Sistem Basis Data

Putu Manik Prihatini
gek_anik@yahoo.com

Referensi :

- Wajib :
  - Abraham Silberschatz, Henry F. Korth, *Database System Concepts*, 3rd Edition, McGraw-Hill, 1999
  - Raghu Ramakrisnan, Gherke, *Database Management System,* 3rd Edition, McGraw-Hill, 2001
  - Ramez Elmasri, Sam Navathe, *Fundamentals of Database Systems,* 4th Edition*,* Addison Wesley Publishing Company, 2000
- Penunjang :
  - C.J. Date, *Pengenalan Sistem Basis Data Jilid 1,* Edisi Ketujuh, Indeks Kelompok Gramedia, 2000
  - Edhy Sutanta, *Sistem Basis Data*, Graha Ilmu, 2004
  - Fathansyah,Ir., "*Basis Data"*, Informatika Bandung, Bandung, 2001

Penilaian

- Tugas : 10%
- Kuis : 20%
- Praktek : 30%
- UTS (Ujian Tengah Semester) : 20%
- UAS (Ujian Akhir Semester) : 20%

Basis Data

Konsep Basis Data

Putu Manik Prihatini
gek_anik@yahoo.com

Data

- Data adalah bahan keterangan tentang kejadian-kejadian nyata yang dirumuskan dalam kelompok lambang tertentu yang tidak acak yang menunjukkan jumlah, tindakan atau hal
- Data dapat berupa catatan-catatan dalam kertas, buku atau tersimpan sebagai file dalam basis data
- Contoh data adalah catatan identitas pegawai, catatan transaksi pembelian, catatan transaksi penjualan, dan lain-lain

Informasi

- Informasi merupakan hasil pengolahan data sehingga menjadi bentuk penting bagi penerimanya dan mempunyai kegunaan sebagai dasar dalam pengambilan keputusan yang dapat dirasakan manfaatnya pada saat itu juga atau pada saat mendatang
- Untuk memperoleh informasi, membutuhkan data dan unit pengolah
- Contoh informasi adalah daftar pegawai berdasarkan unit kerja, rekapitulasi transaksi pembelian selama 1 bulan, rekapitulasi transaksi penjualan selama 3 bulan, dan lain-lain

Definisi Basis Data

- Basis diartikan sebagai gudang
- Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang, hewan, peristiwa, konsep, keadaan dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya

- Basis data adalah :
  - Himpunan kelompok data yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah
  - Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan
  - Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis

Sejarah Basis Data

- Tahap I (awal tahun 1960-an)
  Ciri-cirinya :
  - Data diolah berdasarkan prinsip berkas pada lingkungan komputer mainframe

- Tahap II (akhir tahun 1960-an)
  Ciri-cirinya :
  - Muncul konsep sistem basis data
  - Muncul konsep sistem manajemen basis data
  - Layanan informasi secara *online*
  - Layanan informasi berbasis teks

Sejarah Basis Data

- Tahap III (awal tahun 1970-an)
  Ciri-cirinya :
  - Muncul aplikasi basis data berbasis sistem pakar dalam sistem pendukung keputusan
  - Pemrograman berbasis objek

- Tahap IV (tahun 1980-an)
  Ciri-cirinya :
  - Muncul sistem berbasis *hypertext* yang memungkinkan penampilan informasi berdasarkan suatu kata kunci yang dapat dilakukan secara acak

Sejarah Basis Data

- Tahap V (tahun 1990-an)
  Ciri-cirinya :
  - Muncul aplikasi basis data untuk kecerdasan buatan, *fuzzy logic* dan multimedia
  - Muncul aplikasi basis data berorientasi objek
  - Muncul aplikasi basis data secara *online* untuk internet

Operasi Dasar Basis Data

- Pembuatan basis data baru (*create database*)
- Penghapusan basis data (*drop database*)
- Pembuatan file/tabel baru ke suatu basis data (*create table*)
- Penghapusan file/tabel dari suatu basis data (*drop table*)
- Penambahan/pengisian data baru ke sebuah file/tabel di sebuah basis data (*insert*)
- Pengambilan data dari sebuah file/tabel (*search/retrieve/select*)
- Pengubahan data dari sebuah file/tabel (*update*)
- Penghapusan data dari sebuah file/tabel (*delete*)

Tujuan Basis Data

- Kecepatan dan kemudahan (*speed*)
- Efisiensi ruang penyimpanan (*space*)
- Keakuratan (*accuracy*)
- Ketersediaan (*availability*)
- Kelengkapan (*completeness*)
- Keamanan (*security*)
- Kebersamaan pemakaian (*share ability*)

Definisi Sistem Basis Data

- Sistem basis data adalah sistem yang terdiri atas kumpulan file/tabel yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai dan atau program lain untuk mengakses dan memanipulasi file-file/tabel-tabel tersebut

- Komponen-komponen utama sistem basis data adalah :
  - Perangkat keras (*hardware*)
  - Sistem operasi (*operating system*)
  - Basis data (*database*)
  - Sistem (aplikasi/perangkat lunak) pengelola basis data (DBMS)
  - Pemakai (*user*)
  - Aplikasi (perangkat lunak) lain (bersifat opsional)

Sistem Informasi & Basis Data

- Sistem Informasi Manajemen (SIM) merupakan kumpulan subsistem yang saling berinteraksi untuk melakukan fungsi pengolahan data yaitu menerima input berupa data-data, mengolah dan menghasilkan output berupa informasi

- Komponen penyusun SIM adalah :
  - Perangkat keras (*hardware*)
  - Perangkat lunak (*software*)
  - Berkas basis data
  - Prosedur
  - Manusia

- Peranan basis data dalam SIM adalah :
  - Komponen penyusun SIM
  - Infrastruktur SIM
  - Sumber informasi bagi SIM
  - Sarana mencapai efisiensi SIM
  - Sarana mencapai efektifitas SIM

DBMS (*Database Management System)*

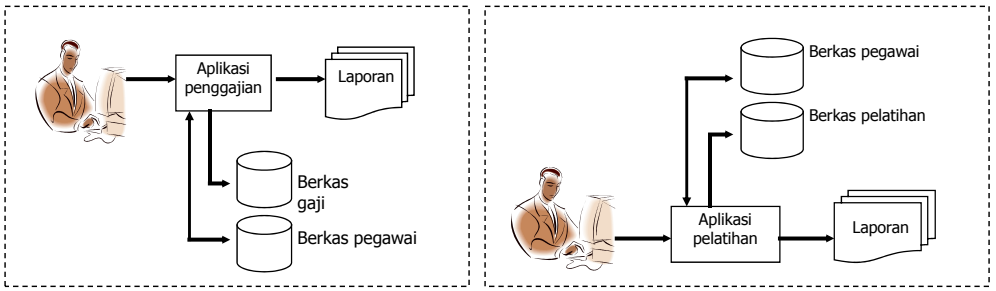
- DBMS adalah perangkat lunak yang mengelola basis data dengan kriteria sebagai berikut :
  - Menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali
  - Mengatur mekanisme pengamanan data, pemakaian data secara bersama-sama, pemaksaan keakuratan/konsistensi data
- Contoh perangkat lunak yang termasuk DBMS adalah dBase, MySQL, Ms. Access, SQL Server, Oracle dan sebagainya

Keuntungan DBMS

- Mengatasi redundansi dan inkonsistensi data yaitu data yang berulang pada beberapa berkas. Jika terjadi perubahan pada suatu berkas, maka berkas-berkas yang lain yang saling terkait harus diubah juga
- Kesulitan pengaksesan data diatasi dengan menggunakan DBMS untuk mengambil data secara langsung dengan bahasa yang mudah digunakan
- Isolasi data untuk standarisasi yaitu semua file dalam satu database dibuat dalam format yang sama sehingga memudahkan membuat program aplikasinya
- Multiple user yaitu data digunakan oleh banyak orang pada waktu yang bersamaan dengan menggunakan program yang sama
- Keamanan dengan cara memberikan hak akses yang berbeda untuk setiap pemakai
- Integritas yaitu file-file yang saling terkait dapat dihubungkan dengan menggunakan field kunci dari setiap file
- Data independece yaitu perubahan yang terjadi pada database tidak mengubah program aplikasinya

