat_Cabang dapat terjadi melalui atribut Kd_Cabang.

DETERMINANT

Determinan dari suatu functional dependency (ketergantungan fungsional) menunjuk/ mengarahkan ke atribut atau kelompok atribut-atribut yang berbeda pada sebelah kiri anak panah gambar 5.1 di atas. Ketika terdapat suatu functional dependency (ketergantungan fungsional) atribut atau group atribut-atribut di sebelah kiri anak panah dinamakan determinan. Pada gambar 5.1 ditunjukkan bahwa A adalah determinan dari B.

ATRIBUT TABEL

Atribut adalah karakteristik atau sifat yang melekat pada sebuah tabel. Pengelompokkan atribut :

1. **Atribut Key**
   Adalah satu atau gabungan dari beberapa atribut yang mewakili atribut yang lain dan memiliki sifat unik. Ada 3 (tiga) key :
   - **Superkey**
     Merupakan satu atau kumpulan atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.
     - **Contoh** : superkey dalam tabel mahasiswa.
       - **(npm, nama, alamat, tgllahir)**
       - **(npm, nama, alamat)**
       - **(npm, nama)**
       - **(npm)**

   - **Candidate Key**
     Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Sebuah CK pasti superkey, tapi belum tentu sebaliknya.
     - **Contoh** : CK dalam tabel mahasiswa.
       - **(npm)**
       - **(nama)**

   - **Primary Key**
     Dari beberapa CK dapat dipilih satu PK sebagai key yang memiliki keunikan paling baik.
     - **Contoh** : Dari tabel mahasiswa, yang layak dijadikan sebagai PK adalah NPM.

   - **Composite Key**
     Gabungan beberapa atribut yang memiliki keunikan yang sama dan dijadikan sebagai key.
     - **Contoh** : **DetailJual (<u>NoInvoice</u>, <u>KdBarang</u>, HargaJual, Qty)**

   - **Alternate Key**

Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik atau sebagai alternatif key. Sebuah AK pasti superkey, tapi belum tentu sebaliknya.

- **Contoh** : Dari tabel mahasiswa, yang layak dijadikan sebagai PK adalah NPM.

2. **Atribut Deskriptif**
   Merupakan atribut yang bukan merupakan anggota dari PK.

3. **Atribut Sederhana**
   Adalah atribut atomik yang tidak dapat dipilah lagi.

4. **Atribut Komposit**
   Adalah atribut yang masih bisa diuraikan lagi menjadi sub-atribut yang masing-masing memiliki makna. Contoh : Alamat → Alamat, Kota, Kelurahan, Kecamatan, Propinsi.
   Gelar → Gelar Depan, Gelar Belakang

5. **Atribut Bernilai Tunggal**
   Ditujukan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data.
   Contoh : NPM, Nama, Tanggal lahir → hanya dapat berisi satu nilai untuk seorang mahasiswa

6. **Atribut Bernilai Banyak**
   Ditujukan pada atribut-atribut yang dapat diisi dengan lebih dari satu nilai, tapi jenisnya sama.
   Contoh : Atribut Hobby → seseorang dapat memiliki banyak hobby

7. **Atribut harus bernilai**
   Adalah atribut yang nilainya tidak boleh kosong, atau harus ada nilainya. Misalnya data NPM dan Nama mahasiswa

LANGKAH-LANGKAH PEMBENTUKAN NORMALISASI

TAHAPAN NORMALISASI

Bentuk Tidak Normal

↓ Menghilangkan perulangan group

Bentuk Normal Pertama (1NF)

↓ Menghilangkan ketergantungan sebagian

Bentuk Normal Kedua (2NF)

↓ Menghilangkan ketergantungan transitif

Bentuk Normal Ketiga (3NF)

↓ Menghilangkan anomali-anomali hasil dari ketergantungan fungsional

Bentuk Normal Boyce-Codd (BCNF)

↓ Menghilangkan Ketergantungan Multivalue

Bentuk Normal Keempat (4NF)

↓ Menghilangkan anomali-anomali yang tersisa

Bentuk Normal Kelima

UNNORMALIZED FORM (BENTUK TIDAK NORMAL)

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

Hal 1 — ***Rumah_Impian*** Perincian Pelanggan — Tanggal : 7-Oct-95

Nama Pelanggan : John Key — Nomor Pelanggan CR 76

No_Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pemilik
PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
PG16	Jl. Huzai /12, Jakarta	1-Sep-95	1-Sep-96	450	CO93	Durki

Gambar 5.2 Faktur Rumah Pelanggan

Berdasarkan sampel dokumen diatas, kita akan identifikasi data-data apa saya yang akan direkam dan dijadikan sebagai bentuk tidak normal.

No_Pelanggan	Nama	Nomor Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pemilik
CR76	Badi	PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
		PG16	Jl. Huzai /12, Jakarta	1-Sep-95	1-Sep-96	450	CO93	Durki
CR56	Sirajuddin	PG4	Jl. Aai / 07, Jakarta	1-Sep-92	10-Jun-93	350	CO40	Ewin
		PG36	Jl. Azhar / 49, Jakarta	10-Oct-93	1-Dec-94	375	CO93	Durki
		PG16	Jl. Huzai /12, Jakarta	1-jan-95	10-Aug-95	450	CO93	Durki

Gambar 5.3 Bentuk Tidak Normal

Dari tabel di Pelanggan Biaya (Tabel 5.2) di atas terdapat multiple value pada beberapa atributnya. Misalkan terdapat dua (2) nilai untuk No_Property yaitu PG4 dan PG16 untuk Nama Pelanggan (Nama) Badi.

Untuk mentransformasikan tabel yang belum ternomalisasi di atas menjadi tabel yang memenuhi kriteria 1NF adalah kita harus merubah seluruh atribut yang multivalue menjadi atribut single value, dengan cara menghilangkan repeating group pada tabel di atas.

Repeating Group (elemen data berulang) adalah (No_Property, Alamat_Property, Tgl_Pinjam, Tgl_Selesai, Biaya, No_Pemilik, Nama_Pemilik)

FIRST NORMAL FORM (BENTUK NORMAL KE SATU)

Pada tahap ini dilakukan penghilangan beberapa group elemen yang berulang agar menjadi satu harga tunggal yang berinteraksi di antara setiap baris pada suatu tabel, dan setiap atribut harus mempunyai nilai data yang atomic (bersifat atomic value). Atom adalah zat terkecil yang masih memiliki sifat induknya, bila terpecah lagi maka ia tidak memiliki sifat induknya.

Syarat normal ke satu (1-NF) antara lain:

1. setiap data dibentuk dalam flat file, data dibentuk dalam satu record demi satu record nilai dari field berupa “atomic value”.

2. tidak ada set atribute yang berulang atau bernilai ganda.
3. telah ditentukannya primary key untuk tabel / relasi tersebut.
4. Tiap atribut hanya memiliki satu pengertian.

Langkah pertama yang dilakukan pada Tabel Pelanggan Biaya (pada Tabel 5.2) tersebut adalah menghilangkan elemen data yang berulang dengan data-data Pelanggan yang sesuai pada setiap baris. Hasil dari tabel yang telah memenuhi bentuk normal pertama dapat dilihat pada Tabel 5.4. kita dapat mengidentifikasi primary key untuk relasi Pelanggan_Biaya yang masih memiliki composite key (No_Pelanggan, No_Property). Pada kasus ini kita akan memperoleh primary key yang bersifat composite key.

Relasi Pelanggan_Biaya dapat didefinisikan sebagai berikut. Pelanggan_Biaya = (No_Pelanggan, No_Property, Nama, Alamat_Property, Tgl_Pinjam, Tgl_Selesai, Biaya, No_Pemilik, Nama_Pemilik)

No_Pe-langgan	Nama	Nomor Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pe-milik
CR76	Badi	PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
CR76	Badi	PG16	Jl. Huzai /12, Jakarta	1-Sep-95	1-Sep-96	450	CO93	Durki
CR56	Siraju ddin	PG416	Jl. Aai / 07, Jakarta	1-jan-95	10-Aug-95	450	CO93	Durki
CR56	Siraju ddin	PG36	Jl. Azhar / 49, Jakarta	10-Oct-93	1-Dec-94	375	CO93	Durki
CR56	Siraju ddin	PG4	Jl. Huzai /12, Jakarta	1-Sep-95	10-Jun-93	350	CO40	Ewin

Gambar 5.4 Bentuk Normal Ke Satu

SECOND NORMAL FORM (BENTUK NORMAL KEDUA)

Bentuk normal kedua didasari atas konsep full functional dependency (ketergantungan fungsional sepenuhnya) yang dapat didefinisikan sebagai berikut.

Jika A adalah atribut-atribut dari suatu relasi, B dikatakan full functional dependency (memiliki ketergantungan fungsional terhadap A, tetapi tidak secara tepat memiliki ketergantungan fungsional dari subset (himpunan bagian) dari A.

FULL FUNCTIONAL DEPENDENCY (Ketergantungan Fungsional Sepenuhnya)

Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sepenuhnya, jika perpindahan beberapa atribut dari A menghasilkan tepat satu pasangan pada atribut B. Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sebagian, jika ada beberapa atribut yang dapat dihilangkan dari A sementara ketergantungan tersebut tetap berlaku /berfungsi.

Sebagai contoh dapat dilihat pada suatu ketergantungan fungsional di bawah ini.

NIK, Nama → Kd_Cabang.

Dari kasus di atas dapat dikatakan bahwa masing-masing nilai dari NK, Nama berasosiasi/berelasi dengan nilai tunggal dari Kd_Cabang. Dengan demikian, relasi di atas tidak memiliki ketergantungan fungsional sepenuhnya (full funcional dependency), karena Kd_Cabang juga masih memiliki ketergantungan fungsional pada himpunan bagian dari (NIK, Nama). Dengan kata lain, Kd_Cabang adalah Full functional dependecy hanya pada NIK. Bentuk normal kedua memungkinkan suatu relasi memiliki composite key, yaitu relasi dengan primary key yang terdiri dari dua atau lebih atribyut. Suatu relasi yang memiliki sigle atribut untuk primary keynya secara otomatis pada akhirnya menjadi 2-NF.

Syarat normal kedua (2-NF) sebagai berikut :

1. Bentuk data telah memenuhi kriteria bentuk normal kesatu.

2. Atribute bukan kunci (non-key) haruslah memiliki ketergantungan fungsional sepenuhnya (fully functional dependency) pada kunci utama / primary key.

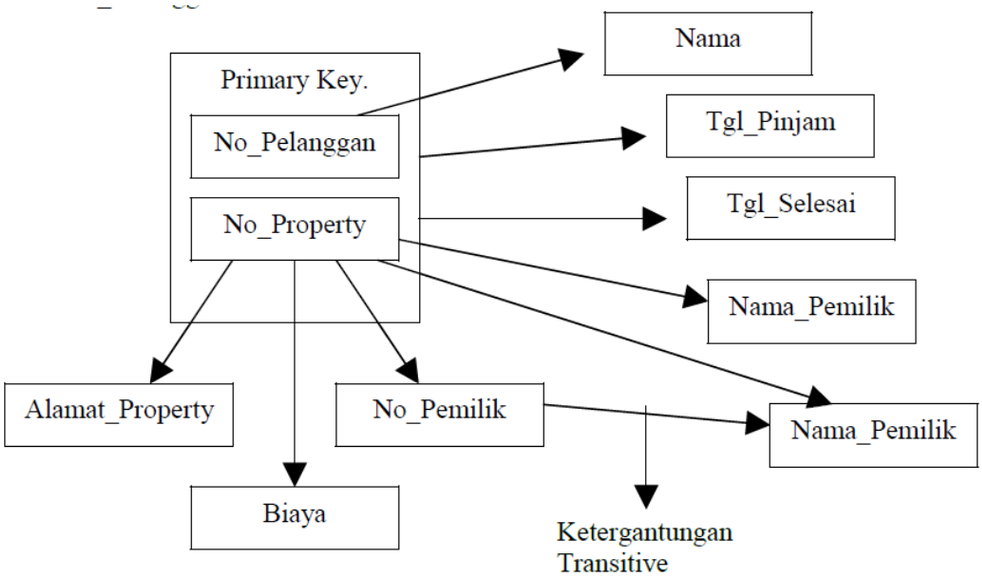
Dengan demikian untuk membentuk normal kedua haruslah sudah ditentukan primary keynya. Primary key tersebut haruslah lebih sederhana, lebih unik, dapat mewakili atribute lain yang menjadi anggotanya, dan lebih sering digunakan pada tabel / relasi tersebut.

