

BASIS DATA



Ayu Manik Dirgayusari, S.Kom., M.MT.
Nazaruddin Ahmad, M.T.
Bagus Tri Mahardika, M.MSI.
Musyrifah, S.Pd., M.Pd.
Husna Gemasih, S.Inf., M.Cs.
Asmawati S, S.Kom., M.Pd.
Asep Abdul Sofyan, S.Kom., M.Kom.
Djamaludin, S.Kom., M.Kom.
Nurul Aini, S.Kom., M.T.
Wiyanto, S.Kom., M.Kom.
Mohammad Ridwan, S.Kom., M.Kom.
Muhammad Yani, S.Kom., M.T.I.
Herianto, S.Pd., MT.
Donny Maulana, S.Kom., M.M.Si.

BOOK CHAPTER
BASIS DATA

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

- i Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
- ii Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
- iii Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan
- iv Pengumuman sebagai bahan ajar; dan Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

BASIS DATA

Ayu Manik Dirgayusari, S.Kom., M.MT.

Nazaruddin Ahmad, M.T.

Bagus Tri Mahardika, M.MSI.

Musyrifah, S.Pd., M.Pd.
Husna Gemasih, S.Inf., M.Cs.

Asmawati S, S.Kom., M.Pd.

Asep Abdul Sofyan, S.Kom., M.Kom.

Djamaludin, S.Kom., M.Kom.

Nurul Aini, S.Kom., M.T.

Wiyanto, S.Kom., M.Kom.

Mohammad Ridwan, S.Kom., M.Kom.

Muhammad Yani, S.Kom., M.T.I.

Herianto, S.Pd., MT.
Donny Maulana, S.Kom., M.M.Si.

Editor:

Dudih Gustian, S.T., M.Kom.

Penerbit



CV. MEDIA SAINS INDONESIA
Melong Asih Regency B40 - Cijerah
Kota Bandung - Jawa Barat
www.penerbit.medsan.co.id

Anggota IKAPI
No. 370/JBA/2020

BASIS DATA

Ayu Manik Dirgayusari, S.Kom., M.MT.

Nazaruddin Ahmad, M.T.

Bagus Tri Mahardika, M.MSI.

Musyrifah, S.Pd., M.Pd.

Husna Gemasih, S.Inf., M.Cs.
Asmawati S, S.Kom., M.Pd.

Asep Abdul Sofyan, S.Kom., M.Kom.

Djamaludin, S.Kom., M.Kom.

Nurul Aini, S.Kom., M.T.

Wiyanto, S.Kom., M.Kom.

Mohammad Ridwan, S.Kom., M.Kom.

Muhammad Yani, S.Kom., M.T.I.

Herianto, S.Pd., MT.

Donny Maulana, S.Kom., M.M.Si.

Editor :

Dudih Gustian, S.T., M.Kom.

Tata Letak :

Mega Restiana Zendrato

Desain Cover :

Syahrul Nugraha

Ukuran :

A5 Unesco: 15,5 x 23 cm

Halaman :

vi, 203

ISBN :

978-623-362-475-6

Terbit Pada :

April 2022

Hak Cipta 2022 @ Media Sains Indonesia dan Penulis

Hak cipta dilindungi Undang-Undang. Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari Penerbit atau Penulis.

PENERBIT MEDIA SAINS INDONESIA

(CV. MEDIA SAINS INDONESIA)

Melong Asih Regency B40 - Cijerah

Kota Bandung - Jawa Barat
www.penerbit.medsan.co.id

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya sehingga buku kolaborasi dalam bentuk book chapter Basis Data dapat dipublikasikan dan dapat sampai di hadapan pembaca. Book chapter ini disusun oleh sejumlah akademisi dan praktisi sesuai dengan kepakarannya masing-masing. Buku ini diharapkan dapat hadir memberi kontribusi positif dalam ilmu pengetahuan khususnya terkait dengan Basis Data.

Sistematika buku Basis Data ini mengacu pada pendekatan konsep teoritis dan contoh penerapan. Buku ini terdiri atas 14 bab yang dibahas secara rinci, diantaranya: Pengantar basis data, Lingkungan basis data, Model basis data relational, Relational key, Bahasa model relational, Pengenalan SQL berserta contohnya, Advanced SQL, RDBMS, Pengenalan database, Oracle, Konsep dasar ER, Transformasi ER ke model data relational, Normalisasi, Studi kasus ERD ke model data relational.