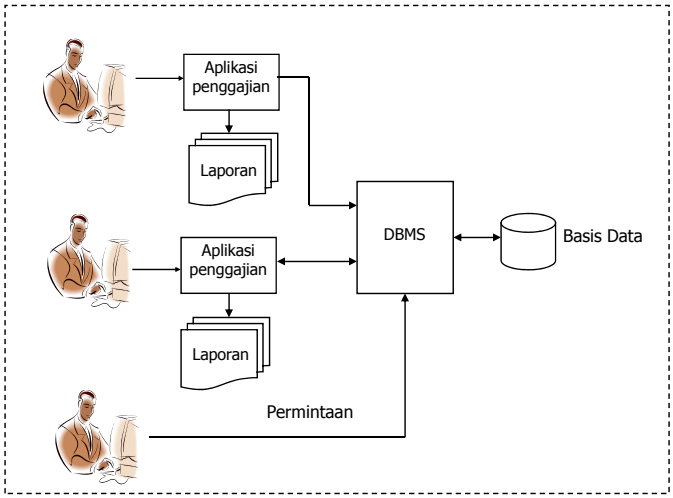
Perbedaan Sistem Berkas & DBMS

- Sistem Berkas



Perbedaan Sistem Berkas & DBMS

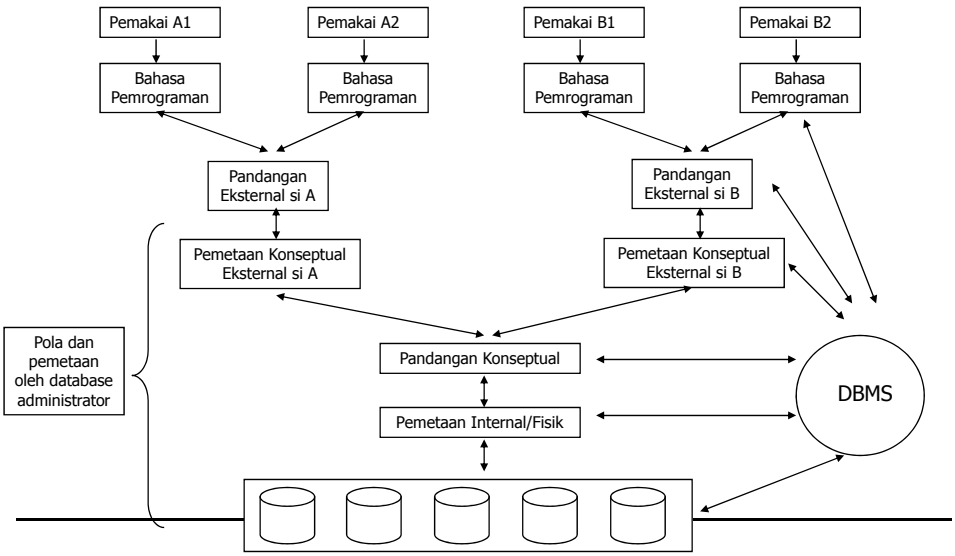
- DBMS

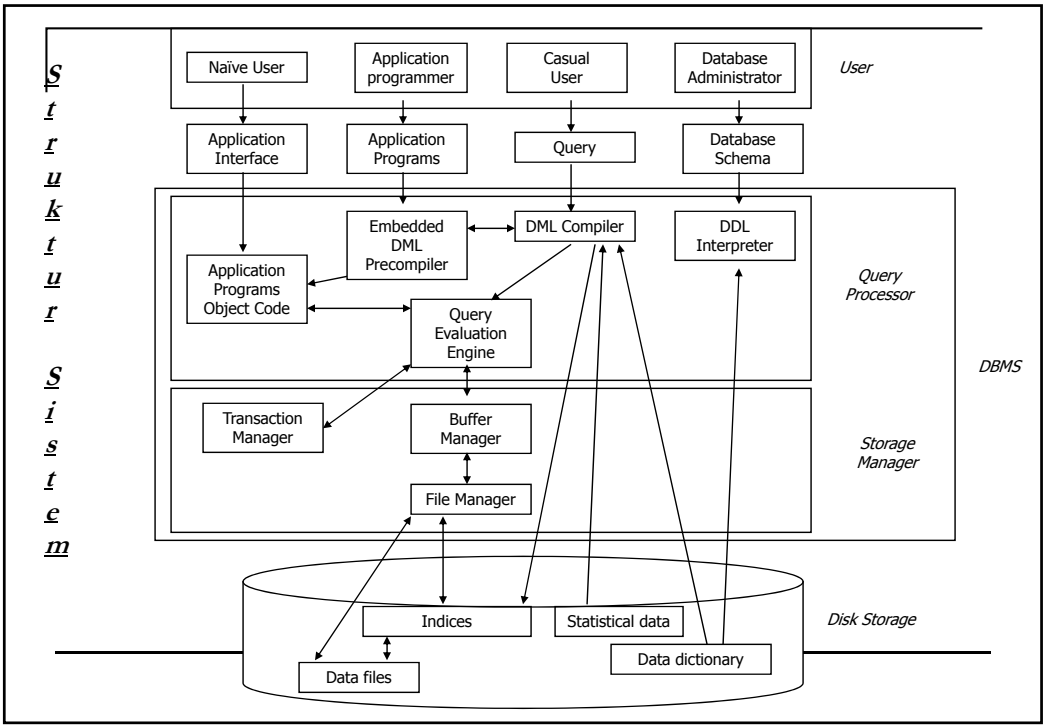


Abstraksi Data

- Abstraksi data merupakan tingkatan/level dalam bagaimana melihat data dalam sebuah sistem basis data
- Ada 3 level abstraksi data :
  - Level Fisik (*Physical Level*)
    - Pemakai melihat data sebagai gabungan dari struktur dan datanya sendiri
    - Pemakai berkompeten dalam mengetahui bagaimana representasi fisik dari penyimpanan/pengorganisasian data
  - Level Konseptual/Logik (*Conceptual Level*)
    - Level yang menggambarkan data apa yang sebenarnya (secara fungsional) disimpan dalam basis data dan hubungannya dengan data yang lain
    - Pemakai mengetahui bahwa data misalnya data pengawai disimpan dalam beberap file/tabel yaitu tabel pribadi, tabel pendidikan, tabel keluarga
  - Level Penampakan (*View Level*)
    - Pemakai hanya membutuhkan sebagian data dalam basis data yang diatur oleh aplikasi *end user*

Arsitektur Sistem Basis Data





Bahasa Basis Data

- Bahasa basis data adalah bahasa yang mengatur komunikasi antara pemakai dengan basis data yang ditetapkan oleh perusahaan pembuat DBMS
- Bahasa ini terdiri atas sejumlah perintah yang diformulasikan dan dapat diberikan oleh *user* dan dikenali/diproses oleh DBMS untuk melakukan suatu aksi/pekerjaan tertentu
- Ada dua jenis yaitu :
  - *Data Definition Language* (DDL)
    - Bahasa ini digunakan untuk membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur penyimpanan tabel, dan sebagainya
  - *Data Manipulation Language* (DML)
    - Bahasa ini digunakan untuk melakukan manipulasi dan pengambilan data pada suatu basis data
    - Manipulasi data dapat berupa :
      - Penyisipan/penambahan data baru ke suatu basis data
      - Penghapusan data dari suatu basis data
      - Pengubahan data di suatu basis data

Pemakai (*User*)

- Programmer Aplikasi
  Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML) yang disertakan dalam program yang ditulis dalam bahasa pemrograman induk (seperti Delphi, Visual Basic)
- User Mahir (*Casual User*)
  Pemakai yang berinteraksi dengan sistem tanpa menulis program, melainkan menyatakan query 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 yang telah ditulis/disediakan sebelumnya
- User Khusus (*Specialized User*)
  Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus, seperti untuk aplikasi Sistem Pakar, yang mengakses basis data dengan/tanpa DBMS yang bersangkutan

Basis Data

Entity Relationship Model

Putu Manik Prihatini
gek_anik@yahoo.com

Model Basis Data

- Model basis data menyatakan hubungan antar tuple yang tersimpan dalam basis data
- Secara umum ada 2 kelompok cara merepresentasikan model data dalam perancangan basis data yaitu :
  - Model Logik Berdasarkan Obyek (*Object-Based Logical Models)*
    - Model Hubungan Entitas *(Entity-Relationship Model)*
    - Model Berorientasi Objek (*Object-Oriented Model)*
    - Model Data Semantik (*Semantic Data Model)*
    - Model Data Fungsional (*Functional Data Model)*
  - *Model Logik Berdasarkan Record (Record-Based Logical Models)*
    - Model Relasional *(Relational Model)*
    - Model Hirarkis (*Hierarchical Model)*
    - Model Jaringan (*Network Model)*

Model Hubungan Entitas (ER-Model)

- ER-Model merupakan suatu model data yang dikembangkan berdasarkan obyek, yang digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pemakai secara logik
- ER-Model didasarkan pada suatu persepsi bahwa dunia nyata (*real world*) terdiri atas obyek-obyek dasar yang mempunyai hubungan satu sama lainnya
- ER-Model digambarkan dalam bentuk diagram yang disebut diagram ER (ER-Diagram / ERD) dengan menggunakan simbol-simbol grafis tertentu

Entitas

- Entitas menunjukkan obyek-obyek dasar yang terkait didalam sebuah sistem
- Berupa orang, benda atau hal yang keterangannya perlu disimpan didalam basis data
- Aturan penggambaran entitas dalam ERD adalah :
  - Dinyatakan dengan simbol persegi panjang
  - Nama entitas dituliskan didalam simbol persegi panjang
  - Nama entitas berupa kata benda tunggal
  - Nama entitas menggunakan nama yang mudah dipahami dan menyatakan maknanya dengan jelas

- Nama entitas dapat tersusun atas lebih dari satu kata yang dipisahkan dengan tanda underscore (_), misalnya
  - Untuk menyatakan nama entitas mata kuliah, menggunakan **Mata_Kuliah**, bukan **Mata** atau **Kuliah**
- Nama entitas sebaiknya jangan terlalu panjang, misalnya
  - Untuk menyatakan nama entitas orang tua atau wali mahasiswa, menggunakan **Wali_Mahasiswa**, bukan **Orang_Tua_Wali_Mahasiswa**
- Nama entitas yang tersusun atas lebih dari satu kata dapat disingkat, misalnya
  - Untuk menyatakan nama entitas program studi, dapat disingkat menjadi entitas **Prodi**

- Penentuan entitas dalam suatu sistem harus cermat dan hati-hati
- Sebagai contoh :
  - Keterangan mengenai kabupaten asal mahasiswa merupakan entitas karena nilai-nilai yang terkandung dalam data kabupaten bersifat tidak pasti (berubah). Demikian juga dengan entitas propinsi, agama dan pekerjaan
  - Keterangan mengenai jenis kelamin mahasiswa tidak merupakan entitas karena nilai-nilai yang terkandung dalam data jenis kelamin bersifat pasti dan tidak akan pernah berubah. Demikian juga dengan golongan darah

Atribut

- Merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan sebagai basis data
- Aturan penggambaran atribut pada ERD adalah :
  - Dinyatakan dengan simbol elips
  - Nama atribut dituliskan didalam simbol elips
  - Nama atribut berupa kata benda tunggal
  - Nama atribut menggunakan nama yang mudah dipahami dan menyatakan maknanya dengan jelas
  - Atribut dihubungkan dengan entitas yang bersesuaian dengan menggunakan sebuah garis (sebaiknya garis lurus, tetapi jika tidak memungkinkan boleh tidak menggunakan garis lurus)

- Ada beberapa jenis atribut, yaitu :
  - Atribut Sederhana
    Atribut sederhana adalah atribut atomik yang tidak dapat dipilah lagi. Pada entitas Mahasiswa, yang merupakan atribut sederhana adalah NIM
  - Atribut Komposit
    Atribut komposit adalah atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing memiliki makna. Pada entitas Mahasiswa, yang merupakan atribut komposit adalah Alamat_Mahasiswa karena masih bisa diuraikan lagi menjadi sub atribut yaitu nama_jalan dan nama_kota

  - Atribut Bernilai Tunggal
    Atribut bernilai tunggal ditujukan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data. Pada entitas Mahasiswa, semua atribut dari entitas ini merupakan atribut bernilai tunggal
  - Atribut Bernilai Banyak
    Atribut bernilai banyak ditujukan pada atribut-atribut yang dapat diisi dengan lebih dari satu nilai. Pada entitas Mahasiswa, bisa ditambahkan atribut hobi, dimana ada mahasiswa yang mempunyai satu hobi saja, ada juga yang mempunyai banyak hobi bahkan ada yang tidak mempunyai hobi

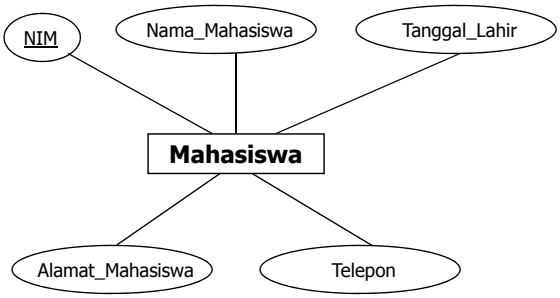
- Atribut Kosong

  Atribut tidak harus bernilai dapat bernilai kosong yang dinyatakan dengan nilai konstanta Null. Pada entitas Mahasiswa, atribut telepon merupakan atribut kosong

- Atribut Turunan

  Merupakan atribut yang nilai-nilainya dapat diturunkan dari atribut atau entitas lain yang berhubungan. Atribut ini sebenarnya dapat ditiadakan dari sebuah tabel karena nilai-nilainya tergantung pada nilai yang ada di atribut lainnya. Contoh, atribut angkatan tidak perlu ditambahkan karena dapat diperoleh dari atribut NIM, sedangkan atribut ip tidak perlu ditambahkan karena dapat diperoleh dari pengolahan atribut indeks_nilai

- Contoh atribut dari entitas Mahasiswa dan penggambarannya :



Relasi Antar Entitas

- Menyatakan hubungan yang terjadi antar dua buah entitas yang keterangannya perlu disimpan dalam basis data
- Aturan penggambaran relasi ini dalam ERD adalah :
  - Dinyatakan dengan simbol belah ketupat
  - Nama relasi dituliskan didalam simbol belah ketupat
  - Relasi menghubungkan dua entitas
  - Nama relasi berupa kata kerja aktif (diawali dengan awalan me-) tunggal
  - Nama relasi menggunakan nama yang mudah dipahami dan menyatakan maknanya dengan jelas

Rasio Kardinalitas

- Kardinalitas Satu ke Satu / 1 – 1 (One to One)
  - Setiap entitas pada himpunan entitas A berhubungan dengan satu entitas pada himpunan entitas B, demikian juga sebaliknya
  - Sebagai contoh, **satu** orang Mahasiswa hanya dimungkinkan **mempunyai satu** orang Wali_Mahasiswa
- Kardinalitas Satu ke Banyak / 1 – n (One to Many)
  - Setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan satu entitas pada himpunan entitas A
  - Sebagai contoh, **satu** program studi dapat **dipilih** oleh **lebih dari satu** mahasiswa