Langkah pertama kita harus mengidentifikasi ketergantungan fungsional dalam relasi Pelanggan_Biaya, sebagai berikut.

No_Pelanggan, No_Property → Tgl_Pinjam, Tgl_Selesai
No_Pelanggan → Nama
No_Property → Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik
No_Pemilik → Nama_Pemilik

Gambar 5.2 di bawah ini akan mengilustrasikan ketergantungan fungsional yang berada dalam relasi Pelanggan_Biaya. Berdasarkan Functional Dependency (Ketergantungan Fungsional) relasi
Pelanggan_Biaya pada gambar 5.2 di atas, maka dapat dilihat beberapa kondisi sebagai berikut.

- Primary key pada Pelanggan_Biaya di atas adalah No_Pelanggan, dan No_Property.
- Atribut Pelanggan (Nama) hanya memiliki (partially dependency) ketergantungan pada sebagian / pada salah satu primary key di relasi Pelanggan_Biaya yaitu hanya atribut No_Pelanggan.



Gambar 5.5 Functional Dependency (Ketergantungan Fungsional) pada relasi

- Sementara atribut-atribut property (Alamat_property, Biaya, No_Pemilik, Nama_Pemilik), juga hanya memiliki (partially dependency) ketergantungan pada sebagian / pada salah satu primary key di relasi Pelanggan_Biaya yaitu hanya atribut No_Property.
- Atribut-atribut Biaya Property (Tgl_Pinjam, Tgl_Selesai), memiliki (fully dependent) tergantung sepenuhnya pada semua primary key di relasi Pelanggan_Biaya yaitu atribut No_Pelanggan, dan No_Property.
- Pada gambar 9.2 di atas juga menunjukkan sebuah (transitive dependency) ketergantungan transitif terhadap primary key, namun hal itu tidak akan melanggar ketentuan pada normalisasi ke dua (2-NF). Ketergantungan transitive akan dihilangkan pada normalisasi ketiga (3-NF).

Seluruh identifikasi yang dilakukan pada point 1 sampai 4 di atas menunjukkan bahwa relasi Pelanggan_Biaya di atas belum memenuhi kriteria bentuk normal ke dua (2-NF). Kita perlu membuat beberapa relasi Pelanggan_Biaya di atas ke dalam bentuk normal ke dua (2-NF), agar seluruh atribut non-key (yang bukan primary key) dapat memiliki ketergantungan sepenuhnya (full functionally dependent) terhadap primary key. Ketiga buah relasi tersebut dapat ditulis dalam bentuk ini.

1. Relasi / Tabel Pelanggan dengan atribut-atribut:
   No_Pelanggan, Nama. {No_Pelanggan berfungsi sebagai primary key pada tabel / relasi tersebut}.
2. Relasi / Tabel Biaya dengan atribut-atribut: No_Pelanggan, No_Property, Tgl_Pinjam, Tgl_Selesai. {No_Pelanggan dan No_Property berfungsi sebagai primary key pada tabel / relasi tersebut}.
3. Relasi / Tabel Property_Pemilik dengan atribut-atribut: No_Property, Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik. { No_Property berfungsi sebagai priamry key pada tabel / relasi tersebut}.

Tabel / Relasi yang telah memenuhi kriteria normal ke dua (2-NF) adalah sebagai berikut.

No_Pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Pelanggan

No_Pelanggan	No_Property	Tgl_Pinjam	Tgl_Selesai
CR76	PG4	1-Jul-93	31-Aug-95
CR76	PG16	1-Sep-95	1-Sep-96
CR56	PG4	1-Sep-95	10-Jun-93
CR56	PG36	10-Oct-93	1-Dec-94
CR56	PG16	1-Jan-95	10-Aug-95

(b) Relasi Biaya

No_Property	Alamat_Property	Biaya	No_Pemilik	Nama_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40	Ewin
PG16	Jl.Huzai /12,Jakarta	450	CO93	Durki
PG36	Jl.Suciana / 68, Bogor	375	CO93	Durki

Gambar 5.6 Bentuk Normal Kedua

BENTUK NORMAL KE TIGA (THIRD NORMAL FORM / 3 NF)

Walaupun relasi 2-NF memiliki redudansi yang lebih sedikit dari pada relasi 1-NF, namun relasi tersebut masih mungkin mengalami kendala bila terjadi anomaly peremajaan (update) terhadap relasi tersebut. Misalkan kita akan melakukan update terhadap nama dari seorang Pemilik (pemilik), seperti Durki (No_Pemilik: CO93), kita harus melakukan update terhadap dua baris dalam relasi Property_Pemilik, Jika kita hanya mengupdate satu baris saja, sementara baris yang lainnya tidak, maka data di dalam database tersebut akan inkonsisten / tidak teratur. Anomaly update ini disebabkan oleh suatu ketergantungan transitif (transitive dependency). Kita harus menghilangkan ketergantungan tersebut dengan melakukan normalisasi ketiga (3-NF).

TRANSITIVE DEPENDENCY (KETERGANTUNGAN TRANSITIF)

Suatu kondisi dimana A, B, dan C adalah atribut-atribut dari suatu relasi sedemikian sehingga A → B dan B → C, maka A → C (C memiliki ketergantungan transitif terhadap A melalui B), dan harus dipastikan bahwa A tidak memiliki ketergantungan fungsional (functional dependent) terhadap B atau C).

NIK → Kd_Cabang dan Kd_ Cabang → Alamat_Cabang

Dapat dilihat ketergantungan transitif NIK → Alamat_Cabang dapat terjadi melalui atribut Kd_Cabang.

Syarat normal ketiga (Third Normal Form / 3 NF) sebagai berikut:

- Bentuk data telah memenuhi kriteria bentuk normal kedua.
- Atribute bukan kunci (non-key) harus tidak memiliki ketergantungan transitif, dengan kata lain suatu atribut bukan kunci (non_key) tidak boleh memiliki ketergantungan fungsional (functional dependency) terhadap atribut bukan kunci lainnya, seluruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap priamry key di relasi itu saja.

Sebagai contoh kita dapat melihat beberapa ketergantungan fungsional (functional dependencies) pada relasi Pelanggan, Biaya, dan Property_Pemilik berikut ini.

- Relasi / Tabel Pelanggan terdiri dari atribut-atribut:
  No_Pelanggan → Nama
  {No_Pelanggan sebagai primary key}

- Relasi / Tabel Biaya terdiri dari atribut-atribut:
  No_Pelanggan, No_property □ Tgl_Pinjam, Tgl_Selesai
  {No_Pelanggan, dan No_property sebagai primary key}

- Relasi / Tabel Property_Pemilik terdiri dari atribut-atribut
  No_property → Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik
  No_Pemilik → Nama_Pemilik
  {No_property sebagai primary key}

Seluruh atribut non-primary key pada relasi Pelanggan dan Biaya di atas terlihat memiliki ketergantungan fungsional (functional dependency) terhadap primary key dari masing-masing tabel / relasi. Relasi / tabel Pelanggan dan Biaya di atas tidak memiliki ketergantungan transitif (transitive dependency), sehingga tabel tersebut telah memenuhi kriteria normal ketiga (3-NF).

Seluruh atribut non-primary key pada relasi Property_Pemilik di atas terlihat memiliki ketergantungan fungsional (functional dependency) terhadap primary key, kecuali Nama_Pemilik yang masih memiliki ketergantungan fungsional (functional dependency) terhadap No_Pemilik. Inilah contoh ketergantungan dari transitif (transitive dependency), yang terjadi ketika atribut non-primary key (Nama_Pemilik) bergantung secara fungsi terhadap satu atau lebih atribut non-primary key lainnya (No_Pemilik). Kita harus menghilangkan ketergantungan transitif (transitive dependency) tersebut dengan menjadikan relasi Property_Pemilik menjadi 2 relasi / tabel dengan format / bentuk sebagai berikut.

- Relasi / Tabel Property_Untuk_Pemilik yang terdiri dari atribut-atribut:
  No_property □ Alamat_Property, Biaya, No_Pemilik
  {No_property sebagai primary key}
- Dan relasi Pemilik yang terdiri dari atribut-atribut:
  No_Pemilik □ Nama_Pemilik
  {No_Pemilik sebagai primary key}

Bila digambarkan ke dalam tabel menjadi seperti berikut ini.

No_Property	Alamat_Property	Biaya	No_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40
PG16	Jl.Huzai /12,Jakarta	450	CO93
PG36	Jl.Suciana / 68, Bogor	375	CO93

(c 1) Relasi Property_Untuk _Biaya

No_Pemilik	Nama_Pemilik
CO40	Ewin
CO93	Durki
CO93	Durki

(c 2) Relasi Pemilik

Hasil akhir normalisasi tabel Pelanggan_Biaya sampai ke bentuk normal ketiga adalah sebagai berikut.

No_Pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Customer

No_Pelanggan	No_Property	Tgl_Pinjam	Tgl_Selesai
CR76	PG4	1-Jul-93	31-Aug-95
CR76	PG16	1-Sep-95	1-Sep-96
CR56	PG4	1-Sep-95	10-Jun-93
CR56	PG36	10-Oct-93	1-Dec-94
CR56	PG16	1-Jan-95	10-Aug-95

(b) Relasi Rental

No_Property	Alamat_Property	Biaya	No_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40
PG16	Jl.Huzai /12,Jakarta	450	CO93
PG36	Jl.Suciana / 68, Bogor	375	CO93

(c) Relasi Property_for _Rent

No_Pemilik	Nama_Pemilik
CO40	Ewin
CO93	Durki
CO93	Durki

(d) Relasi Owner

Gambar 5.7 Hasil Normal Bentuk Ketiga

LATIHAN SOAL

a. Lakukan normalisasi berdasarkan kasus-kasus yang anda temukan di sekitar anda, dilengkapi dengan bukti dokumen sebagai bahan normalisasi.

***) DIKUMPUL SATU MINGGU SETELAH PERTEMUAN INI !!!**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA DEFINITION LANGUAGE

CHAPTER 6

Antara SQL dan SQL SERVER 2008

Structure Query Language (SQL) adalah suatu bahasa standar yang dapat dimengerti komputer yang dikhususkan untuk membahasakan/penyeleksian terhadap sebagai data pada satu basisdata atau pada banyak basis data dengan tujuan untuk menghasilkan informasi yang diinginkan. Tak dipungkiri bahwasanya SQL adalah bahasa yang wajib dimengerti oleh para programmer selain dari bahasa pemrograman*interface* itu sendiri. Dengan SQL memungkinkan kita untuk mengakses dan memanipulasi atau mengolah basisdata sesuai dengan bahasa standar dari query.SQL terdiri dari **1)DDL** (*data definition Language*), untuk membuat (CREATE), merubah (ALTER), dan menghapus (DROP) basisdata.**2)DML** (*Data Manipulation Language*), untuk memanipulasi isi tabel seperti **INSERT**, **SELECT**, **UPDATE** dan **DELETE, 3)DTL**(*Data Transaction Language*), untuk mengolah transaksi basis data seperti **COMMIT, ROLLBACK, 4) DCL** (*Data Control Language*), untuk pengendalian akses pada basisdata seperti **GRANT, REVOKE.**

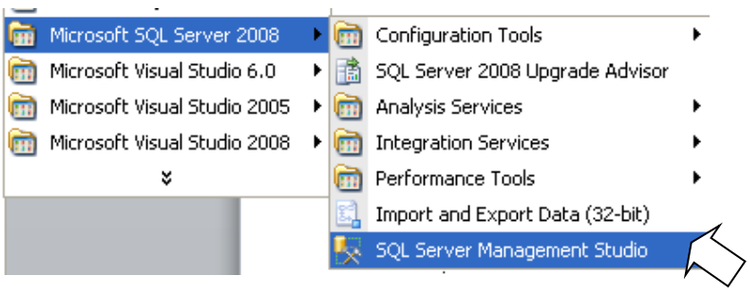
Sql server 2008 adalah versi pengembangan basisdata setelah Sql Server 2005 dan 2000. Penggunaan Sql Server 2008 sedikit berbeda dengan versi sebelumnya, sebagai contoh bila dibandingkan dengan Sql Server 2000, terdapat pilihan untuk membuka record dengan hanya memilih pilihan khusus (klik kanan) mouse (*return all rows*) pada tabel yang diinginkan. Sedangkan pada Sql Server 2008, untuk melihat setiap record yang telah disimpan, anda selalu menggunakan perintah dalam bentuk Sql.

Pada chapter ini kita akan pelajari terlebih cara menggunakan SQL Server 2008 dimulai dari membuka SQL Server 2008, membuat database dan tabel.

1. MEMULAI SQL SERVER 2008

Untuk memulai menggunakan SQL Server 2008, salah satu langkah yang harus dilakukan adalah sebagai berikut:

1. Pastikan anda telah membuat folder terlebih dahulu untuk media simpan projek SQL SERVER 2008
2. Klik menu ***START>>ALLPROGRAMS***
3. Pilih *Microsoft*SQL Server 2008 >> SQL Server Management Studio



Gambar 1.1 Membuka SQL Server 2008

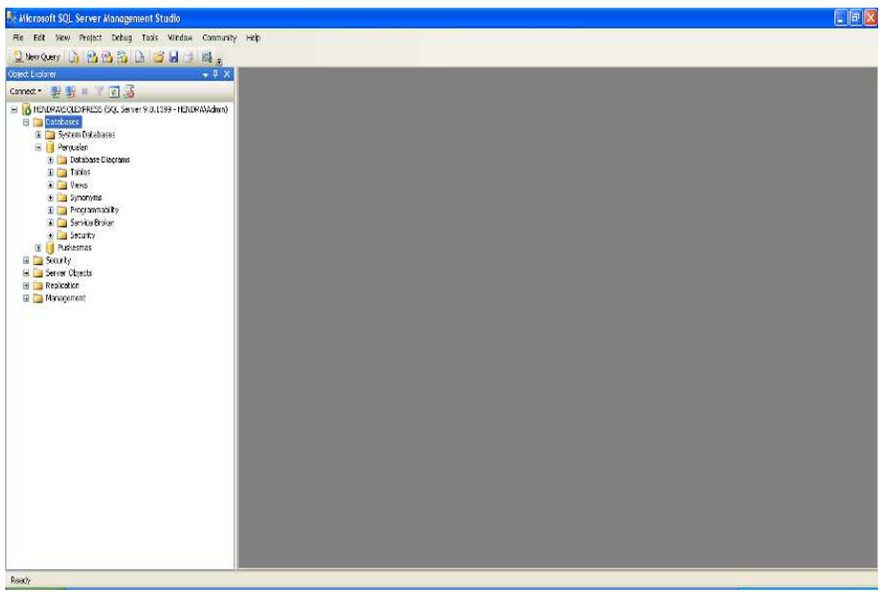
4. Lalu akan muncul gambar seperti dibawah ini:

Server Name : Isi dengan nama server yang terinstall

Authentication :
1.Windows Authentication, dipilih Berdasarkan sistem operasinya.
2. SQL Server Authentication, denganmengisi**username : sa** dan**Password** sesuai dengan yang telah ditetapkan pada saat instalasi

Gambar 1.2 Tampila Koneksi ke SQL Server 2008

Setelah pemilihan pada kotak doalog Connect to Server berhasil, maka akan tampil sbb:

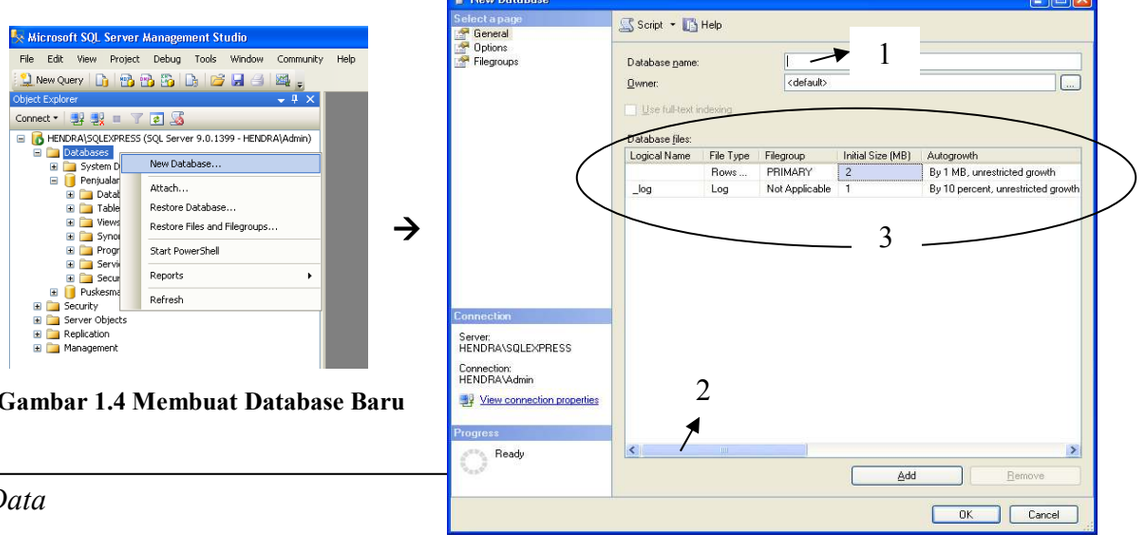


Gambar 1.3 Tampilan SQL Server Management Studio

2. MEMBUAT DATABASE TANPA QUERY

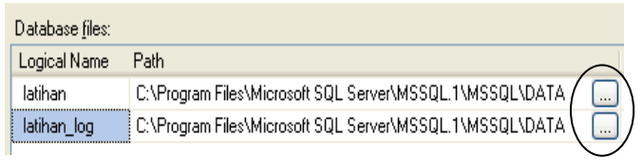
Untuk membuat *database*dengan cara ini diperlukan persiapan terlebih dahulu. Persiapan tersebut adalah sebagai berikut:

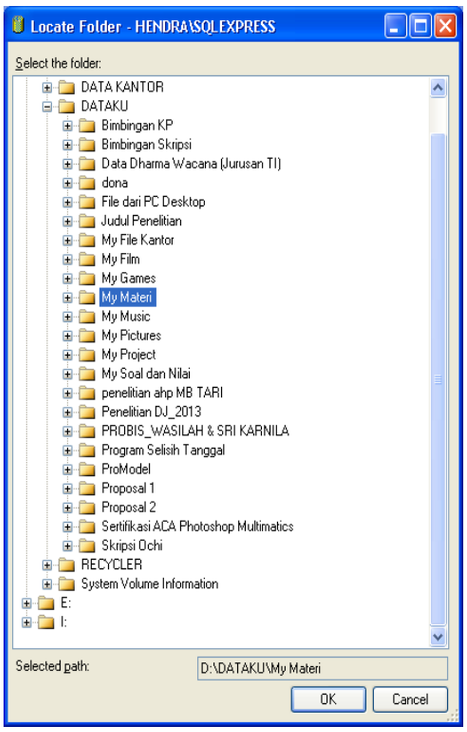
1. Buat folder di driveD:>DATAMHS dan beri nama latihan atau nama yang lainnya.
2. Posisi saat ini adalah seperti pada gambar 1.3, lalu klik kanan pada **databases→ New Database**



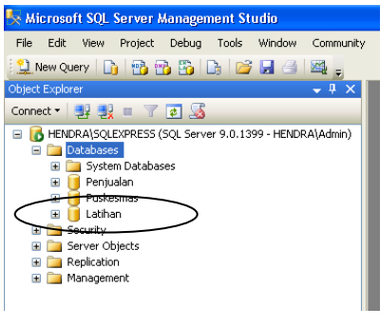
Gambar 1.4 Membuat Database Baru

3. Nomor **1,** isikan dengan nama database yang akan dibuat.
   Nomor **2,** lakukan scroll kekanan untuk mendapat kolom dengan nama**path.**
   Nomor **3,** pada kotak **database files** terdapat dua baris file yang harus diarahkan pada **path** atau folder yang telah kita persiapkan tadi.





Note : Kedua baris ini merupakan file dengan Ekstensi .MDF dan Log. Kedua file tersebut Wajib disimpan pada area penyimpanan Yang sama. Jika salah satu dari file tersebut Tidak ada/disimpan, maka database tidak dapat diaktifkan.

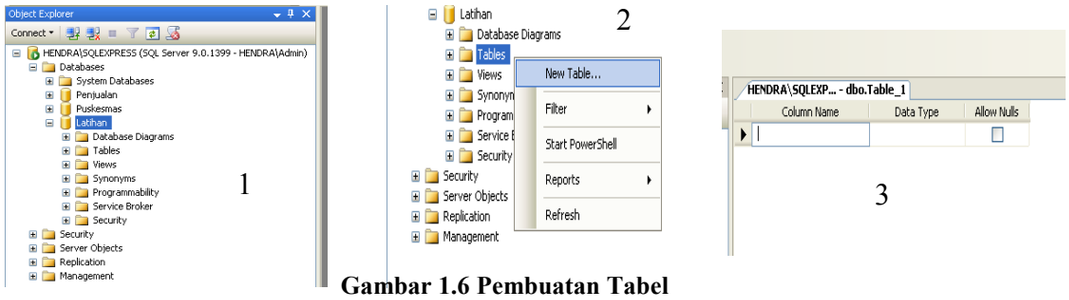


Gambar 1.5 Hasil dari Pembuatan *database*

3. MEMBUAT TABLE TANPA QUERY

Persiapan dalam pembuatan tabel sebagai berikut:

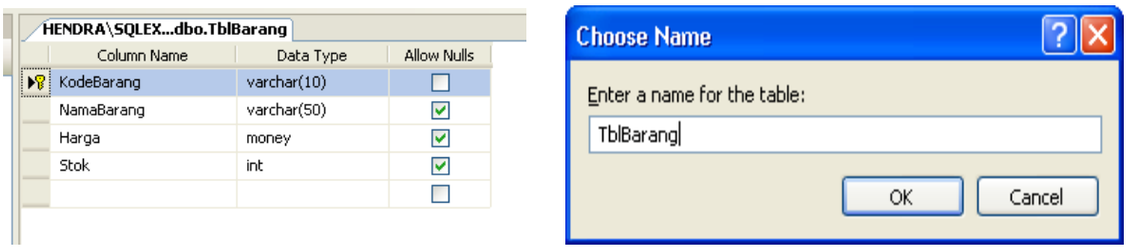
1. Pastikan anda telah membuat *database.* Lalu pilih *database* dimana tabel tersebut akan dibuat pada kotak *Object Explorer*dengan mengklik tanda **(+),** untuk membuka sub-sub pada *database.*



→→

Gambar 1.6 Pembuatan Tabel

2. Pada gambar nomor 3, anda diminta untuk mengisikan *fields* pada tabel yang digunakan untuk menampung data disesuaikan dengan keperluan atau tujuan dari pembuatan tabel tersebut.**1)Column Name,** untuk memasukkan nama*fields,* **2) Data Type,** untuk menentukan tipe dari data yang akan dimasukkan pada *field*tersebut.**3) Allow Nulls,** jika anda checklist pada kotak, maka instruksi terhadap data tersebut adalah diizinkan untuk diisi kosong, namun sebaliknya jika kotak tidak di checklist.
3. Jika pembuatan *fields*, tipe data dan allow null selesai, maka selanjutnya penentuan/pemberian kunci pada salah satu *fields*yang dirasa memiliki keunikan. (**Note**: tidak setiap tabel harus diberikan kunciyang bersifat ***primary key,*** tergantung kebutuhan, namun tetap mengikuti kaidah-kaidah pembuatan basisdata yang baik dan benar). Caranya: pilih salah satu *field* lalu klik ikon **kunci** (***Set Primary Key***) pada *toolbar.*
4. Lalu klik Ikon **SIMPAN,** lalu beri nama tabel tersebut sesuai dengan keinginan anda.