- Kardinalitas Banyak ke Satu / n – 1 (Many to One)
  - Setiap entitas pada himpunan entitas A berhubungan dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan banyak entitas pada himpunan entitas A
  - Sebagai contoh, **lebih dari satu** mahasiswa dapat **memilih** hanya **satu** buah program studi

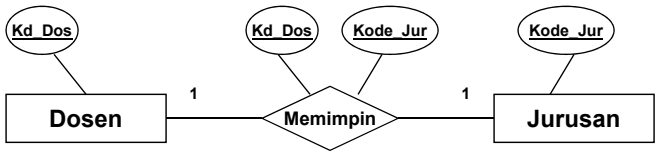
- Kardinalitas Banyak ke Banyak / n – n (Many to Many)
  - Setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, demikian juga sebaliknya
  - Sebagai contoh, **lebih dari satu** mahasiswa dapat **mengikuti lebih dari satu** mata kuliah

Catatan :

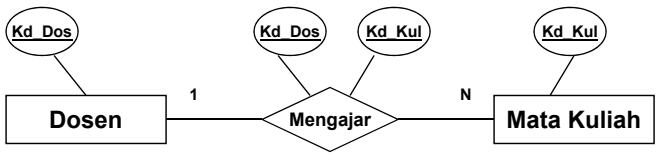
- Dalam ER Model ada kejadian yang benar-benar terjadi tetapi keterangannya tidak perlu disimpan dalam basis data
- Kejadian ini tidak termasuk sebagai relasi antar entitas
- Sebagai contoh :
  - Mahasiswa **memilih** Mata Kuliah yang akan diikutinya, kemudian mengisikannya ke dalam blanko KRS. Kejadian ini tidak termasuk relasi antar entitas, tetapi hasil akhir mata kuliah yang benar-benar akan diikutinya dan dicantumkan dalam KRS yang termasuk dalam relasi antar entitas
  - Mahasiswa **menanyakan** daftar mata kuliah yang ditawarkan pada Program Studi
  - Mahasiswa **menuliskan** daftar Mata Kuliah dalam blanko KRS
  - Dosen **menanyakan** Mata Kuliah yang diikuti oleh mahasiswa

Contoh Kasus Penggambaran ERD

1. Relasi satu ke satu
   Adanya relasi entitas Dosen dengan entitas Jurusan yang diberi nama Memimpin. Setiap dosen hanya bisa memimpin satu jurusan, sebaliknya satu jurusan hanya dipimpin oleh satu orang dosen

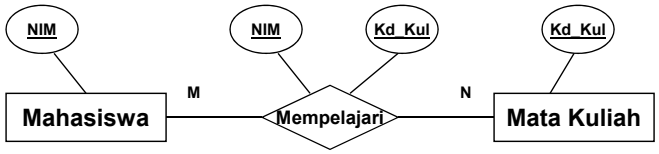


2. Relasi satu ke banyak
   Adanya relasi antara entitas Dosen dengan entitas Kuliah yang diberi nama Mengajar. Setiap dosen dapat mengajar lebih dari satu mata kuliah, sedangkan setiap mata kuliah hanya dapat diajarkan oleh satu orang dosen.



3. Relasi banyak ke banyak

   Adanya relasi antara entitas Mahasiswa dengan entitas Kuliah yang diberi nama Belajar. Setiap mahasiswa dapat mempelajari lebih dari satu mata kuliah, sebaliknya setiap mata kuliah dapat dipelajari oleh lebih dari satu orang mahasiswa.



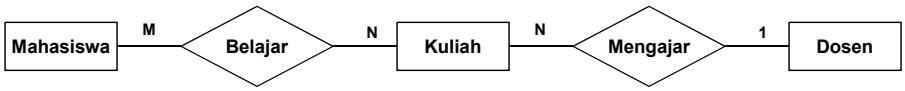
Tahapan Pembuatan ERD

- Ada dua kelompok tahap dalam pembuatan ERD yaitu :
  - Tahap pembuatan ERD awal (*preliminary design)*
  - Tahap optimasi ERD (*final design)*

- Langkah-langkah teknis dalam pembuatan ERD adalah:
  1. Mengidentifikasi dan menetapkan seluruh entitas yang terlibat
  2. Menentukan atribut-atribut key dari masing-masing entitas
  3. Mengidentifikasi dan menetapkan seluruh relasi di antara entitas-entitas yang ada beserta foreign key (jika terjadi kardinalitas one to many atau many to many)
  4. Menentukan derajat/kardinalitas relasi untuk setiap relasi
  5. Melengkapi entitas dan relasi dengan atribut-atribut deskriptif

     (Karena cenderung dianggap mengganggu, maka untuk sistem yang besar dan kompleks langkah ini seringkali tidak dilakukan, sehingga penggambaran ERD hanya sampai langkah keempat)

- Penggambaran ERD boleh dilengkapi dengan kamus data
- Contoh :



- Kamus Data :
  - Mahasiswa = (NIM, Nama_Mhs, Alamat, Tmpt_Lahir, Tgl_Lahir)
  - Kuliah = (Kode_MK, Nama_MK, SKS, Semester)
  - Dosen = (Kd_Dos, Nama_Dos, Keahlian, Alamat)
  - Belajar = (NIM, Kode_MK, Index_Prestasi)
  - Mengajar = (Kd_MK, Kd_Dos, Waktu, Ruang)

Derajat Relasi Minimum

- Derajat relasi minimum menunjukkan hubungan minimum yang boleh terjadi dalam sebuah relasi antar himpunan entitas
- Derajat relasi minimum ini bisa bernilai nol atau satu
- Contoh : pada relasi MAHASISWA MENGAMBIL KULIAH

  Seorang mahasiswa boleh mengambil banyak mata kuliah sekaligus demikian juga sebaliknya sebuah mata kuliah dapat diambil oleh banyak mahasiswa

  Derajat relasi minimum disini dapat diperoleh dari adanya fakta bahwa seorang mahasiswa boleh tidak mengambil mata kuliah apapun misalnya karena cuti dan sebuah mata kuliah bisa tidak diambil oleh siapapun misalnya karena mata kuliah pilihan

  Jadi, derajat minimum disini adalah nol

- Contoh : pada relasi DOSEN MENGAJAR KULIAH

  Seorang dosen bisa mengajar banyak mata kuliah sekaligus sedangkan sebuah mata kuliah hanya dapat diajarkan oleh satu orang dosen

  Derajat relasi minimum disini dapat diperoleh dari adanya fakta bahwa seorang dosen mungkin saja belum / tidak dimungkinkan untuk mengajar satu mata kuliahpun sehingga derajat relasi minimum-nya adalah nol

  Akan tetapi ada fakta, bahwa setiap mata kuliah harus sudah ditentukan dosen yang akan mengajarkannya, sehingga membuat derajat relasi minimumnya menjadi satu
- Cara penggambarannya adalah :

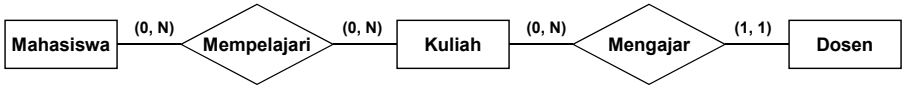
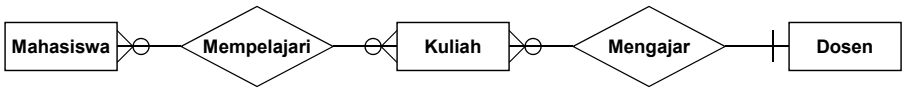


Diagram ER dalam Notasi Lain

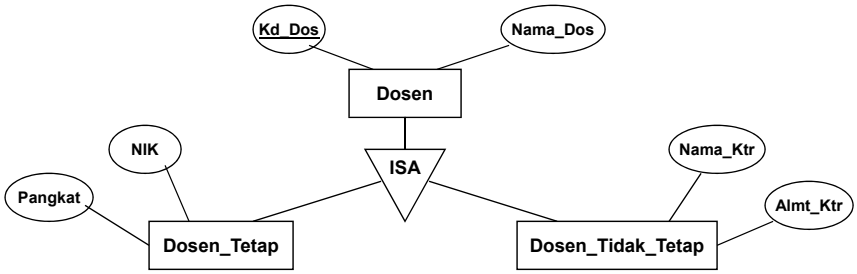
- Contoh penggambaran diagram ER dalam notasi lain adalah :



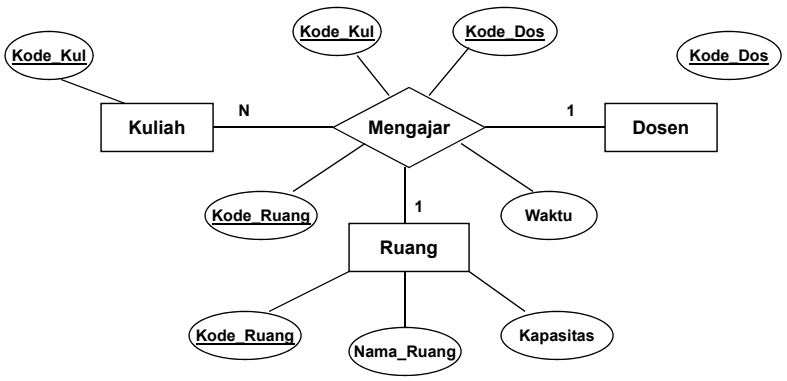
Notasi	Derajat Relasi Minimum -Maksimum
atau	(0, N)
atau	(1, N)
atau	(1, 1)
atau	(0, 1)

Model Data Lanjutan

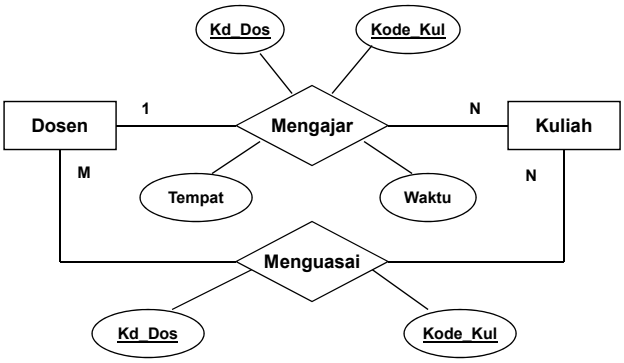
- Entitas Lemah (*Weak Entity*)
  - Entitas-entitas yang kemunculannya tergantung pada keberadaannya dalam sebuah relasi terhadap entitas lainnya
- Sub Entitas
  - Entitas yang beranggotakan entitas-entitas yang merupakan bagian dari himpunan entitas yang lebih superior
  - Contoh :



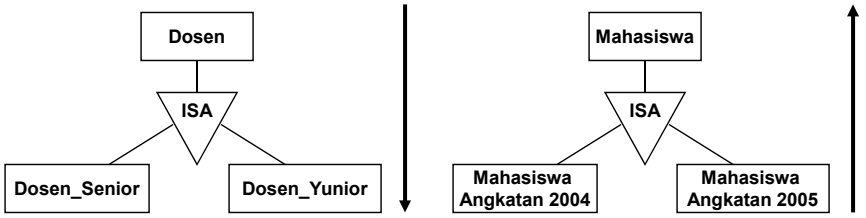
- Relasi Multi Entitas
  - Relasi dari tiga himpunan entitas atau lebih
  - Contoh :



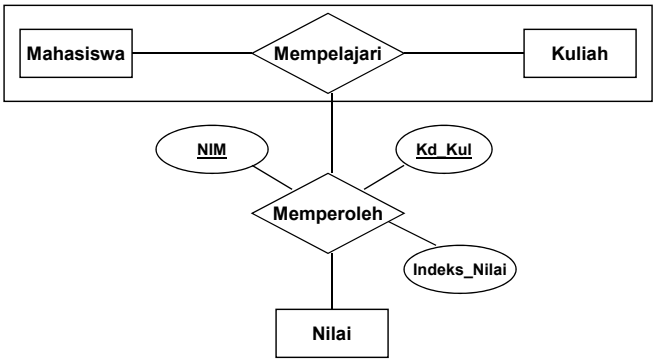
- Relasi Ganda
  - Relasi yang muncul antara dua entitas tidak hanya satu relasi, melainkan lebih dari satu relasi
  - Contoh :



- Spesialisasi dan Generalisasi
  - Spesialisasi adalah dekomposisi sebuah entitas menjadi beberapa entitas baru
  - Spesialisasi menekankan pada perbedaan antar kelompok entitas
  - Generalisasi adalah pengelompokan beberapa entitas menjadi sebuah entitas baru
  - Generalisasi menekankan pada persamaan antar kelompok entitas
  - Contoh :



- Agregrasi
  - Menggambarkan sebuah relasi yang secara langsung menghubungkan sebuah entitas dengan sebuah relasi dalam diagram ER
  - Contoh :



Latihan :

- Diberikan dokumen dasar seperti berikut. Gambarkan ERD-nya !

PT. SANTA PURI
Jalan Senopati No. 11
Yogyakarta

FAKTUR PEMBELIAN BARANG

Kode Supplier : G01
Nama Supplier : Gobel Nustra

Tanggal : 19/06/2008
Nomor : 998

Kode	Nama Barang	Qty	Harga	Jumlah
A01	AC Split 1/2 PK	10	1.350.000	13.500.000
A02	AC Split 1 PK	10	2.000.000	20.000.000
			Total Faktur	33.500.000

Jatuh Tempo Faktur : 26/06/2008

Latihan :

- Diberikan kasus seperti berikut :

  Sebuah perpustakaan di suatu universitas menyediakan buku-buku untuk dipinjam. Bagi mahasiswa yang mau meminjam buku harus menjadi anggota terlebih dahulu. Keterlambatan pengembalian buku akan dikenakan denda.

  Gambarkan ERD dari kasus diatas !

- Diberikan kasus seperti berikut :

  Setiap semester, mahasiswa di suatu universitas melakukan daftar ulang, dimana mereka akan menentukan mata kuliah apa saja yang akan diambil pada semester tersebut. Kemudian, mata kuliah yang diselenggarakan pada semester tersebut akan ditentukan dosen siapa yang akan mengajarkannya. Satu orang dosen bisa mengajarkan lebih dari satu mata kuliah, demikian juga sebaliknya. Setelah perkuliahan berakhir, mahasiswa tersebut akan memperoleh nilai untuk mata kuliah yang diambil.

  Gambarkan ERD dari kasus diatas !

Basis Data

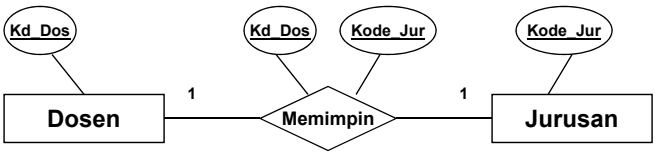
Implementasi Basis Data

Putu Manik Prihatini
gek_anik@yahoo.com

Transformasi Umum/Dasar

- Aturan umum dalam pemetaan model data dalam bentuk ERD menjadi basis data fisik adalah
  - Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel (file data)
  - Relasi dengan derajat relasi 1-1 yang menghubungkan 2 buah himpunan entitas akan direpresentasikan dalam bentuk penambahan/penyertaan atribut-atribut relasi ke tabel yang mewakili salah satu dari kedua himpunan entitas. Untuk menentukan pilihan yang tepat perlu dilihat derajat relasi minimum-nya, dimana penambahan atribut relasi dilakukan terhadap tabel yang memiliki derajat relasi minimum lebih besar. Akan tetapi, jika derajat relasi minimum-nya sama, maka penambahan atribut relasi dilakukan terhadap tabel yang jumlah barisnya lebih sedikit atau yang ukuran tabelnya diperkirakan lebih kecil

    Contoh pada relasi DOSEN MEMIMPIN JURUSAN



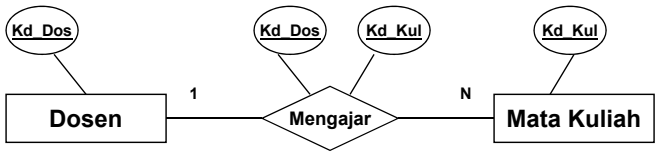
- Derajat minimum dosen adalah nol, sedangkan derajat minimum jurusan adalah satu

Tabel Dosen		
Kd_Dos	Nama_Dos	Alamat_Dos

Tabel Jurusan		
Kode_Jur	Nama_Jur	Kd_Dos

- Relasi dengan derajat relasi 1 – N yang menghubungkan 2 buah himpunan entitas direpresentasikan dengan pemberian primary key dari himpunan entitas pertama (yang berderajat 1) ke tabel yang mewakili himpunan entitas kedua (yang berderajat N)

  Contoh pada relasi DOSEN MENGAJAR KULIAH

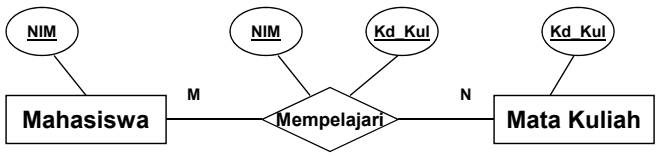


Tabel Dosen		
Kd_Dos	Nama_Dos	Alamat_Dos

Tabel Kuliah				
Kd_Kul	Nama_Kul	SKS	Semester	Kd_Dos

- Relasi dengan derajat relasi N – N yang menghubungkan 2 buah himpunan entitas direpresentasikan dengan membentuk tabel (file data) khusus yang memiliki field (tepatnya *foreign key*) yang berasal dari key-key dari himpunan entitas yang dihubungkannya

  Contoh pada relasi MAHASISWA MENGAMBIL KULIAH



Tabel Mahasiswa		
NIM	Nama_Mhs	Alamat_Mhs

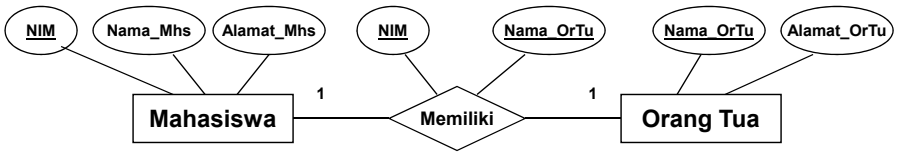
Tabel Kuliah			
Kd_Kul	Nama_Kul	SKS	Semester

Tabel Mempelajari	
NIM	Kd_Kul

Implementasi Himpunan Entitas Lemah dan Sub Entitas

- Himpunan entitas lemah dan sub entitas dalam diagram ER diimplementasikan dalam bentuk tabel dengan menyertakan atribut key yang ada di himpunan entitas kuat/reguler yang berelasi dengannya, dimana atribut key tersebut akan menjadi key bagi tabel hasil implementasi himpunan entitas lemah dan sub entitas
- Contoh :

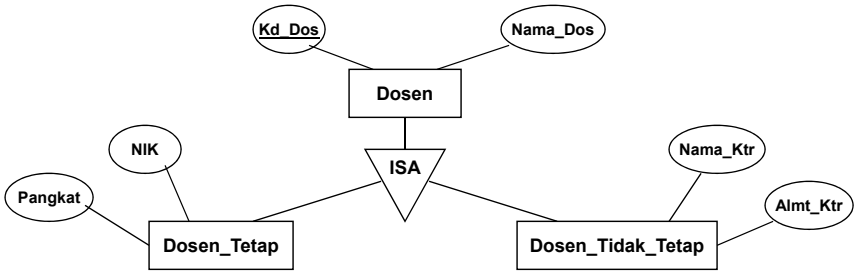
- Contoh himpunan entitas lemah



Tabel Mahasiswa		
NIM	Nama_Mhs	Alamat_Mhs

Tabel Orang_Tua		
NIM	Nama_Ortu	Alamat_Ortu

- Contoh sub entitas :



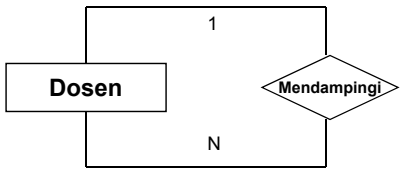
Tabel Dosen	
Kd_Dos	Nama_Dos

Tabel Dosen_Tetap		
Kd_Dos	NIK	Pangkat

Tabel Dosen_Tidak_Tetap		
Kd_Dos	Nama_Ktr	Almt_Ktr

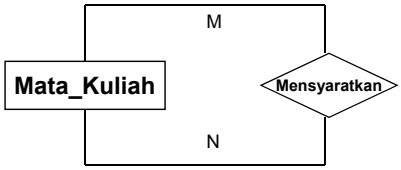
Implementasi Relasi Tunggal

- Untuk relasi tunggal dengan derajat relasi 1 – N dapat diimplementasikan melalui penggunaan field key dua kali tapi untuk fungsi yang berbeda
- Contoh :



Tabel Dosen			
Kd_Dos	Nama_Dos	Alamat_Dos	Kd_Dos_Pend

- Untuk relasi tunggal dengan derajat relasi N – N dapat diimplementasikan melalui pembentukan tabel baru dengan mendapatkan field dari semua atribut relasi (jika ada) yang ditambah dengan atribut key dari himpunan entitasnya
- Contoh :

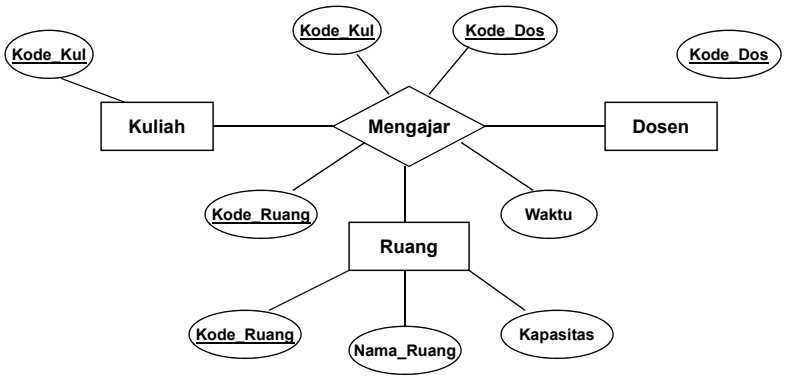


Tabel Kuliah			
Kd_Kul	Nama_Kul	SKS	Semester

Tabel Prasyarat	
Kd_Kul	Kd_Kul_Syarat

Implementasi Relasi Multi Entitas

- Secara umum, relasi multi entitas yang menghubungkan lebih dari dua himpunan entitas akan diimplementasikan sebagai sebuah tabel khusus
- Jika derajat relasi parsial di antara (N-1) buah himpunan entitas X adalah 1 – N, maka relasi tadi tidak perlu diwujudkan dalam bentuk tabel khusus dan atribut-atributnya cukup dilekatkan pada himpunan entitas X tersebut
- Jika derajat relasi parsial di antara (N-1) buah himpunan entitas X adalah N – N, maka relasi tadi perlu diwujudkan dalam bentuk tabel khusus