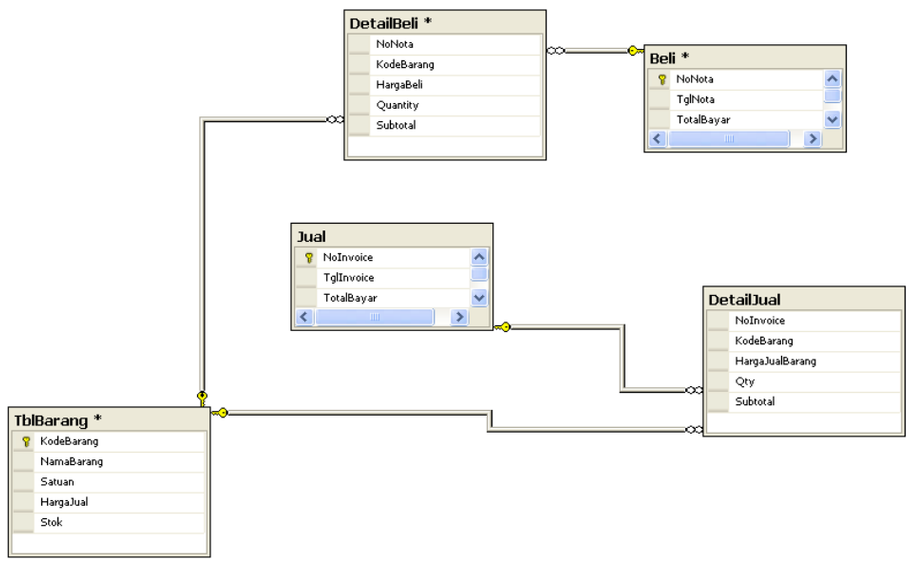


Gambar 1.7Pembuatan dan Penyimpanan Tabel

4. MEMBUAT *DATABASE DIAGRAM*

Untuk membuat *database* diagram syarat yang harus dipenuhi adalah jumlah tabel harus lebih dari satu tabel dan tabel tersebut memiliki hubungan/*relationship*berdasarkan fungsi/kegunaannya masing-masing didalam pemenuhan kebutuhan penyimpanan data untuk suatu kasus tertentu. Adapun cara pembuatan *database* diagram sebagai berikut :

a. Klik tanda (+) pada database yang akabn dibuatkan *database* diagramnya. Lalu pada tabel tersebut terdapat folder *database diagram*.
b. Klik kanan pada folder tersebut lalu ***New Database Diagram***. Maka akan tampil kotak dialog **Add Table.**Lakukan add pada seluruh tabel yang ada, sampai kotak dialog add table kosong → Close
c. Lakukan *drag and drop* pada tabel yang berelasi (Tabel Sumber → Tabel Tujuan). Contoh: KodeBarang(TblBarang) → KodeBarang(DetailJual) / KodeBarang(DetailBeli) Artinya tabel yang memiliki kunci yang bersifat sebagai kunci utama memiliki hubungan dengan tabel yang lain namun kunci tersebut bersifat *foreign key*.



Gambar 1.8 Hasil dari Pembuatan *Database Diagram*

5. MEMBUAT *DATABASE* DENGAN *QUERY*

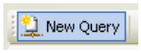
Aturan Penulisan:

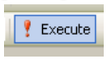
```
CREATE DATABASE Kirana
     ON

(name=Kirana,filename="D:\DATAMHS\Latihan\Kirana.mdf",Size=10,maxsize=100,
filegrowth=2)
     LOG ON
    (name=Kirana _log,filename="
D:\DATAMHS\Latihan\Kirana.ldf",Size=4,maxsize=100,filegrowth=2)
```

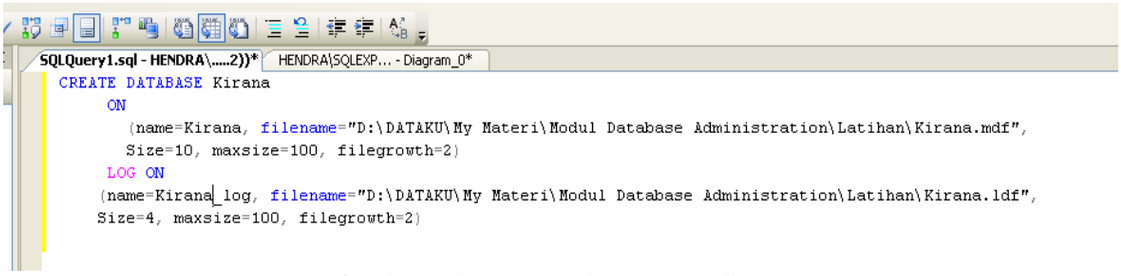
Keterangan :
Kirana → Nama *database* yang dibuat;

Name=Kirana → Nama *database* yang akan digunakan untuk transaksi SQL;
Filename=Kirana.mdf →nama*file* yang disimpan dan ditulis beserta *path*-nya dan berekstensi .mdf;
Size → ukuran *file* dalam *harddisk*, bisa menggunakan KB, MB, GB.*Default*nya adalah MB;
Maxsize → Besar kapasitas maksimal untuk *database* tersebut;
Filegrowth →besarnya penambahan kapasitas saat dibutuhkan.

Untuk menuliskan *query* tersebut, anda cukup dengan mengklik ***New Query*** 

pada *toolbar*.Lalu ketikkan sintaks dibawah ini lalu ***Execute*** 

lalu anda lihat hasilnya pada folder *database*(*Refresh* terlebih dahulu).

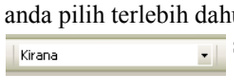


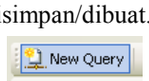
Gambar 1.9 *Query* pembuatan *Database*

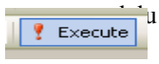
6. MEMBUAT*TABLE* DENGAN *QUERY*

Aturan Penulisan:

```
CREATE TABLE Table_Name
(
   <Column_name>, <data_type>[null | not null]
   [Constraint primary key>]
   [Identity (Nilai_Awal,Nilai_Akhir]
);
```

Untuk menuliskan *query* tersebut, anda pilih terlebih dahulu nama*database*tempat table akan disimpan/dibuat. Anda pilih *database*pada toolbar 

setelah itu anda klik anda ***New Query*** 

pada*toolbar*. Lalu ketikkan sintaks dibawah ini lalu ***Execute*** 

u anda lihat hasilnya pada folder *database* (*Refresh* terlebih dahulu).

```
Create table jurusan
(
   KodeJurusan Char(2)Primary key not null,
   Jurusan Varchar(50) not null
)

Create table Mahasiswa
(
   NPM Varchar(12)Primary key not null,
   NamaMahasiswa Varchar(50)not null,
   KodeJurusan Char(2)Foreign Key references Jurusan(KodeJurusan)
   On delete cascade on update cascade
   )
```

7. TIPE DATA SQL SERVER 2008

Tipe data digunakan ketika membuat suatu tabel, pemakai harus menentukan informasi tertentu pada saat pembuatan tabel. Ada beberapa tipe data yang disimpan dalam basis data, diantaranya adalah :

INTEGER	KETERANGAN
BIT	Integer dengan niali 0 atau 1
INT	Nilaiintergerantara–2.147.483.648 s/d 2.147. 483.647
SMALLINT	Nilai antara –32. 768 s/d 32. 767
TINYINT	Nilai antara 0 s/d 255
DECIMAL / NUMERIC	Akurasi angka tetap antara –10^38-1s/d 10^38-1
MONEY	Data moneter dari –2^63 s/d 2^63-1, dengan tingkat

SMALLMONEY	akurasi hingga sepersepuluh ribu unit moneter -214.748,3648 s/d 214.748,3647 dengan tingkat akurasi sepersepuluh ribu unit moneter
FLOAT	-1.79E+308 s/d 1.79E+308 -3.40E+38 s/d 3.40E+38
REAL	1 januari 1753 s/d 31 desember 9999
DATETIME	1 januari 1900 s/d 6 juni 2079
SMALLDATETIME	

RUPA -RUPA	KETERANGAN
CURSOR	Referensi ke kursor
TIMESTAMP	Angka eklusif yang dikenali oleh basas data
UNIQUEIDENTIFIER	Pengenal global yang eklusif

STRING	KETERANGAN
CHAR	Field tetap dengan ukuran max 8000 byte
VARCHAR	Field tetap dengan ukuran max 8000 byte
TEXT	Variabel dengan ukuran 2^31-1 byte

UNICODE STRING	KETERANGAN
NCHAR	Karakter unicode dengan ukuran 4000 byte
NVARCHAR	Karter unicode dengan ukuran berfariasi 4000 byte
NTEXT	Variaber berukuran 2^30-1 byte

BINARY STRING	KETERANGAN
BINARY	Ukuran tetap hingga 8000 byte
VARBINARY	Ukuran bervariasi 8000 byte
IMAGE	Ukuran bervariasi hingga 2^31-1 byte

PENGANTAR TRANSACT SQL

Transact SQL atau sering juga disebut dengan T-SQL merupakan pengembangan SQL sebagai *database query* dan bahasa pemrograman yang digunakan untuk mengakses *database,* melakukan *query, update database,* serta mengelola hubungan *system database.*SQL Server memiliki tiga jenis Transact-SQL yaitu [1)] DDL, [2)] DML dan [3)] DCL.

***DATA DEFINITION LANGUAGE* (DDL)**

Data definition language digunakan untuk membangun objek, seperti *databases, tables, views.*Yang termasuk didalam perintah DDL adalah sebagai berikut:

a. ***CREATE,*** digunakan untuk membuat *database, table, etc.*
b. ***ALTER,***digunakan untuk memodifikasi struktur *tables, views, etc*
c. ***DROP,***digunakan untuk perintah menghapus *database, table*

ALTER

Alter dapat digunakan untuk memodifikasi sesuatu. Alter bisa digunakan untuk memodifikasi tabel, prosedur, trigger, view.Pada modul ini kita akan membahas alter pada tabel. Perintah alter yang sering digunakan antara lain pada:

a. **ADD,** digunakan untuk menambahkan kolom pada suatu tabel.
   ALTER TABLE *Table_Name* **ADD** *Column_Name Data_Type*

```
ALTER TABLE JURUSAN ADD KELAS CHAR(2)ATAU

ALTER TABLE JURUSAN ADD KELAS1 CHAR(2), JUR CHAR(3)
```

b. DROP COLUMN, digunakan untuk menghapus kolom pada suatu tabel atau untuk menghapus tabel dan *database*.