- Pada relasi pengajaran diatas, setiap mata kuliah dapat diajarkan oleh seorang dosen, sedangkan satu orang dosen dapat mengajarkan lebih dari satu mata kuliah, sehingga derajat relasi parsial Dosen – Kuliah = 1 – N

- Pada relasi pengajaran diatas, setiap mata kuliah hanya dapat diselenggarakan di sebuah ruang yang telah ditentukan, sedangkan setiap ruang pada saat yang berbeda dapat digunakan untuk pengajaran berbagai mata kuliah, sehingga derajat relasi parsial Ruang – Kuliah = 1 – N
- Pada relasi pengajaran diatas, setiap ruangan dapat digunakan oleh banyak dosen, dan setiap dosen dapat menggunakan berbagai ruangan karena dapat mengajarkan lebih dari satu mata kuliah, sehingga derajat relasi parsial Ruang – Dosen = N – N
- Jadi, derajat relasi parsial antara Dosen atau Ruang terhadap Kuliah = 1 – N sehingga tidak perlu membuat tabel khusus

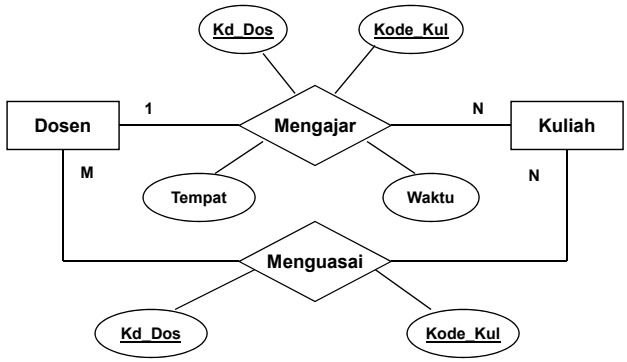
Tabel Kuliah						
Kd_Kul	Nama_Kul	SKS	Semester	Kd_Dos	Kd_Ruang	Waktu

- Jika suatu saat, suatu mata kuliah (dengan jumlah sks yang besar) dapat dilaksanakan lebih dari satu kali dalam seminggu, dan mungkin untuk diselenggarakan di ruang yang berbeda, maka derajat relasi parsial Ruang – Kuliah menjadi N – N, sehingga perlu diwujudkan dalam bentuk tabel khusus

Tabel Pengajaran			
Kd_Kul	Kd_Dos	Kd_Ruang	Waktu

Implementasi Relasi Ganda

- Implementasinya berdasarkan derajat relasi masing-masing relasi
- Contoh :



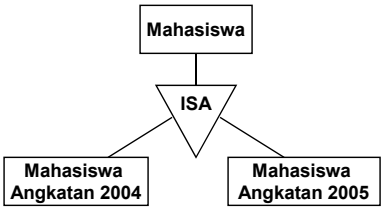
Tabel Dosen	
Kd_Dos	Nama_Dos

Tabel Kuliah						
Kd_Kul	Nama_Kul	SKS	Semester	Tempat	Waktu	Kd_Dos

Tabel Menguasai	
Kd_Dos	Kd_Kul

Implementasi Spesialisasi & Generalisasi

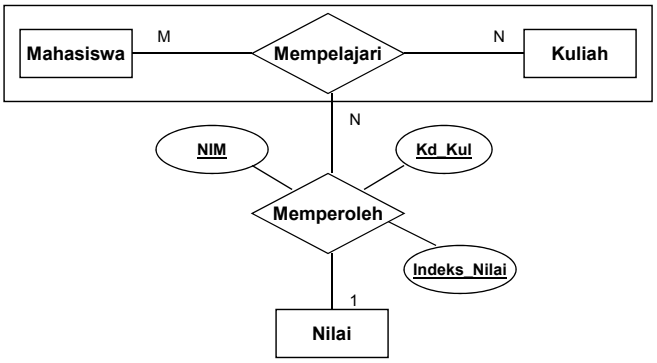
- Spesialisasi terhadap sebuah himpunan entitas akan menghasilkan sejumlah himpunan entitas baru yaitu satu himpunan entitas kuat yang akan menjadi acuan bagi himpunan entitas lainnya dan sisanya merupakan sub entitas
- Contohnya dapat dilihat pada implementasi sub entitas
- Generalisasi dilakukan dengan “mengabaikan” perbedaan beberapa himpunan entitas
- Pada tahap generalisasi akan dilakukan penyusutan jumlah himpunan entitas menjadi hanya sebuah tabel saja
- Untuk tetap mengakomodasi perbedaan, di tabel tersebut akan ditambahkan sebuah atribut yang nantinya akan diisi dengan kode khusus yang menyatakan perbedaan tersebut



Tabel Mahasiswa				
NIM	Nama_Mhs	Alamat_Mhs	Tgl_Lahir	Angkatan

Implementasi Agregasi

- Relasi pada agregasi dibentuk dari relasi lain, maka implementasinya harus dilakukan setelah relasi prasyarat tersebut diimplementasikan
- Contoh :



- Derajat relasi antara himpunan entitas Mahasiswa – Kuliah adalah M – N sehingga terbentuk sebuah tabel khusus

Tabel Mahasiswa		
NIM	Nama_Mhs	Alamat_Mhs

Tabel Kuliah			
Kd_Kul	Nama_Kul	SKS	Semester

Tabel Mengambil	
NIM	Kd_Kul

- Derajat relasi antara relasi Mengambil (sebagai hasil relasi antara himpunan entitas Mahasiswa – Kuliah) dengan himpunan entitas Nilai adalah N – 1 sehingga tidak perlu terbentuk tabel khusus, tetapi cukup menambahkan atribut key dari himpunan entitas Nilai ke relasi Mengambil

Tabel Nilai	
Indeks_Nilai	Grade

Tabel Mengambil		
NIM	Kd_Kul	Indeks_Nilai

Latihan :

- Diberikan dokumen dasar seperti berikut. Gambarkan ERD-nya !

PT. SANTA PURI
Jalan Senopati No. 11
Yogyakarta

FAKTUR PEMBELIAN BARANG

Kode Supplier : G01
Nama Supplier : Gobel Nustra

Tanggal : 19/06/2008
Nomor : 998

Kode	Nama Barang	Qty	Harga	Jumlah
A01	AC Split 1/2 PK	10	1.350.000	13.500.000
A02	AC Split 1 PK	10	2.000.000	20.000.000
			Total Faktur	33.500.000

Jatuh Tempo Faktur : 26/06/2008

Latihan :

- Diberikan kasus seperti berikut :

  Sebuah perpustakaan di suatu universitas menyediakan buku-buku untuk dipinjam. Bagi mahasiswa yang mau meminjam buku harus menjadi anggota terlebih dahulu. Keterlambatan pengembalian buku akan dikenakan denda.

  Gambarkan ERD dari kasus diatas ! Implementasikan ke tabel !

- Diberikan kasus seperti berikut :

  Setiap semester, mahasiswa di suatu universitas melakukan daftar ulang, dimana mereka akan menentukan mata kuliah apa saja yang akan diambil pada semester tersebut. Kemudian, mata kuliah yang diselenggarakan pada semester tersebut akan ditentukan dosen siapa yang akan mengajarkannya. Satu orang dosen bisa mengajarkan lebih dari satu mata kuliah, demikian juga sebaliknya. Setelah perkuliahan berakhir, mahasiswa tersebut akan memperoleh nilai untuk mata kuliah yang diambil.

  Gambarkan ERD dari kasus diatas ! Implementasikan ke tabel !

Basis Data

Model Relasional

Putu Manik Prihatini
gek_anik@yahoo.com

Definisi Model Relasional

- RDBM diperkenalkan oleh E.F. Codd pada tahun 1970
- Model ini menunjukkan mekanisme yang digunakan untuk mengelola data secara fisik dalam memori sekunder
- Model ini menjelaskan kepada pemakai tentang hubungan logik antar data dalam basis data dengan merepresentasikannya ke dalam bentuk relasi-relasi berupa tabel mendatar yang terdiri atas sejumlah baris yang menunjukkan *record* dan kolom yang menunjukkan atribut tertentu
- Dalam sebuah basis data, kerelasian antar relasi satu dengan yang lainnya ditunjukkan dengan menggunakan *Foreign Key / FK*

Istilah-Istilah RDBM

Istilah Formal	Istilah Non Formal	Keterangan
Elemen data		Sebuah nilai data aktual
Atribut	Kolom, field	Sekelompok rinci data yang mempunyai arti. Atribut memiliki tipe, ukuran dan domain yang sama
Record/tuple	Baris, rekaman	Sekumpulan atribut yang mempunyai hubungan terhadap obyek tertentu
Relasi	Tabel	Sekumpulan record yang sejenis
Derajat	Aritas	Jumlah atribut dalam sebuah relasi
Kardinalitas		Jumlah record dalam sebuah relasi
Kerelasian		Hubungan antar relasi
Unary Relation		Relasi yang tersusun oleh satu atribut

Istilah-Istilah RDBM

Istilah Formal	Istilah Non Formal	Keterangan
Binary relation		Relasi yang tersusun oleh dua atribut
Ternary relation		Relasi yang tersusu oleh tiga atribut
N-ary relation		Relasi yang tersusu oleh n atribut
Key		Satu atau gabungan atribut bersifat unik yang digunakan untuk mengidentifikasi setiap record dalam relasi
Candidate Key (CK)		Satu atau gabungan minimal atribut bersifat unik yang digunakan untuk mengidentifikasi setiap record dalam relasi
Primary Key (PK)		Bagian dari CK yang dipilih sebagai kunci utama dalam relasi

Istilah-Istilah RDBM

Istilah Formal	Istilah Non Formal	Keterangan
Alternate Key (AK)		Bagian dari CK yang tidak digunakan sebagai kunci utama dalam relasi
Foreign Key (FK)	Kunci tamu	Satu atau gabungan sembarang atribut yang menjadi PK dalam relasi lain yang mempunyai hubungan secara logik
Domain		Himpunan nilai yang memenuhi syarat
Schema		Deskripsi hubungan logik secara global, termasuk didalamnya nama dan dekripsi tipe dan ukuran atribut dan hubungan logik antar relasi basis data dalam lingkup sebuah sistem
Subschema		Deskripsi hubungan logik secara terpisah, termasuk didalamnya nama dan dekripsi tipe dan ukuran atribut dan hubungan logik antar relasi basis data dalam lingkup sebuah sub sistem aplikasi basis data

Karakteristik Relasi pada RDBM

- Semua elemen data pada suatu record dan atribut tertentu harus mempunyai nilai tunggal, bukan suatu perulangan dan harus berupa nilai yang tidak dapat dibagi
- Semua elemen data pada suatu atribut tertentu dalam sebuah relasi harus mempunyai tipe dan ukuran yang sama
- Masing-masing atribut dalam sebuah relasi mempunyai nama yang unik (sekalipun tidak disarankan, nama atribut pada suatu relasi diijjinkan sama dengan nama atribut pada relasi yang lain
- Pada sebuah relasi tidak ada dua record data yang identik

Key (Kunci)

- Key adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua nilai dari suatu entitas secara unik. Artinya, jika suatu atribut dijadikan sebagai key, maka tidak boleh ada dua atau lebih nilai yang sama untuk atribut tersebut
- Berdasarkan jumlah atribut penyusunnya, kunci dibagi menjadi 2 yaitu :
  - Kunci sederhana

    Kunci yang tersusun atas sebuah atribut
  - Kunci komposit

    Kunci yang tersusun atas gabungan atribut. Hal ini terjadi jika untuk mencapai sifat unik, tidak dapat dipenuhi oleh sebuah atribut, tetapi harus menggabungkan lebih dari satu atribut