ALTER TABLE *Table_Name* **DROP COLUMN***Column_Name*

```
ALTER TABLE JURUSAN DROP COLUMN JUR ATAU
ALTER TABLE JURUSAN DROP COLUMN JUR, JUR1
```

c. ALTER COLUMN, digunakan untuk mengubah karakter suatu kolom pada tabel.
`ALTER TABLE JURUSAN ALTER COLUMN KELAS CHAR(1)→ TIPE DATA AWAL CHAR(2)`

DROP

DROP, digunakan untuk menghapus table dan *database*

DROP TABLE *Table_Name*	**DROP DATABASE** *Database_Name*
`Drop table Jurusan`	`Drop database Kirana`

LATIHAN DAN PENGAMBILAN NILAI

a. Buat *database* dengan nama Akademik, lalu buat tabel seperti dibawah ini:

TblGuru

Column Name	Data Type	Constraint
Nip	Varchar(18)	Primary Key, Not Null
Nama	Varchar(50)	Not Null
JenisKelamin	Char(1)	Not Null
TempatLahir	Varchar(50)	Not Null
TanggalLahir	Datetime	Not Null

TblMataPelajaran

Column Name	Data Type	Constraint
KdMapel	Varchar(12)	Primary Key, Not Null
Mapel	Varchar(50)	

TblKelas

Column Name	Data Type	Constraint
Kelas	Char(2)	Primary Key, Not Null
Nip	Varchar ()	Foreign Key, Not Null
Kapasitas	Int	Not Null
Jumlah	Int	Not Null

TblJadwal

Column Name	Data Type	Constraint
NIP	Varchar(12)	Primary Key, Not Null
KdMapel	Varchar(50)	Primary Key, Not Null
Hari	Char(6)	Not Null

b. Buat *database diagram* untuk 4 (empat) tabel diatas dan diberi nama Akademik

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :
Nilai

(……..)
NIK. _____

DATA MANIPULATION LANGUAGE (DML) (INSERT, UPDATE, DELETE)

CHAPTER 7

Data Manipulation Language **(DML)** Adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan, dan penghapusan data. Perintah yang termasuk dalam kelompok DML adalah SELECT, INSERT, DELETE, dan UPDATE. Berikut **statement DML :**

1. **INSERT** : Perintah yang digunakan untuk memasukkan data ke table.
2. **UPDATE** : Perintah yang digunakan untuk memodifikasi data pada tabel.
3. **DELETE** : Perintah yang digunakan untuk menghapus data pada tabel.

INSERT

Insert, merupakan *query* yang digunakan untuk memasukkan data atau baris baru ke dalam tabel. Ketentuan yang harus diperhatikan dalam penggunaan INSERT adalah sebagai berikut:

a. Jika type data seperti (varchar, char, text, etc), maka pada penulisannya harus menggunakan tanda petik tunggal (‘ . . . . . ‘)
b. Jika tipe data numeric (int, numerical, money, decimal), maka tidak boleh menggunakan tanda petik tunggal.
c. Jika kolom bersifat not null, maka harus disertakan.
d. Jika kolom bersifat null, maka harus tidak harus disertakan.
e. Jika kolom bersifat autoincrement/autonumber, maka sebaiknya data tidak diisi.

Ada dua cara memasukkan data:

1. Tanpa menyertakan daftar kolom pada penulisannya, dengan ketentuan tidak ada kolom *auto number* atau *auto complete* dalam tabel tersebut dan daftar nilai yang diisikan harus urut sesuai urutan kolom.

```
Insert Into Nama_Tabel Values (daftar_nilai)
```

Contoh:

```
Insert Into Jurusan values ('SI','Sistem Informasi')atau
Insert Into Mahasiswa values ('03050006','Hendra
Kurniawan','25-08-1981','Jl. Belitung','0891323233')
```

2. Menyertakan daftar kolom

```
Insert Into Nama_Tabel (field1, field n) Values (daftar_nilai)
```

Contoh:

```
Insert Into Jurusan (KodeJurusan, Jurusan) values ('SI','Sistem
Informasi')
```

atau

```
Insert Into Mahasiswa (NPM, NamaMahasiswa, Alamat, NoHp) Values
('03050006','Hendra Kurniawan','25-08-1981','Jl.
Belitung','0891323233')
```

UPDATE

Update, digunakan untuk mengubah data pada sebuah tabel.*Update* terdiri dari dari ***Update All*** atau ***Update*** **dengan kondisi.** Aturan penulisannya:

Update Nama_Tabel **set** kolom_1=Nilai_1, Kolom_n=Nilai_n (*Update All Column*)
Contoh:

Update Jurusan set Jurusan='Sistem Informasi'
Perintah ini akan merubah seluruh kolom/*fields*jurusan menjadi Sistem Informasi

Atau

Update Nama_Tabel set kolom_1=Nilai_1, Kolom_n=Nilai_n where kondisi (*Update dengan kondisi*)

Contoh:

Update Mahasiswa set Nama='Kirana',Alamat='Metro'Where NPM='03050006'

Perintah ini akan merubah kolom/*fields*nama dan alamat menjadi kirana dan Metro, namun hanya untuk yangNPM='03050006'

DELETE

DELETE,merupakan *query* yang digunakan untuk menghapus data pada sebuah tabel. Aturan penulisan:

Delete from nama_tabel Where Kondisi

Contoh:

Delete From Jurusan Where KodeJurusan='SI'

Perintah ini akan menghapus data dengan kondisi KodeJurusan='SI'

(Tambahan:Jika anda ingin menghapus seluruh tabel, anda bisa menggunakannya. Tetapi tanpa menggunakan **Where**)

Delete From Jurusan(Perintah ini akan menghapus seluruh data pada tabel jurusan tanpa kondisi apapun)

LATIHAN DAN PENGAMBILAN NILAI

a. Masih dengan *database* dan *table* sebelumnya. Lakukan perintah ***Insert*** seperti tabel dibawah ini.

TblGuru

NIP	Nama	Jenis Kelamin	Tempat Lahir	Tanggal Lahir
1234567891011121314	Hendra Kurniawan	L	Tanjung Karang	25-08-1982
1234567891110121314	Kirana	P	Metro	11-09-2011
1234567891012111314	Anggorowati	P	Metro	18-06-1985
1234567891410111213	Kenzi	L	Metro	25-08-2014

Lalu gunakan perintah***Update*** untuk merubah kolom/*fields*dengan hasil seperti tabel dibawah ini. (Note : Hasil query anda simpan dan laporkan pada Dosen untuk dinilai)

TblGuru

NIP	Nama	Jenis Kelamin	Tempat Lahir	Tanggal Lahir
1234567891011121314	Kirana	P	Tanjung Karang	11-09-2011
1234567891110121314	Hendra Kurniawan	L	Metro	25-08-1982
1234567891012111314	Kiki Anggorowati	P	Hadimulyo Barat	18-06-1985
1234567891410111213	Kenzi Himura	L	Metro Pusat	25-08-2014

Lalu gunakan perintah ***Delete*** untuk menghapus kolom/*fields*dengan hasil seperti dibawah ini. (Note : Hasil query anda simpan dan laporkan pada Dosen untuk dinilai)

TblGuru

NIP	Nama	Jenis Kelamin	Tempat Lahir	Tanggal Lahir
1234567891011121314	Kirana	P	Tanjung Karang	11-09-2011
1234567891012111314	Kiki Anggorowati	P	Hadimulyo Barat	18-06-1985

Lalu gunakan perintah***Insert*** untuk memasukkan nilaidengan hasil seperti dibawah ini. (Note : Hasil query anda simpan dan laporkan pada Dosen untuk dinilai)

TblGuru

NIP	Nama	Jenis Kelamin	Tempat Lahir	Tanggal Lahir
1234567891011121314	Hendra Kurniawan	L	Tanjung Karang	25-08-1982
1234567891110121314	Kirana	P	Metro	11-09-2011
1234567891012111314	Anggorowati	P	Metro	18-06-1985
1234567891410111213	Kenzi	L	Metro	25-08-2014

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA MANIPULATION LANGUAGE (DML)

(SELECT, WHERE, AND, OR, NOT, DISTINCT, BETWEEN, KOLOM ALIAS, LIKE, SORTING, GROUPING, JOIN)

CHAPTER 8

SELECT

SELECT, merupakan *query* yang digunakan untuk memilih atau mengambil atau menampilkan atau menyeleksi data dari satu atau banyak baris, kolom dari satu atau banyak tabel.Aturan penulisan SELECT adalah sebagai berikut:

```
Select daftar_kolom From Nama_Tabel Where Kondisi (satu tabel)
atau

Select nama_tabel1.daftar_kolom1, nama_tabel2.daftar_kolom1 From
Nama_Tabel1,Nama_Tabel2 Where Nama_Tabel1.kolom=Nama_Tabel2.kolom
```

(**Note:** apabila anda ingin menyeleksi/memilihkolom dari dua tabel yang berbeda, maka nama tabel dituliskan beserta nama kolom

Aturan penulisan SELECT:

1. Kolom yang ingin ditampilkan tidak perlu berurutan
2. Jika semua kolom ingin ditampilkan, maka daftar kolom cukup diwakili dengan tanda (*)
   Select * from nama_tabel
3. Pengurutan data bersifat ascending (kecil → besar) atau descending (besar → kecil)

'Diskusikan dengan dosen anda perintah ini agar anda lebih mengerti'

WHERE

Klausa ***WHERE,*** digunakan untuk menyatakan kondisi yang harus dipenuhi oleh data pada suatu record data yang akan dipilih oleh *query* SELECT. Aturan penulisan WHERE adalah sebagai berikut:

```
Select daftar_kolom From Nama_Tabel Where Kondisi
```

Operator-operator yang akan sering digunakan dalam Where sebagai berikut:

Operator	Arti	Operator	Arti
=	Sama dengan	<	Lebih kecil
<>	Tidak sama dengan	>=	Lebih besar atau sama dengan
>	Lebih besar	<=	Lebih kecil atau sama dengan
!>	Tidak lebih dari	!<	Tidak kurang dari
!=	Tidak sama dengan	Like	Mencari bagian yang sama
Between	Diantara dua nilai		

a. **Operator And, Or**, digunakan untuk membandingkan kondisi pada klausa Where. **Not,** digunakan untuk membalikkan nilai dari kondisi yang dicari.

```
Contoh:
Select * From Jurusan Where KodeJurusan='SI' And KodeJurusan='TI' atau
Select * From Jurusan Where KodeJurusan='SI'OrKodeJurusan='TI'  atau
Select * From Jurusan Where NotKodeJurusan='SI'
```

b. **DISTINCT**, digunakan untuk menyaring data ganda/lebih dari satu baris yang sama ditampilkan menjadi satu baris saja.

Contoh:

```
Select Distinct (KodeJurusan) From Jurusan
```

c. **BETWEEN**, digunakan untuk mengecek suatu nilai diantara range nilai yang diberikan atau ditentukan. BETWEEN biasanya seringkali digunakan untuk mengecek nilai yang bertipe angka.

Contoh:

```
Select * from Penjualan Where TotalPenjualan Between 100000 and 120000
```

KOLOM ALIAS

KOLOM ALIAS, digunakan untuk memberikan nama lain pada kolom. Alias ditulis setelah nama kolom atau ekspresi yang akan diganti atau diberi nama dengan didahului menggunakan *keyword 'AS'* sebagai perintah ganti. Aturan penulisan sebagai berikut:

```
Select KodeJurusan as [Kode Jurusan], jurusan as [Nama Jurusan] from
Jurusan

(Note: Jika kolom ingin diberi spasi, maka gunakan tanda [])
```

LIKE

LIKE,digunakan untuk mencari data berdasarkan karakter per karakter disesuaikan dengan kebutuhan pencarian.Dengan menggunakan kondisi **LIKE**, karakter apapun dapat dicari, dapat berupa satu, dua dan semua karakter yang ada didalam string yang dicari.Untuk mendapatkan karakter yang diinginkan maka kita dapat menggunakan karakter WILDCARD. WILDCARD berfungsi untuk mewakili karakter yang akan dicari sesuai query yang ada. Dengan menggunakan karakter wildcard dapat membuat operator/kondisi LIKE lebih fleksibel daripada menggunakan operator pembanding string – atau !=.

Berikut ini macam karakter wildcard yang dapat digunakan untuk membandingkan data yang akan dicari.