- Berdasarkan macamnya, kunci dapat dibagi menjadi :
  - Kunci kandidat (*Candidate Key* / CK)
    Merupakan satu atau gabungan minimal atribut yang bersifat unik yang dapat digunakan untuk mengidentifikasi setiap record dalam relasi. Dalam setiap relasi minimal mempunyai sebuah kunci kandidat
  - Kunci primer (*Primary Key* / PK)
    Merupakan bagian dari CK yang dipilih sebagai kunci utama untuk mengidentifikasi setiap record dalam relasi. Dalam setiap relasi harus mempunyai satu PK yang bersifat unik dan tidak boleh kosong

---

  - Kunci penghubung (*Foreign Key* / FK)
    Disebut juga kunci tamu atau kunci asing, yaitu satu atau gabungan sembarang atribut yang menjadi PK dalam relasi lain yang mempunyai hubungan secara logik. FK tidak harus dimiliki dalam sebuah relasi. Jika FK muncul dalam sebuah relasi, maka FK tersebut akan menunjukkan adanya hubungan antar relasi dalam basis data. Relasi yang mengacu pada relasi lain disebut relasi anak, sedangkan relasi yang menjadi acuan disebut relasi induk. FK dan PK harus mempunyai tipe dan ukuran data yang sama
  - Kunci alternatif (*Alternate Key* / AK)
    Merupakan bagian dari CK yang tidak dipilih sebagai PK. Setiap relasi bisa memiliki atau tidak memiliki AK

Kerelasian Antar Relasi

- Relasi menyatakan sebuah tabel dalam basis data, sedangkan kerelasian menyatakan hubungan antar relasi dalam basis data
- Kerelasian antar relasi dapat ditunjukkan dengan menggunakan sebuah diagram yang disebut Diagram Kerelasian Antar Relasi
- Jenis-jenis kerelasian antar relasi dalam RDBM sama dengan jenis-jenis kerelasian antar relasi dalam ER-Model
- Langkah-langkah menggambar diagram kerelasian antar relasi adalah :
  - Tuliskan setiap relasi dan atribut pada setiap relasi dalam bentuk tabel satu kolom, dimana kepala tabel memuat nama relasi dan isi tabel memuat nama-nama atributnya

Kerelasian Antar Relasi

- Tentukan PK dan FK (jika ada) dalam setiap relasi. Berikan tanda bintang (*) pada PK dan dua bintang (**) pada FK
- Gambarkan kerelasian antar relasi dengan cara menghubungkan setiap FK dengan atribut yang sesuai pada relasi induknya dengan tanda garis
- Gambarkan jenis kerelasian antar entitas dengan menggunakan tanda panah ganda untuk jenis banyak dan sebuah mata panah untuk jenis satu

Penyimpangan (Anomali)

- Anomali adalah proses pada basis data yang memberikan efek samping yang tidak diharapkan (misalnya menyebabkan ketidakkonsistenan data atau membuat sesuatu data menjadi hilang ketika data lain dihapus)

- Ada 3 macam yaitu :
  - Anomali peremajaan
  - Anomali penyisipan
  - Anomali penghapusan

- Anomali Peremajaan
  - Anomali ini terjadi bila ada pengubahan pada sejumlah data yang tidak berguna, tetapi tidak seluruhnya diubah

Relasi Pesanan_Beli			
Pemasok	Kota	Barang	Jumlah
Kartika	Jakarta	Monitor	10
Citra	Bandung	ZIP-Drive	4
Candra	Jakarta	Keyboard	5
Citra	Bandung	Mouse	25

- Seandainya pemasok Citra berpindah ke kota lain, misalnya ke Bogor dan pengubahan hanya dilakukan pada data yang pertama saja, maka hasilnya akan menyebabkan data menjadi tidak konsisten

- Anomali Penyisipan
  - Anomali ini terjadi jika pada saat penambahan hendak dilakukan ternyata ada elemen data yang masih kosong dan elemen data tersebut justru menjadi kunci

Relasi Ruang_Kuliah		
Kuliah	Ruang	Tempat
Jaringan Komputer	Merapi	Gedung Utara
Pengantar Basis Data	Merbabu	Gedung Utara
Matematika I	Rama	Gedung Selatan
Sistem Pakar	Sinta	Gedung Selatan
Kecerdasan Buatan	Merapi	Gedung Utara

  - Relasi tersebut menyatakan
    - Kuiah menggunakan Ruang
    - Ruang berada disuatu Tempat
  - Jika dilakukan penambahan ruang baru bernama Arjuna yang berada di tempat Gedung Selatan, maka penyisipan tidak dapat dilakukan karena informasi tentang Kuliah yang menggunakan ruang tersebut tidak ada

- Anomali Penghapusan
  - Anomali ini terjadi jika suatu baris (tupel) yang tidak terpakai dihapus, maka akan mengakibatkan adanya data lain yang hilang
  - Contoh, pada relasi Ruang_Kuliah, jika data yang pertama dihapus yaitu kuliah Jaringan Komputer di ruang Merapi yang ada di Gedung Utara, maka informasi yang menyatakan ruang Merapi berada di Gedung Utara akan hilang

Ketergantungan Data (Data Dependency)

- Dependensi menjelaskan hubungan antar atribut atau secara lebih khusus menjelaskan nilai suatu atribut yang menentukan nilai atribut yang lainnya

- Ada 4 macam yaitu :
  - Dependensi Fungsional
  - Dependensi Fungsional Sepenuhnya
  - Dependensi Total
  - Dependesi Transitif

- Dependensi Fungsional
  - Suatu atribut Y mempunyai dependensi fungsional terhadap atribut X jika dan hanya jika setiap nilai X berhubungan dengan setiap nilai Y

- Contoh :

Relasi Pesanan_Jual			
Pembeli	Kota	Barang	Jumlah
P1	Yogya	B1	10
P1	Yogya	B2	5
P2	Solo	B1	7
P2	Solo	B2	6
P2	Solo	B3	6
P3	Klaten	B3	7
P3	Klaten	B4	6

  - Dependensi fungsional yang terjadi pada relasi diatas adalah :
    PEMBELI → KOTA
    { PEMBELI, BARANG } → {JUMLAH, KOTA}

- Dependensi Fungsional Sepenuhnya

  - Suatu atribut Y mempunyai dependensi fungsional penuh terhadap atribut X jika :
    - Y mempunyai dependensi fungsional terhadap X
    - Y tidak mempunyai dependensi terhadap bagian dari X
  - Contoh :

    Pelanggan (Kode_Pelanggan, Nama, Kota, Nomor_Fax)

    Pada relasi ini :
    - {Kode_Pelanggan, Kota} → Nomor_Fax
    - Kode_Pelanggan → Nomor_Fax

    Yang merupakan dependensi fungsional sepenuhnya adalah :

    Kode_Pelanggan → Nomor_Fax

- Dependensi Total

  - Suatu atribut Y mempunyai dependensi total terhadap atribut X jika :
    - Y mempunyai dependensi fungsional terhadap X
    - X mempunyai dependensi fungsional terhadap Y
  - Contoh :

Relasi Pemasok		
Kode_Pemasok	Nama_Pemasok	Kota
K1	Kartika	Jakarta
C1	Citra	Bandung
C2	Candra	Jakarta

    Yang merupakan dependensi total adalah :

    Kode_Pemasok→ Nama_Pemasok

    dengan asumsi bahwa tidak ada nama pemasok yang sama

- Dependensi Transitif
  - Suatu atribut Z mempunyai dependensi transitif terhadap atribut X jika:
    - Y mempunyai dependensi fungsional terhadap X
    - Z mempunyai dependensi fungsional terhadap Y
  - Contoh :

Relasi Kuliah			
Kuliah	Ruang	Tempat	Waktu
Jaringan Komputer	Merapi	Gedung Utara	Senin, 08.00 – 09.50
Matematika I	Rama	Gedung Selatan	Selasa, 07.00 – 08.45
Sistem Pakar	Sinta	Gedung Selatan	Rabu, 10.00 – 11.45
Fisika I	Merapi	Gedung Utara	Selasa, 08.00 – 08.50

Pada relasi ini :

Kuliah → {Ruang, Waktu}

Ruang → Tempat

Maka : Kuliah → Ruang → Tempat

Normalisasi

- Normalisasi adalah suatu proses untuk mengubah suatu relasi yang memiliki masalah tertentu ke dalam dua buah relasi atau lebih yang tidak memiliki masalah tersebut
- Masalah yang dimaksud adalah anomali
- Bentuk-bentuk normalisasi meliputi :
  - *First Norm Form (1NF), Second Norm Form (2NF), Third Norm Form (3NF)* dikemukakan oleh E.F. Codd
  - *Boyce-Codd Norm Form (BCNF)* dikemukakan oleh R.F. Boyce dan E.F. Codd
  - *Fourth Norm Form (4NF), Fifth Norm Form*
  - *Domain Key Norm Form (DKNF)*
  - *Restriction Union Norm Form (RUNF)*

- Relasi Bentuk Tidak Normal (*Un Normalized Form/UNF*)
  - Kriterianya :
    - Relasi memiliki bentuk *non flat file*
    - Relasi memuat set atribut yang berulang (*non single value*)
    - Relasi memuat atribut *non atomic value*
  - Contoh :

No_Siswa	Nama	Kelas_1	Wali_1	Kelas_2	Wali_2	Kelas_3	Wali_3
080100	Didi Riyadi	1234	Budi Rahmat	1543	Rizki Kurniawan		
080101	Nia Damayanti	1234	Budi Rahmat	1775	Citra Dewanti	1543	Rizki Kurniawan

- Relasi Bentuk Normal Pertama (*1NF*)
  - Suatu relasi dikatakan dalam bentuk normal pertama jika dan hanya jika:
    - Seluruh atribut dalam relasi bernilai atomik
    - Seluruh atribut dalam relasi bernilai tunggal
    - Relasi tidak memuat set atribut berulang
    - Semua record mempunyai sejumlah atribut yang sama
  - Dari contoh bentuk tidak normal sebelumnya, maka dapat diubah menjadi bentuk normal pertama sebagai berikut

No_Siswa	Nama	Kelas	Wali
080100	Didi Riyadi	1234	Budi Rahmat
080100	Didi Riyadi	1543	Rizky Kurniawan
080101	Nia Damayanti	1234	Budi Rahmat
080101	Nia Damayanti	1775	Citra Dewanti
080101	Nia Damayanti	1543	Rizky Kurniawan

- Relasi Bentuk Normal Kedua (*2NF*)
  - Suatu relasi dikatakan dalam bentuk normal kedua jika dan hanya jika
    - Berada pada bentuk normal pertama
    - Semua atribut bukan kunci memiliki dependensi fungsional sepenuhnya terhadap kunci primer
  - Dari hasil relasi bentuk normal pertama sebelumnya, misalnya diasumsikan kunci primernya adalah no_siswa, maka dependensi fungsional sepenuhnya yang terjadi adalah :

    No_Siswa → Nama

    { No_Siswa, Kelas } → Wali
  - Maka relasi dalam bentuk normal pertama tadi berdasarkan ketergantungan yang ada, harus dipecah menjadi dua buah relasi yaitu :

    Relasi Siswa (No_Siswa, Nama)

    Relasi Siswa_Kelas (No_Siswa, Kelas, Wali)

No_Siswa	Nama
080100	Didi Riyadi
080101	Nia Damayanti

No_Siswa	Kelas	Wali
080100	1234	Budi Rahmat
080100	1543	Rizky Kurniawan
080101	1234	Budi Rahmat
080101	1775	Citra Dewanti
080101	1543	Rizky Kurniawan

- Relasi Bentuk Normal Ketiga (*3NF*)
  - Suatu relasi dikatakan dalam bentuk normal ketiga jika dan hanya jika
    - Berada pada bentuk normal kedua
    - Semua atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci primer

- Dari hasil relasi bentuk normal kedua sebelumnya, terlihat adanya dependensi transitif yang terjadi pada relasi Siswa_Kelas yaitu :
  No_Siswa → Kelas → Wali

- Maka relasi Siswa_Kelas dalam bentuk normal kedua tadi berdasarkan ketergantungan yang ada, harus dipecah menjadi dua buah relasi yaitu :
  { No_Siswa, Kelas }
  Kelas → Wali

No_Siswa	Nama
080100	Didi Riyadi
080101	Nia Damayanti

No_Siswa	Kelas
080100	1234
080100	1543
080101	1234
080101	1775
080101	1543

Kelas	Wali
1234	Budi Rahmat
1543	Rizky Kurniawan
1775	Citra Dewanti

Latihan :

- Berikut diberikan relasi Supplier dalam bentuk tidak normal

Kode_Supplier	Status	Kota	Kode_Barang	Jumlah_Barang
S01	10	Jakarta	B01	100
			B02	150
			B03	200
S02	20	Surabaya	B02	250
			B04	200
S03	30	Yogyakarta	B05	150
			B06	100

Latihan :

- Berikut diberikan relasi KRS dalam bentuk tidak normal

NIM	Nama_Mahasiswa	Kode_MK_1	Sks_1	Tahun_Smt_1	Kode_MK_2	Sks_2	Tahun_Smt_2
001	Koko	MK01	2	20021	MK02	2	20022
002	Kiki	MK01	2	20021	MK02	2	20022
003	Kiko	MK01	2	20031	MK03	2	20032
004	Koki	MK01	2	20031	MK04	2	20032

Basis Data

SQL

Putu Manik Prihatini
gek_anik@yahoo.com

Definisi SQL

- SQL (*Structured Query Language*) merupakan bahasa query standar yang digunakan untuk mengakses basis data relasional

- Elemen SQL mencakup :
  - Pernyataan
  - Nama
  - Tipe Data
  - Konstanta
  - Ekspresi
  - Fungsi Bawaan

- Pernyataan adalah perintah SQL yang meminta sesuatu tindakan kepada DBMS
- SQL memiliki kira-kira 30 pernyataan
- Beberapa pernyataan dasar SQL adalah :

Pernyataan	Keterangan
ALTER	Mengubah struktur tabel
COMMIT	Mengakhiri sebuah eksekusi transaksi
CREATE	Menciptakan tabel, indeks atau view
DELETE	Menghapus baris pada tabel
DROP	Menghapus tabel, indeks atau view
GRANT	Menugaskan hak terhadap basis data kepada pengguna atau grup pengguna
INSERT	Menambahkan sebuah baris pada tabel
REVOKE	Membatalkan hak terhadap basis data
ROLLBACK	Mengembalikan ke keadaan semula sekiranya suatu transaksi gagal dilakukan
SELECT	Memilih baris dan kolom pada tabel
UPDATE	Mengubah nilai pada sebuah baris

- Nama digunakan sebagai identitas bagi objek-objek pada DBMS
- Setiap data memiliki tipe data
- Beberapa tipe data standar :

Tipe Data	Keterangan
CHAR	Menyatakan deretan karakter (string)
INTEGER	Menyatakan bilangan bulat
NUMERIC	Menyatakan bilangan real

- Beberapa tipe data perluasan :

Tipe Data	Keterangan
VARCHAR	Menyatakan string yang panjangnya bervariasi
MONEY	Menyatakan uang
BOOLEAN	Menyatakan tipe logika (true atau false)
BLOB	Menyatakan data biner
SERIAL / AUTOINCREMENT	Menyatakan nilai urut

- Konstanta menyatakan nilai yang tetap
- Beberapa contoh konstanta adalah :
  - Konstanta numerik : 123, -245, 5.45
  - Konstanta string : 'Jl. Sukapura 23'
- Ekspresi adalah segala sesuatu yang menghasilkan nilai, digunakan untuk menghitung nilai
- Contoh ekspresi untuk membagi isi variabel LABA dengan MODAL, dan kemudian dikalikan dengan 100 adalah :

  (LABA / MODAL) * 100
- Simbol-simbol yang dapat digunakan adalah *, /, +, -
- Fungsi adalah sebuah subprogram yang menghasilkan suatu nilai jika dipanggil
- SQL memiliki beberapa fungsi bawaan seperti MIN untuk memperoleh nilai terkecil atau AVG untuk memperoleh nilai rata-rata

Kelompok Pernyataan SQL

- DDL (*Data Definition Language*) merupakan perintah untuk mendefinisikan atribut-atribut basis data, tabel, atribut (kolom) dan batasan-batasan terhadap suatu atribut serta hubungan antar tabel
- Perintah DDL mencakup CREATE, ALTER, DROP

- DML (*Data Manipulation Language*) merupakan perintah untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan
- Perintah DML mencakup :
  - SELECT, memilih data
  - INSERT, menambah data
  - UPDATE, mengubah data
  - DELETE, menghapus data

- DCL (*Data Control Language*) merupakan perintah untuk mengendalikan pengaksesan data berdasarkan per pengguna, per tabel, per kolom ataupun per operasi yang boleh dilakukan
- Perintah DCL mencakup :
  - GRANT, memberikan kendali pengaksesan data
  - REVOKE, mencabut kemampuan pengaksesan data
  - LOCK TABLE, mengunci tabel

- Pengendali transaksi adalah perintah yang berfungsi untuk mengendalikan pengeksekusian transaksi
- Perintahnya mencakup :
  - COMMIT, menyetujui rangkaian perintah yang berhubungan erat (disebut transaksi) yang telah berhasil dilakukan
  - ROLLBACK, membatalkan transaksi yang dilakukan karena adanya kesalahan atau kegagalan pada salah satu rangkaian perintah

- Pengendali programatik mencakup pernyataan-pernyataan yang berhubungan dengan pemanfaatan SQL dalam bahasa lain (SQL yang dilekatkan)
- Pernyataan ini biasa dipakai pada bahasa konvensional (3-GL) seperti COBOL
- Perintahnya mencakup :
  - DECLARE, menutup kursor (pointer yang menunjuk ke tabel)
  - OPEN, mendeklarasikan kursor
  - FETCH, mengambil nilai baris berikutnya
  - CLOSE, membuka kursor

Menciptakan & Menghapus Tabel

- Tabel diciptakan melalui pernyataan CREATE TABLE
- Sebagai contoh :

```
CREATE TABLE tblpengarang (
        kd_peng     INTEGER,
        nama_peng   CHAR(15),
        alamat      CHAR(30),
        kota        CHAR(15))
```

- Setiap kolom dapat dilengkapi dengan UNIQUE dan NOT NULL
  - NULL, menyatakan bahwa nilai kolom bisa tidak diisi (default)
  - NOT NULL, menyatakan bahwa nilai suatu kolom harus diisi
  - UNIQUE, menyatakan bahwa nilai pada kolom tidak boleh ada yang sama (bersifat unik)
  - NOT UNIQUE, menyatakan bahwa nilai pada kolom boleh kembar (default)

- Sebagai contoh :

```
CREATE TABLE tblpengarang (
        kd_peng     INTEGER UNIQUE NOT NULL,
        nama_peng CHAR(15) NOT NULL,
        alamat      CHAR(30),
        kota        CHAR(15))
```

- Tabel diatas mensyaratkan bahwa :
  - kd_peng harus diisi dan bersifat unik
  - Nama harus diisi
  - Lainnya bisa tidak diisi

- Untuk menghapus tabel tblpengarang diatas dapat menggunakan perintah berikut :

```
DROP TABLE tblpengarang
```

Menciptakan & Menghapus Indeks

- Indeks biasa diciptakan dengan tujuan :
  - Indeks dapat meningkatkan kinerja
  - Indeks menjamin bahwa suatu kolom bersifat unik
- Sebagai contoh, jika seringkali terdapat perintah untuk mengurutkan pengarang berdasarkan nama pengarang, maka nama pengarang akan lebih baik kalau diindeks
- Indeks mempercepat pencarian data berdasarkan kolom yang diindeks, akan tetapi memperlambat proses penambahan dan penghapusan baris pada tabel karena saat terjadi penambahan atau penghapusan baris, indeks perlu diperbaharui
- Indeks diciptakan melalui pernyataan CREATE INDEX
- Sebagai contoh :

```
CREATE INDEX idx_nama
        ON tblpengarang(nama_peng)
```

- Jika yang diindeks adalah kolom yang nilainya bersifat unik, maka kata UNIQUE perlu ditambahkan
- Sebagai contoh :

```
CREATE UNIQUE INDEX idx_kd_peng
        ON tblpengarang(kd_peng)
```

- Jika indeks tersusun atas lebih dari satu kolom (untuk kunci komposit), maka bentuk penciptaan indeks berupa:

```
CREATE UNIQUE INDEX nama_indeks
        ON nama_tabel(kolom_x, kolom_y)
```

- Untuk menghapus indeks dapat menggunakan perintah berikut :

```
DROP INDEX nama_indeks
        ON nama_tabel
```

- Jika nama indeks yang akan dihapus hanya digunakan pada sebuah tabel, maka klausa ON nama_tabel tidak perlu ditulis

Mengubah Struktur Tabel

- Untuk mengubah struktur dari tabel yang sudah dibuat dapat menggunakan pernyataan ALTER TABLE
- Sebagai contoh, untuk menambahkan kolom bernama jenis_kelamin yang bertipe CHAR dengan panjang kolom 1 pada tabel tblpengarang adalah:

```
ALTER TABLE tblpengarang
        ADD jenis_kelamin CHAR(1)
```

- Untuk menghapus kolom tertentu pada suatu tabel, dapat menggunakan klausa DROP pada pernyataan ALTER TABLE
- Sebagai contoh, untuk menghapus kolom bernama jenis_kelamin pada tabel tblpengarang adalah:

```
ALTER TABLE tblpengarang
        DROP jenis_kelamin
```

Menambah Data

- Untuk memasukkan data ke tabel yang sudah dibuat dapat menggunakan pernyataan INSERT
- Sebagai contoh,

```
INSERT INTO tblpengarang
        VALUES (1, 'Ashadi', 'Jl. Beo 34', 'Yogya', 'P')
```

- Urutan nilai yang diletakkan dalam tanda kurung disesuaikan dengan urutan kolom dalam tabel
- INSERT juga memungkinkan penambahan data pada kolom-kolom tertentu dengan syarat kolom-kolom yang tidak disebutkan harus NOT NULL
- Sebagai contoh,

```
INSERT INTO tblpengarang (kd_peng, nama_peng)
        VALUES (11, 'Evi Tamala')
```

Mengubah Data

- Untuk mengubah data yang telah tersimpan dalam tabel dapat menggunakan pernyataan UPDATE
- Sebagai contoh,

```
UPDATE tblpengarang
        SET nama_peng = 'Evi Tamala Sari' WHERE kd_peng = 11
```

- Pada pernyataan diatas :
  - SET untuk menentukan kolom yang akan diubah dan nilai penggantinya
  - WHERE untuk menentukan kondisi dari baris-baris yang akan diganti
- Sebagai contoh,

```
UPDATE tblpengarang
        SET alamat = 'Jl. Pancakarya 5', kota = 'Semarang'
        WHERE kd_peng = 11
```