1. **Karakter %**
   Karakter % digunakan untuk mempresentasikan semua karakter yang belum diketahui sebelum, ditengah atau sesudah karakter yang akan dicari.Aturan penulisan sebagai berikut:

```
Select KodeJurusan fromJurusanwhere Jurusan like 'SI%' atau
Penjelasan: mencari kode Jurusan dengan karakter awal S
Select KodeJurusan from Jurusan where KodeJurusan like '%S' atau
Penjelasan: mencari kode Jurusan dengan karakter kedua S
Select KodeJurusan from Jurusan where KodeJurusan like '%S%'
Penjelasan: mencari kode Jurusan dengan karakter ditengah S

(Note: Jika kolom ingin diberi spasi, maka gunakan tanda []

'Diskusikan dengan dosen andaperintah ini agar anda lebih mengerti'
```

2. **Karakter _ (Underscore)**
   Karakter _ digunakan untuk mempresentasikan satu karakter yang akan dicari. Aturan penulisan sebagai berikut:

```
Select KodeJurusan from Jurusan where Jurusan like '_A%' atau
Select KodeJurusan from Jurusan where Jurusan like 'A_%' atau
```

SORTING

SORTING, digunakan untuk mengurutkan data hasil dari SELECT. Klausa yang digunakan untuk proses pengurutan (*Sorting)* adalah **ORDER BY.** Penggurutan dapat dilakukan dengan ascending dan descending. Aturan penulisan sebagai berikut:

```
Select KodeJurusan, Jurusan from Jurusan Order By Jurusan Asc
Select KodeJurusan, Jurusan from Jurusan Order By Jurusan Desc
```

GROUPING

GROUPING, digunakan untuk mengelompokkan data berdasarkan daftar kolom yang disebutkan dalam perintah SELECT. Aturan penulisan sebagai berikut:

```
Select KodeJurusan, Jurusan
from Jurusan
Group By Jurusan
Order By Jurusan Asc
```

HAVING

HAVING, digunakan untuk membatasi hasil mengelompokkan data berdasarkan daftar kolom yang disebutkan dalam perintah SELECT. Aturan penulisan sebagai berikut:

```
Select *
From Penjualan
Group By NoInvoice
Order By NoInvoice Asc
Having TotalPenjualan>120000
```

JOIN

JOIN, merupakan operasi yang digunakan untuk mendapatkan data gabungan dari dua tabel atau lebih. Join dipakai untuk memperoleh data-dat yang lebih detail dari tabel-tabel yang saling memiliki hubungan. Perintah join bisanya berada/digunakan dalam perintah *Select.*Join memiliki tiga jenis operasi, antara lain:

a. ***Cross Join,***menghasilkan kombinasi semua baris yang terdapat pada tabel-tabel yang di-join, baik yang berpasangan, maupun yang tidak berpasangan. Join ini pada kenyataannya tidak digunakan. Contoh (ada 2 tabel jurusan dan mahasiswa)

Tabel Jurusan **Tabel Mahasiswa**

Kode_Jurusan	Jurusan	NPM	Kode_Jurusan
SI	Sistem Informasi	03050006	SI
MI	Manajemen Informatika	03050036	MI

Select * From Jurusan Cross Join Mahasiswa→ **hasilnya:**

Kode_Jurusan	Jurusan	NPM	Kode_Jurusan
SI	Sistem Informasi	03050006	SI
MI	Manajemen Informatika	03050006	SI
SI	Sistem Informasi	03050036	MI
MI	Manajemen Informatika	03050036	MI

b. ***Inner Join,*** menghasilkan semua baris yang bernilai samayang terdapat pada tabel-tabel yang di-joinkan.Aturan Penulisan:

Select Jurusan.Jurusan, Mahasiswa.NPM From Jurusan
INNER JOIN Mahasiswa ON Jurusan.KodeJurusan=Mahasiswa.KodeJurusan

Tabel Jurusan **Tabel Mahasiswa**

Kode_Jurusan	Jurusan	NPM	Kode_Jurusan
SI	Sistem Informasi	03050006	SI
MI	Manajemen Informatika	03050036	MI
TI	Teknik Informatika		

Tabel Mengajar

NPM	Ruang
03050006	F.3.6

Hasilnya:

Jurusan	NPM
Sistem Informasi	03050006
Manajemen Informatika	03050036

Select Jurusan.Jurusan, Mahasiswa.NPM, Mengajar.RuangFrom Jurusan
INNER JOIN Mahasiswa ON Jurusan.KodeJurusan=Mahasiswa.KodeJurusan
INNER JOIN MENGAJAR ON Mahasiswa.NPM=Mengajar.NPM

Hasilnya:

Jurusan	NPM	Ruang
Sistem Informasi	03050006	F.3.6

c. ***Outer Join,*** sama dengan *Inner Join,* bedanya adalah pada *outer join* menampilkan data yang memiliki pasangan maupun tidak memiliki pasangan. Sedangkan *inner join* hanya untuk yang berpasangan saja.*Outer join* terbagi menjadi 3 jenis, yaitu:

1. ***Left Outer Join****,* data yang ada pada tabel sebelah kiri akan ditampilkan semua sesuai dengan pasangannya, jika tidak ada pasangannya maka tabel sebelah kanan akan diisikan null.
2. ***Right Outer Join****,* data yang ada pada tabel sebelah kananakan ditampilkan semua sesuai dengan pasangannya, jika tidak ada pasangannya maka tabel sebelah kiri akan diisikan null.
3. ***Full Outer Join****,* semua data dari tabel-tabel yang di-join ditampilkan, baik itu yang secara *left outer join, right outer join, inner join*.

Contoh :(berdasarkan data pada ketiga tabel diatas)

a. ***Left Outer Join***

```
Select Jurusan.Jurusan, Mahasiswa.NPM From Jurusan
LEFT JOIN Mahasiswa ON Jurusan.KodeJurusan=Mahasiswa.KodeJurusan
```

Hasilnya:

Jurusan	NPM
Sistem Informasi	03050006
Manajemen Informatika	03050036
Teknik Informatika	Null

b. ***Right Outer Join***

```
Select Jurusan.Jurusan, Mahasiswa.NPM From Jurusan
RIGHT JOIN Mahasiswa ON Jurusan.KodeJurusan=Mahasiswa.KodeJurusan
```

Hasilnya:

Jurusan	NPM
Sistem Informasi	03050006
Manajemen Informatika	03050036

c. ***Full Outer Join***

```
Select Jurusan.Jurusan, Mahasiswa.NPM From Jurusan
FULL JOIN Mahasiswa ON Jurusan.KodeJurusan=Mahasiswa.KodeJurusan
```

Hasilnya:

Jurusan	NPM
Sistem Informasi	03050006
Manajemen Informatika	03050036
Teknik Informatika	Null

LATIHAN DAN PENGAMBILAN NILAI

a. Buat database dan beri nama INNER_NPM dan table seperti dibawah ini:

TblGuru

NIP	Nama	KodeJabatan
1234567891011121314	Hendra Kurniawan	01
1234567891110121314	Kirana	02
1234567891012111314	Anggorowati	03
1234567891410111213	Keola	03

TblJabatan

KodeJabatan	Jabatan
01	Kepala Sekolah
02	Wakil Kepala Sekolah
03	Guru
04	Guru BP
05	Konseling

b. Seleksi Nama dan Jabatan dengan parameter kode jabatan="01"
c. Seleksi nama dan kodejabatan dengan parameter nama="kirana"
d. Lakukan order by dan gouping untuk tabel guru dan jabatan
e. Lakukan perintah inner join, left outer join, right join, full join.
f. Lakukan perintah distinct untuk mengeksekusi dua tabel tersebut.
g. Tampilkan data dengan nilai data '01 dan '02
h. Tampilkan data dengan nilai data '01 atau '02
i. Tampilkan data dengan nilai data selain '01 dan '02

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA MANIPULATION LANGUAGE (DML)

(AGREGATE, DATE AND TIME, MATHEMATICAL, STRING)

CHAPTER 9

AGREGATE

AGREGATE terdiri dari fungsi-fungsi statistik seperti:

1. ***Avg (Average),*** yaitu fungsi untuk menghitung nilai rata-rata;
2. ***Count,*** yaitu berfungsi untuk menghitung jumlah baris data;
3. ***Max (Maximum),*** yaitu berfungsi untuk mengembalikan nilai maksimum dari sejumlah baris data;
4. ***Min (Minimum),*** yaitu berfungsi untuk mengembalikan nilai minimum dari sejumlah baris data;
5. ***Stdev,*** yaitu berfungsi untuk menghitung standar deviasi dari sejumlah data;
6. ***Sum,*** yaitu berfungsi untuk menghitung jumlah nilai dari baris data yang dipilih;
7. ***Var,*** yaitu berfungsi untuk menghitung nilai varian dari sejumlah data;

Untuk mengetahui fungsi-fungsi diatas, buatlah sebuah tabel dan diisi dengan data tersebut:

```
Create Table Score{
     Id Char(4), Score Int
}
go

Insert Into Score values (‘0001’,’85)
Insert Into Score values (‘0002’,’92)
Insert Into Score values (‘0003’,’100)
Insert Into Score values (‘0004’,’25)
Insert Into Score values (‘0005’,’40)
Insert Into Score values (‘0006’,’15)
Insert Into Score values (‘0007’,’56)
Insert Into Score values (‘0008’,’90)
Insert Into Score values (‘0009’,’74)
Insert Into Score values (‘0010’,’35)
Insert Into Score values (‘0011’,’73)
Insert Into Score values (‘0012’,’97)
Insert Into Score values (‘0013’,’50)
Insert Into Score values (‘0014’,’20)
Insert Into Score values (‘0015’,’39)
Insert Into Score values (‘0016’,’10)
Insert Into Score values (‘0017’,’26)
Insert Into Score values (‘0018’,’80)
```

Gunakan perintah select berikut untuk mengetahui hasilnya:

```
Select * From Score
```

Id	Score
0001	85
0002	92
0003	100
0004	25
0005	40
0006	15
0007	56
0008	90
0009	70

```
0010 35
0011 73
0012 97
0013 50
0014 20
0015 65
0016 10
0017 26
0018 80
```

AVG

Untuk melihat rata-rata dari semua data diatas, fungsi agregate yang dipergunakan adalah avg, yaitu sebagai berikut:

```
Select avg (Score) From Score
```

```
Hasil :
55      (Silakan anda coba dengan fungsi diatas)
```

Penggunaan klausa *where* untuk mendapatkkan score rata-rata untuk range score tertentu. Sintaknya adalah sebagai berikut:

```
Select avg (Score) From Score
Where Score > 50
```

```
Hasil :
83(Silakan anda coba dengan fungsi diatas)
```

COUNT

Digunakan untuk menghitung jumlah baris data pada suatu tabel. Aturan penulisan:

```
Select count (*) From Score

Atau

Select count (Id) From Score

Atau

Select count (Score) From Score
Where Score >50 and Score <90
```

Perbedaan antara count menggunakan (*) dengan count menggunakan (*field)* adalah bahwa ketika *field* tersebut bernilai kosong, maka fungsi count(*) akan tetap menghitung nilai data yang ada. Tidak sebaliknya dengan count(*field),* maka data yang bernilai null tidak akan dihitung oleh fungsi tersebut.

Untuk lebih jelasnya mari kita coba pada tabel score, tambahkan satu record data, tetapi score kita kosongkan.

```
Insert Into Score values('0019',null)→ lalu ketik dibawah,

Select * from score

Id   Score

0001  85
0002 92
0003 100
0004 25
0005 40
0006 15
```

```
0007 56
0008 90
0009 70
0010 35
0011 73
0012 97
0013 50
0014 20
0015 65
0016 10
0017 26
0018 80
0019 Null
```

Langkah selanjutnya adalah anda tulis query dibawah ini:

```
Select count(*) from Score
```

Hasil :
19

```
Select count(Score) from Score
```

Hasil :
18

MAX

Digunakan untuk mengembalikan nilai maksimum dari sejumlah baris data. Aturan penulisan sebagai berikut:

```
Select Max(Score) from Score
```

Hasil :
100

MIN

Digunakan untuk mengembalikan nilai minimum dari sejumlah baris data. Aturan penulisan sebagai berikut:

```
Select Min(Score) from Score
```

Hasil :
10

STDEV (Standar Deviation)

Digunakan untuk menghitung nilai standar deviasi dari sejumlah data. Aturan penulisan sebagai berikut:

```
Select StDev(Score) from Score
```

Hasil :
30.7503*******

SUM

Digunakan untuk menghitung jumlah nilai dari*field* yang dipilih. Aturan penulisan sebagai berikut:

```
Select Sum(Score) from Score
```

Hasil :

1007

VAR

Digunakan untuk menghitung nilai varian dari sejumlah data yang dipilih. Aturan penulisan sebagai berikut:

```
Select Var(Score) from Score
```

Hasil :
945.584*******

GROUPING

GROUPING, digunakan untuk mengelompokkan data berdasarkan daftar kolom yang disebutkan dalam perintah SELECT dan bisa juga dikombinasikan dengan fungsi *agregate*. Untuk mengetahui cara melakukan grouping dengan kombinasi fungsi *agregate*buat tabel berikut ini:

```
Create Table ScoreClass{
     Id Char(4), Class Char(1),
     Score Int
}
go

Insert Into ScoreClass values ('0001','A','90)
Insert Into ScoreClass values ('0002','B','100)
Insert Into ScoreClass values ('0003','A','80)
Insert Into ScoreClass values ('0004','A','55)
Insert Into ScoreClass values ('0005','B','60)
Insert Into ScoreClass values ('0006','B','85)
Insert Into ScoreClass values ('0007','A','70)
Insert Into ScoreClass values ('0008','C','40)
Insert Into ScoreClass values ('0009','A','45)
Insert Into ScoreClass values ('0010','C','95)

Select * from ScoreClass
```

Id	Class	Score
0001	A	90
0002	B	100
0003	A	80
0004	A	55
0005	B	60
0006	B	85
0007	A	70
0008	C	40
0009	A	45
0010	C	95

Untuk menghitung nilai rata-rata dari masing-masing kelas, gunakan perintah *query*:

```
Select Class, Avg(Score) As [Rata-Rata]
from ScoreClass
Group By Class
```

Hasil:

Class	Rata-Rata
A	68
B	81
C	67

Atau anda juga bisa menggunakan fungsi lainnya:

```
Select Class, Count(*) As Jumlah
from ScoreClass
Group By Class
```

Hasil:

Class	Rata-Rata
A	5
B	3
C	2

HAVING

HAVING, digunakan untuk membatasi hasil mengelompokkan data berdasarkan daftar kolom yang disebutkan dalam perintah SELECT dan bisa dikombinasikan dengan *agregate*. Aturan penulisan sebagai berikut:

```
SelectClass, Count(*)
fromScore
Group ByClass
Order ByClass Asc
Having Count(*)>2
```

Hasil:

Class	Rata-Rata
A	5
B	3

LATIHAN DAN PENGAMBILAN NILAI

a. Buat database dan beri nama *Agregate* dan table seperti dibawah ini:

TblMahasiswa

NPM	Nama	KodeJurusan
03050006	Hendra Kurniawan	SI
03050036	Kirana	MI
03050007	Anggorowati	TI
03050008	Kenzi	MI

TblJurusan

KodeJurusan	Jurusan
SI	Sistem Informasi
MI	Manajemen Informatika
TI	Teknik Informatika
MA	Manajemen

TblNilai

NPM	Nilai
03050006	80
03050036	86
03050007	50
03050007	55
03050006	70
03050036	65
03050007	50
03050008	55

b. Hitung nilai total mahasiswa berdasarkan NPM

c. Hitung jumlah mahasiswa yang mendapatkan nilai >50
d. Hitung nilai rata-rata untuk mahasiswa dengan NPM=’03050007’
e. Cari nilai mahasiswa dengan NPM=’03050036’
f. Tampilkan Mahasiswa per jurusan dengan nilai diatas >75
g. Tampilkan jumlah mahasiswa per jurusan

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA MANIPULATION LANGUAGE (DML)

(DATE AND TIME, MATHEMATICAL, STRING)

CHAPTER 10

DATE AND TIME FUNCTION

Merupakan fungsi-fungsi *built-in*yang digunakan untuk pengolahan data tanggal dan jam.

1. Menampilkan tanggal sistem

```
Select Getdate()
```

Hasil:
03-03-2015 13:13:13

2. Mengambil tanggal sistem

```
Select Day('03-03-2015')
```

Hasil:
03

3. Mengambil bulan sistem

```
Select Month('03-03-2015')
```

Hasil:
03

4. Mengambil tahun sistem

```
Select Year('03-03-2015')
```

Hasil:
2015

5. **DateADD()**, menghasilkan sebuah nilai tanggal baru berdasar pada penambahan dengan suatu interval pada suatu nilai tanggal.

   Cara Penulisan/sintak:

   ```
   DATEADD (datepart ,number,date )
   ```

   Fungsi DATEADD memiliki 3 Argumen:

   datepart *:*Adalah parameter untuk menentukan data apa yang akan ditambah nilainya, misalnya data hari, bulan atau tahun.

   number *:*Adalah nilai angka berapa besar data akan ditambahkan. Jika nilai angka berisi nilai pecahan, maka akan dibulatkan ke bawah. Misalnya angka 1.75 akan dibulatkan menjadi 1.

   date *:*Adalah ekpresi data berupa tipe datatime atau smalldatetime

Format	Fungsi	Contoh	Hasil
“D”	Menambah Tanggal	Select DateAdd(“D”,1,’2015-03-03 12:12:12’)	2015-03-04 12:12:12
“M”	Menambah Bulan	Select DateAdd(“M”,1,’2015-03-03 12:12:12’)	2015-04-03 12:12:12
“YY”	Menambah Tahun	Select DateAdd(“YY”,1,’2015-03-03 12:12:12’)	2016-03-04 12:12:12
“HH”	Menambah Jam	Select DateAdd(“ HH”,1,’2015-03-03 12:12:12’)	2015-03-04 13:12:12
“N”	Menambah Menit	Select DateAdd(“D”,1,’2015-03-03 12:12:12’)	2015-03-04 12:13:12
“S”	Menambah Detik	Select DateAdd(“D”,1,’2015-03-03 12:12:12’)	2015-03-04 12:12:13

Contoh penggunaan fungsi DATEADD() pada SQL Server:

```
SELECT DATEADD(“D”, 5, tglbeli) AS Tanggal_Bayar FROM penjualan;
```

Perintah select ini akan menampilkan data Tanggal_Bayar yang datanya diambil dari field tglbeli ditambah 5 hari. Jika data tglbeli bernilai 03Maret 2015, maka data Tanggal_Bayar yang ditampilkan adalah 8 Maret 2015.

6. **Datediff()**, digunakan untuk mengurangkan tanggal maupun jam. Bentuk dari datediff adalah sebagai berikut:

```
Datediff(format, tanggal_awal, tanggal_akhir)

Contoh :
Select datediff(“D”,’2015-01-31’,’2015-02-15’)

Hasil:
15
```

7. **String Function()**, merupakan fungsi built-in untuk memanipulasi string, diantaranya adalah sebagai berikut:

Format	Fungsi	Contoh	Hasil
Ascii	Nilai Ascii Suatu Karakter	SELECT ASCII (‘A’)	65
Char	Karakter dari suatu nilai ascii	SELECT CHAR (65)	A
Left	Mengambil Karakter dari Kiri	SELECT LEFT(‘KIRANA’,2)	KI
Right	Mengambil Karakter dari Kanan	SELECT RIGHT(‘KIRANA’,2)	NA
Substring	Mengambil Karakter dari Kiri	SELECT SUBSTRING (‘KIRANA’,3,2)	RA
Lower	Mengubah Karakter ke Huruf	SELECT Lower (‘KIRANA’)	kirana

	Kecil		
Upper	Mengubah Karakter ke Huruf Besar	SELECT Upper('kirana')	KIRANA
Replace	Mengganti String didalam String	SELECT Replace ('KINANA','NA','RA')	KIRANA
Ltrim	Menghilangkan String Kosong dari kiri	SELECT Ltrim (' KI')	KI
Rtrim	Menghilangkan String Kosong dari kiri	SELECT Rtrim ('KI ')	KI

1. **Fungsi Matematika,** fungsi matematika merupakan *built in*, diantaranya adalah:

Format	Fungsi	Contoh	Hasil
Abs	Nilai Absolut	SELECT ABS (-12)	12
Cos	Nilai Cosinus	SELECT COS (0)	1
Sin	Nilai Sinus	SELECT SIN (0)	0
Tan	Nilai Tangent	SELECT TAN (0)	0
Sqrt	Akar Kuadrat	SELECT SQRT(9)	3
Square	Kuadrat	SELECTSQUARE (9)	81
Power	Pwer (m, n) m pangkat n	SELECT POWER (9,3)	729
Log	Logaritma Natural (len)	SELECT LOG (2.7)	0.993251773010283
Log10	Logaritma 10	SELECT LOG10 (100)	2
Ceiling	Pembulatan Ke atad	SELECT CEILING (10.2)	11
Floor	Pembulatan Ke bawah	SELECT FLOOR (10.2)	10

LATIHAN DAN PENGAMBILAN NILAI

a. Buat database dan beri nama Tanggal dan table seperti dibawah.
b. Lakukan perintah insert untuk memasukkan data berdasarkan nilai-nilai pada tabel.

TblMahasiswa

NPM	Nama	Tanggal Lahir	Tanggal Sekarang	Nilai
03050006	Hendra Kurniawan	25-08-1982	Gunakan tanggal sekarang	97.3
03050005	Kirana	11-09-2011	Gunakan tanggal sekarang	77
03050007	Anggorowati	18-06-1985	Gunakan tanggal sekarang	89
03050008	Kenzi	25-08-2014	Gunakan tanggal sekarang	90.4

c. Gunakan fungsi alias dan datediff untuk mendapatkan umur berdasarkan tabel diatas.
d. Gunakan fungsi like untuk mendapatkan umur yang inisialnya adalah '_e%'.
e. Gunakan fungsi pembulatan keatas untuk data dala tabel.
f. Gunakan fungsi pembulatan kebawah untuk data dala tabel.
g. Dapatkan nilai string dari kolom nama sebanyak **2 karakter** dan beri nama alias **karakter left** dan dapatkan umurnya.
h. Lakukan perintah **Order By** untuk menurutkan data diatas.
i. Ambil nilai bulan untuk kolom nama dengan inisial 'H'
j. Ambil nilai tahun dan pembulatan keatas untuk inisial 'K'
k. Ambil nilai hari dengan Inisial 'A'.
l. Ambil data untuk kolom nama dengan hasil karakter sebelah kanan sebanyak **2 karakter**

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

DATA MANIPULATION LANGUAGE (DML)
(UNION, INTERSECT, EXCEPT)

CHAPTER 11

UNION

Merupakan fungsi yang dipergunakan untuk menggabungkan hasil select dari beberapa query menjadi satu kesatuan. Aturan penulisannya sebagai berikut:

```
Select Field1,Field2,Field3From NamaTabel1
UNION
Select Field4,Field5,Field6 From NamaTabel2
```

untuk lebih memahami fungsi UNION, maka yang perlu kita persiapkan adalah tabel seperti dibawah ini:

Gunakan perintah **CREATE** dan **INSERT** untuk tabel dibawah ini.