Menghapus Data

- Untuk menghapus data yang sudah tidak terpakai dalam tabel dapat menggunakan pernyataan DELETE
- Sebagai contoh,

```
DELETE FROM tblpengarang
        WHERE kd_peng = 11
```

- Bila klausa WHERE tidak disebutkan, maka seluruh baris pada tabel akan dihapus
- Sebagai contoh,

```
DELETE FROM tblpengarang
```

Tabel : tblpengarang				
Kd_peng	**Nama_peng**	**Alamat**	**Kota**	**Kelamin**
1	Ashadi	Jl. Beo 34	Yogya	P
2	Rian H	Jl. Solo 123	Yogya	P
3	Suadi Marwan	Jl. Semangka II/1	Bandung	P
4	Siti Halimah	Jl. Sukaria 5	Solo	W
5	Amir Hamzah	Jl. Gajah Mada 18A	Kudus	P
6	Suparman	Jl. Setia 1	Jakarta	P
7	Jaja M	Jl. Hangtuah 3	Bandung	P
8	Saman	Jl. Gedong Kuning	Yogya	P
9	Anwar Junaidi	Jl. Tidar 6A	Magelang	P
10	Fatmawati	Jl. Renjana 4	Bogor	W

Tabel : tblbuku		
Kd_buku	**Judul**	**Kd_peng**
1	Pemrograman C++	1
2	Pengantar Basis Data	1
3	Panduan Microsoft Office	2
4	Pemrograman Visual dBASE	1
5	Sistem Pakar	4
6	Pemrograman C++	3
7	Visual C++	6
8	QBASIC	5
9	Pemrograman Pascal	8
10	Pemrograman Delphi	8

Tabel : tblgaji	
Nama	**Gaji**
Rina	600000
Aji	700000
Sita	500000
Wiwid	1000000
Edi	600000
Vita	750000

Pernyataan SELECT

- SELECT merupakan pernyataan SQL yang berguna untuk menampilkan isi tabel
- Sebagai contoh,

```
SELECT kd_peng, nama_peng
        FROM tblpengarang
```

- Untuk menampilkan semua kolom yang terdapat pada tabel tblpengarang maka dapat menggunakan simbol * sesudah kata SELECT
- Sebagai contoh,

```
SELECT * FROM tblpengarang
```

- SELECT juga dapat digunakan untuk menampilkan baris-baris tertentu dengan menggunakan klausa WHERE
- Sebagai contoh,

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE kd_peng = 5
```

- Sebagai contoh lain,

```
SELECT nama_peng FROM tblpengarang
        WHERE kota = 'Yogya'
```

- Beberapa operator yang dapat digunakan selain '=' adalah :
  - > lebih dari
  - < kurang dari
  - <> tidak sama dengan
  - >= lebih dari atau sama dengan
  - <= kurang dari atau sama dengan

ORDER BY, GROUP BY, HAVING

- Untuk mengurutkan data menurut suatu kolom dapat menggunakan pernyataan ORDER BY
- Sebagai contoh, untuk menampilkan isi kolom kd_peng serta nama_peng yang diurutkan berdasarkan kolom nama_peng adalah :

```
SELECT kd_peng, nama_peng FROM tblpengarang ORDER BY nama_peng
```

- Untuk mengelompokkan data dapat menggunakan pernyataan GROUP BY
- Sebagai contoh, untuk menampilkan isi tabel tblpengarang yang dikelompokkan berdasarkan kota adalah :

```
SELECT kota FROM tblpengarang GROUP BY kota
```

- Klausa HAVING disediakan untuk mendukung klausa GROUP BY
- Sebagai contoh, untuk menampilkan kota yang jumlah pengarangnya lebih dari satu adalah :

```
SELECT nama_peng, kota FROM tblpengarang GROUP BY kota HAVING COUNT(kota) > 1
```

AVG, COUNT, MAX, MIN, SUM

- AVG adalah fungsi yang digunakan untuk menghitung rata-rata
- Sebagai contoh, untuk menghitung gaji rata-rata pada tabel tblgaji adalah :

```
SELECT AVG(gaji) FROM tblgaji
```

- COUNT adalah fungsi yang digunakan untuk menghitung cacah data
- Sebagai contoh, untuk menghitung jumlah pengarang per kota adalah :

```
SELECT kota, COUNT(kota) FROM tblpengarang GROUP BY kota
```

- Sebagai contoh, untuk menghitung jumlah pengarang adalah :

```
SELECT COUNT(*) FROM tblpengarang
```

- MAX adalah fungsi yang digunakan untuk memperoleh nilai terbesar
- Sebagai contoh, untuk menampilkan gaji terbesar pada tabel tblgaji adalah :

```
SELECT MAX(gaji) FROM tblgaji
```

- MIN adalah fungsi yang digunakan untuk memperoleh nilai terkecil
- Sebagai contoh, untuk menampilkan gaji terkecil pada tabel tblgaji adalah :

```
SELECT MIN(gaji) FROM tblgaji
```

- SUM adalah fungsi yang digunakan untuk memperoleh jumlahan data
- Sebagai contoh, untuk menampilkan total gaji pada tabel tblgaji adalah :

```
SELECT SUM(gaji) FROM tblgaji
```

OPERATOR

- Beberapa operator yang tersedia adalah AND, OR, NOT, BETWEEN-AND, IN dan LIKE

- Operator AND untuk menyatakan keadaan ‘dan’
- Sebagai contoh, untuk menampilkan daftar nama pengarang dengan kode 1 sampai 6 adalah :

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE kd_peng >= 1 AND kd_peng <= 6
```

- Operator OR untuk menyatakan keadaan ‘atau’
- Sebagai contoh, untuk menampilkan daftar nama pengarang yang tinggal di Yogya atau Solo adalah :

```
SELECT kd_peng, nama_peng, kota FROM tblpengarang
        WHERE kota = ‘Yogya’ OR kota = ‘Solo’
```

- Operator NOT untuk menyatakan keadaan ‘tidak’
- Sebagai contoh, untuk menampilkan daftar nama pengarang yang tidak tinggal di Yogya adalah :

```
SELECT kd_peng, nama_peng, kota FROM tblpengarang
        WHERE NOT kota = ‘Yogya’
```

- Operator BETWEEN-AND untuk menangani operasi ‘jangkauan’
- Sebagai contoh, untuk menampilkan daftar nama pengarang dengan kode 1 hingga 6 adalah :

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE kd_peng BETWEEN 1 AND 6
```

- Operator IN untuk menyatakan keadaan ‘salah satu diantara’
- Sebagai contoh, untuk menampilkan daftar nama pengarang yang tinggal di Yogya, Solo atau Magelang adalah :

```
SELECT kd_peng, nama_peng, kota FROM tblpengarang
        WHERE kota IN (‘Yogya’, ‘Solo’, ‘Magelang’)
```

- Operator LIKE digunakan untuk pencocokan
- Sebagai contoh, untuk menampilkan daftar nama pengarang yang diawali dengan huruf ‘A’ adalah :

```
SELECT nama_peng, kota FROM tblpengarang
        WHERE nama_peng LIKE ‘A%’
```

Tanda % berarti “nol, satu atau sejumlah karakter apa saja”

SUBQUERY

- Subquery berarti query didalam query
- Subquery terletak didalam klausa WHERE atau HAVING
- Pada klausa WHERE, subquery digunakan untuk memilih baris-baris tertentu yang kemudian digunakan oleh query
- Pada klausa HAVING, subquery digunakan untuk memilih kelompok baris yang kemudian digunakan oleh query
- Sebagai contoh, untuk menampilkan daftar kode dan nama pengarang yang kode pengarangnya tercantum pada tabel tblbuku adalah :

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE kd_peng IN
                (SELECT kd_peng FROM tblbuku)
```

Pada contoh diatas :

`SELECT kd_peng FROM tblbuku` disebut subquery

`SELECT kd_peng, nama_peng` disebut query

Operator EXIST

- Operator EXISTS digunakan untuk memeriksa keberadaan baris yang dihasilkan oleh query terhadap yang dihasilkan oleh subquery
- Operator EXISTS menghasilkan True jika subquery menghasilkan baris yang sesuai dengan yang dihasilkan query
- Sebagai contoh,

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE kd_peng IN
                (SELECT kd_peng FROM tblbuku)
```

dapat ditulis menjadi :

```
SELECT kd_peng, nama_peng FROM tblpengarang
        WHERE EXISTS
                (SELECT * FROM tblbuku
                WHERE kd_peng = tblpengarang.kd_peng)
```

Operator ANY, ALL

- Operator ANY menghasilkan True jika paling tidak salah satu pembandingan dengan hasil subquery menghasilkan nilai True
- Sebagai contoh, untuk menampilkan semua nama yang gajinya bukan yang terkecil adalah :

```
SELECT nama_peng FROM tblgaji
        WHERE gaji > ANY
                (SELECT gaji FROM tblgaji)
```

- Operator ALL menghasilkan True jika subquery tidak menghasilkan apapun atau jika pembandingan menghasilkan True untuk setiap nilai query terhadap hasil subquery
- Sebagai contoh, untuk menampilkan semua nama yang bergaji paling besar adalah :

```
SELECT nama_peng FROM tblgaji
        WHERE gaji >= ALL
                (SELECT gaji FROM tblgaji)
```

Operator UNION

- Operator UNION digunakan untuk menggabungkan hasil query
- Sebagai contoh, untuk menampilkan nama pengarang yang tinggal di Yogya dan Solo adalah :

```
SELECT nama_peng, kota FROM tblpengarang
        WHERE kota = 'Yogya'
UNION
SELECT nama_peng, kota FROM tblpengarang
        WHERE kota = 'Solo'
```

- Pernyataan diatas identik dengan :

```
SELECT kd_peng, nama_peng, kota FROM tblpengarang
        WHERE kota = 'Yogya' OR kota = 'Solo'
```

Namun tidak semua penggabungan dapat dilakukan dengan OR

Operasi JOIN

- JOIN merupakan operasi yang digunakan untuk menggabungkan dua tabel atau lebih dengan hasil berupa gabungan dari kolom-kolom yang berasal dari tabel-tabel tersebut
- Sebagai contoh, untuk menampilkan nama pengarang (milik tblpengarang) dan judul buku (milik tblbuku) adalah :

```
SELECT nama_peng, judul FROM tblpengarang, tblbuku
        WHERE tblbuku.kd_peng = tblpengarang.kd_peng
```

- Sebagai contoh, untuk menampilkan semua pengarang yang data buku karangannya terdapat pada tabel tblbuku adalah :

```
SELECT DISTINCT nama_peng FROM tblpengarang, tblbuku
        WHERE tblbuku.kd_peng = tblpengarang.kd_peng
```

Kata DISTINCT digunakan agar kalau ada pengarang yang menulis lebih dari satu buku maka hanya akan ditampilkan sekali

- Operator AND dapat ditambahkan pada klausa WHERE
- Sebagai contoh, untuk menampilkan semua judul buku yang ditulis oleh Ashadi adalah :

```
SELECT judul FROM tblpengarang, tblbuku
        WHERE tblbuku.kd_peng = tblpengarang.kd_peng
        AND nama_peng = 'Ashadi'
```

- Klausa ALIAS dapat digunakan pada klausa FROM
- Sebagai contoh, untuk menampilkan dua pasang pengarang yang tinggal di kota yang sama, dengan ketentuan seseorang tidak boleh berpasangan dengan dirinya sendiri :

```
SELECT alias1.namapengarang, alias1.kotapengarang,
alias2.namapengarang, alias2.kotapengarang
FROM tbpengarang alias1, tbpengarang alias2
WHERE alias1.kotapengarang = alias2.kotapengarang AND
alias1.kdpengarang = alias2.kdpengarang
```