TblMahasiswa

NPM	KodeJurusan
03050006	SI
03050005	TI
03050007	MI
03050008	SI

TblJurusan

KodeJurusan	Jurusan
SI	Sistem Informasi
TI	Teknik Informatika
MI	Manajemen Informatika
SK	Sistem Komputer
TK	Teknik Komputer
MA	Manajemen
AK	Akuntansi

Pada kesempatan ini kita akan menggunakan fungsi UNION sebagai berikut:

```
Select Mahasiswa.* From TblMahasiswa
UNION
Select Jurusan.* From TblJurusan
```

Hasilnya:

```
NPM                  Kode_Jurusan
-----------------------------------
03050006             SI
03050005             TI
03050007             MI
03050008             SI
SI                   Sistem Informasi
MI                   Manajemen Informatika
TI                   Teknik Informatika
SK                   Sistem Komputer
TK                   Teknik Informatika
```

```
MA                      Manajemen
AK                      Akuntansi
```

INTERSECT

Merupakan fungsi yang dipergunakan untuk memberikanhasil select dari beberapa query menjadi satu kesatuan, namun hanya yang memiliki irisan saja. Artinya himpunan data yang sama saja pada himpunan-himpunan irisan tersebut. Argumen yang digunakan terdiri dari IN. Aturan penulisannya sebagai berikut:

```
Select Field1,Field2,Field3 From NamaTabel1
WHERE Ekpresi Field IN
(Select Field4,Field5,Field6 From NamaTabel2)
```

Pada kesempatan ini kita akan menggunakan fungsi UNION sebagai berikut:

```
SELECT     Kode_Jurusan, Jurusan
FROM      Jurusan
WHERE     (Kode_Jurusan IN
           (SELECT Kode_Jurusan From Mahasiswa))
```

Hasilnya:

```
Kode_Jurusan            Jurusan
---------------------------------
SI                      Sistem Informasi
TI                      Teknik Informatika
MI                      Manajemen Informatika
```

EXCEPT

Merupakan fungsi yang dipergunakan untuk memberikan hasil select dari beberapa query menjadi satu kesatuan, namunjika himpunan A yang tidak terdapat pada himpunan B. Artinya himpunan data yang sama saja pada himpunan-himpunan irisan tersebut. Argumen yang digunakan terdiri dari NOT IN. Aturan penulisannya sebagai berikut:

```
Select Field1,Field2,Field3 From NamaTabel1
WHERE Ekpresi Field NOT IN
(Select Field4,Field5,Field6 From NamaTabel2)
```

Pada kesempatan ini kita akan menggunakan fungsi UNION sebagai berikut:

```
SELECT     Kode_Jurusan, Jurusan
FROM      Jurusan
WHERE     (Kode_Jurusan NOT IN
           (SELECT Kode_Jurusan From Mahasiswa))
```

Hasilnya:

```
Kode_Jurusan            Jurusan
---------------------------------
SK                      Sistem Komputer
TK                      Teknik Komputer
MA                      Manajemen
AK                      Akuntansi
```

LATIHAN DAN PENGAMBILAN NILAI

a. Buat database dan beri namaGabung dan table seperti dibawah ini:

TblMahasiswa

NPM	Nama	KodeJurusan
03050006	Hendra Kurniawan	SI
03050036	Kirana	MI
03050007	Anggorowati	TI
03050008	Kenzi	MI

TblJurusan

KodeJurusan	Jurusan
SI	Sistem Informasi
MI	Manajemen Informatika
TI	Teknik Informatika
MA	Manajemen

TblNilai

NPM	Nilai
03050006	80
03050036	86
03050007	50
03050007	55
03050006	70
03050036	65
03050007	50
03050008	55

b. Gunakan fungsi UNION untuk menggabungkan tabel-tabel diatas
c. Tampilkan data pada tabel nilai dan tabel mahasiswa dengan perintah UNION
d. Tampilkan data pada tabel nilai yang ada pada tabel mahasiswa dengan perintah IN
e. Tampilkan data pada tabel jurusan yang ada pada tabel nilai dengan perintah IN
f. Tampilkan data pada tabel jurusan yang tidak ada pada tabel mahasiswa dengan perintah NOT IN.
g. Tampilkan data pada tabel nilai yang ada pada tabel mahasiswa dengan perintah IN dengan nilai diatas 70

***) Pengambilan nilai dilakukan pada saat praktek di laboratorium**

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)
NIK. _____

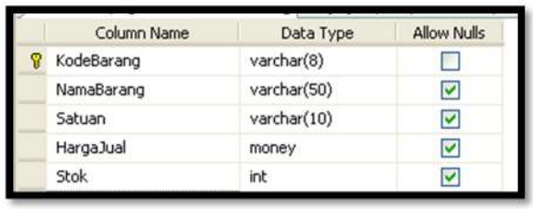
STUDI KASUS

CHAPTER 12

Pada bab sebelumnya kita telah banyak mempelajari dan mempraktekkan fungsi-fungsi SQL *Script* pada SQL Server. Pada bab ini kita akan mengerjakan studi kasus penjualan dan pembelian dengan tetap memanfaatkan konsep-konsep *query* yang telah dipelajari pada bab-bab sebelumnya.

Sebagai contoh kasus buatlah tabel seperti tabel dibawah ini dengan SQL *Script.*

TblBarang

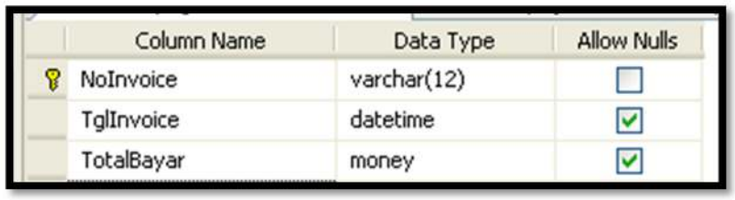


	Column Name	Data Type	Allow Nulls
🔑	KodeBarang	varchar(8)	☐
	NamaBarang	varchar(50)	☑
	Satuan	varchar(10)	☑
	HargaJual	money	☑
	Stok	int	☑

Lalu isikan datanya sebagai berikut:

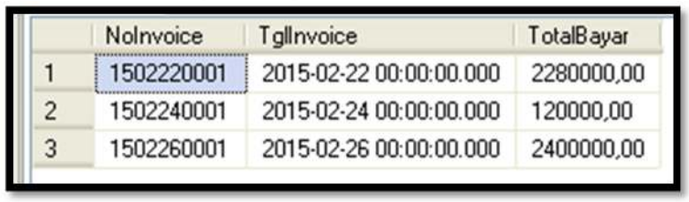
	KodeBarang	NamaBarang	Satuan	HargaJual	Stok
1	B-0001	Monitor LCD 17 Inchi	Buah	7000,00	7902
2	B-0002	PC Dual Core	Buah	6000,00	5881
3	B-0003	Keyboar Wireless	Buah	150000,00	100
4	B-0004	Mouse Wireless	Buah	75000,00	100
5	B-0005	PC Camera	Buah	500000,00	100

Jual



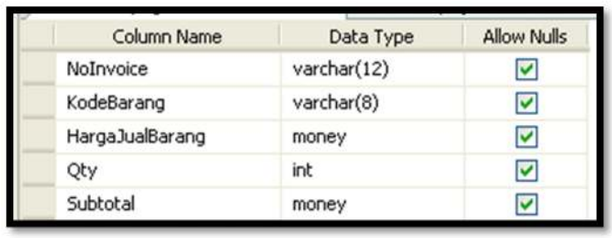
	Column Name	Data Type	Allow Nulls
🔑	NoInvoice	varchar(12)	☐
	TglInvoice	datetime	☑
	TotalBayar	money	☑

Lalu isikan datanya sebagai berikut:



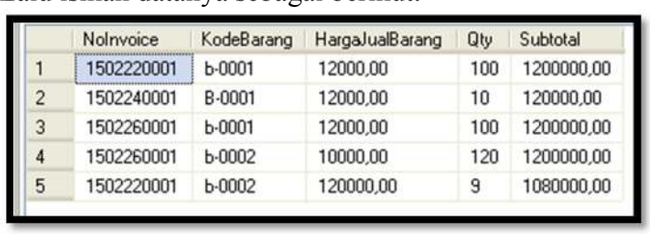
	NoInvoice	TglInvoice	TotalBayar
1	1502220001	2015-02-22 00:00:00.000	2280000,00
2	1502240001	2015-02-24 00:00:00.000	120000,00
3	1502260001	2015-02-26 00:00:00.000	2400000,00

DetailJual



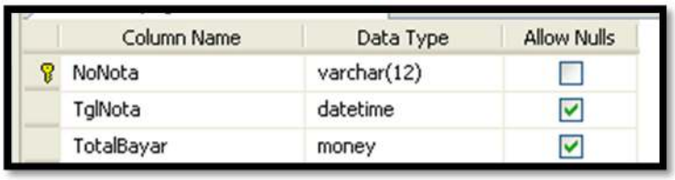
	Column Name	Data Type	Allow Nulls
	NoInvoice	varchar(12)	☑
	KodeBarang	varchar(8)	☑
	HargaJualBarang	money	☑
	Qty	int	☑
	Subtotal	money	☑

Lalu isikan datanya sebagai berikut:



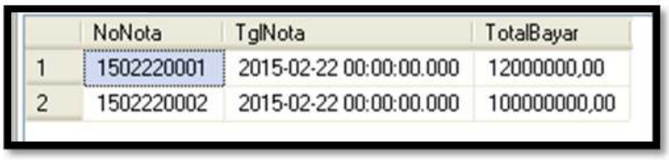
	NoInvoice	KodeBarang	HargaJualBarang	Qty	Subtotal
1	1502220001	b-0001	12000,00	100	1200000,00
2	1502240001	B-0001	12000,00	10	120000,00
3	1502260001	b-0001	12000,00	100	1200000,00
4	1502260001	b-0002	10000,00	120	1200000,00
5	1502220001	b-0002	120000,00	9	1080000,00

Beli



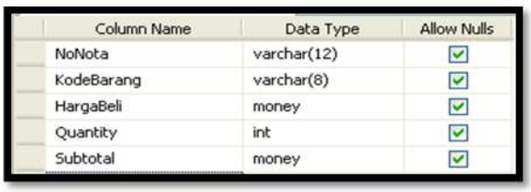
	Column Name	Data Type	Allow Nulls
🔑	NoNota	varchar(12)	☐
	TglNota	datetime	☑
	TotalBayar	money	☑

Lalu isikan datanya sebagai berikut:



	NoNota	TglNota	TotalBayar
1	1502220001	2015-02-22 00:00:00.000	12000000,00
2	1502220002	2015-02-22 00:00:00.000	100000000,00

DetailBeli



Column Name	Data Type	Allow Nulls
NoNota	varchar(12)	☑
KodeBarang	varchar(8)	☑
HargaBeli	money	☑
Quantity	int	☑
Subtotal	money	☑

Lalu isikan datanya sebagai berikut:

	NoNota	KodeBarang	HargaBeli	Quantity	Subtotal
1	1502220001	B-0001	120000,00	100	12000000,00
2	1502220002	B-0001	100000,00	1000	100000000,00

LATIHAN DAN PENGAMBILAN NILAI

a. Buat *database* dan tabel dengan menggunakan SQL Script.
b. Input data pada tabel dengan menggunakan Script SQL (Select).
c. Tampilkan data pembelian barang dengan KodeBarang='B-0001'.
d. Tampilkan seluruh data pembelian.
e. Tampilkan seluruh data penjualan.
f. Tampilkan seluruh data pembelian dengan dikelompokkan berdasarkan NoNota.
g. Tampilkan seluruh data penjualan dengan dikelompokkan berdasarkan NoInvoice.
h. Tampilkan seluruh data pembelian pada bulan-bulan tertentu.
i. Tampilkan seluruh data pembelian pada tahun-tahun tertentu.
j. Tampilkan seluruh data pembelian dengan pengurutan berdasarkan NoNota dan dikelompokkan berdasarkan NoNota.

Diperiksa tanggal : ____ /____ /___ Nama Laboratorium : _____ NPM : _____
Dosen Pengampu :

Nilai

(……..)

DAFTAR PUSTAKA

Abraham Silberschatz, Henry F Korth, S Sudarshan, Database System Concept Fifth Edition – Mc Graw – Hill International Edition;

Connoly, Thomas; Begg, Caroly; Strachan, Anne; *Database System : A Practical Approach to Design, Implementation and Management,* 3rd Edition, Addison Wesley, 2001

_____. Diktat Sistem Basis Data