Kami menyadari bahwa tulisan ini jauh dari kesempurnaan dan masih terdapat banyak kekurangan, sejatinya kesempurnaan itu hanya milik yang kuasa. Oleh sebab itu, kami tentu menerima masukan dan saran dari pembaca demi penyempurnaan lebih lanjut.

Akhirnya kami mengucapkan terima kasih yang tak terhingga kepada semua pihak yang telah mendukung dalam proses penyusunan dan penerbitan buku ini, secara khusus kepada Penerbit Media Sains Indonesia sebagai inisiator book chapter ini. Semoga buku ini dapat bermanfaat bagi pembaca sekalian.

Bandung 30 Maret 2022
Editor.

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
1 PENGANTAR BASIS DATA	1
Sejarah Basis Data	1
Definisi Basis Data	2
Tujuan Pemanfaatan Basis Data.....	5
Komponen Sistem Basis Data	8
Penerapan Basis Data.....	10
2 LINGKUNGAN BASIS DATA.....	15
Pendahuluan	15
Lingkungan Basis Data.....	15
3 MODEL BASIS DATA RELASIONAL.....	31
Model Data Relasional	31
Kelebihan.....	31
Karakteristik Model Basisdata Relasi	31
Istilah Model Basisdata Relasi	32
Kardinalitas (Derajat Relasi).....	36
Jenis-Jenis “Key” Dalam Relasi Database	38
Integritas Referensial	39
Batasan Integritas	41
Relasi yang Berstruktur Baik.....	41
Basisdata yang Baik	41
4 <i>REALASIONAL KEY (SUPER KEY, CANDIDATE KEY, PRIMARY KEY, ALTERNATIF)</i>	43
<i>Super Key</i>	45
<i>Candidate Key</i>	47

<i>Primary Key</i>	48
<i>Alternative Key</i>	50
5 BAHASA PADA MODEL RELASIONAL	55
Pendahuluan	55
Aljabar Relasional	56
Kalkulus Relasional	66
6 PENGENALAN SQL BESERTA CONTOHNYA	71
Pendahuluan	71
Sejarah SQL.....	72
Apa Pentingnya SQL	73
Penulisan Perintah SQL	73
<i>Data Definition Language (DDL)</i>	74
<i>Data Manipulation Language (DML)</i>	76
<i>Data Control Language (DCL)</i>	80
Perintah SQL Menggunakan Operator Dasar	82
7 ADVENCED SQL (EMBEDDED DAN DYNAMIC)....	89
Asep Abdul Sofyan, S.Kom., M.Kom.....	89
<i>Advenced SQL</i>	89
<i>Embedded SQL</i>	91
Dinamic SQL.....	96
Pendekatan dan Teknik Lain.....	99
8 RDBMS (<i>RELATIONAL DATABASE MANAGEMENT SYSTEM</i>)	103
RDBMS.....	103
Sejarah Sistem Pengelolaan Basis Data Relasional (Relational Database Management System/RDBMS).....	113

Perangkat Lunak RDBMS	113
Jenis Relasi Tabel	114
Fungsi RDBMS	115
Perbedaan RDBMS dengan DBMS	116
Kelebihan dari RDBMS	117
Kekurangan dari RDBMS.....	117
9 PENGENALAN DATABASE 2	121
Pengetian MySQL.....	121
Fungsi MySQL	122
Kelebihan dan Kekurangan MySQL.....	122
Akses Ke MySQL.....	125
10 Membuat Database di MySQL.....	127
PENGENALAN ORACLE	133
Pengenalan <i>Oracle</i>	133
Sejarah <i>Oracle</i>	134
Perkembangan <i>Oracle</i>	135
Membuat Basis Data dengan <i>Oracle</i>	136
Membuat Table Menggunakan Menu SQL Workshop – SQL Commands.....	140
11 KONSEP DASAR MODEL ER.....	147
Tinjauan Model ER	147
Komponen ER Diagram.....	149
Langkah-Langkah Membuat ERD (E-R Digram).....	156
12 TRANSFORMASI ER KE MODEL DATA RELASIONAL.....	163
Pendahuluan	163
Model Data Relasional	163

Keuntungan Model Data Relasional	164
Istilah-istilah dalam Model Data Relasional	164
Aturan Umum dalam Transformasi ER Menjadi Model Data Relasional	165
Transformasi Model ER untuk Kardinalitas Satu ke Satu.....	168
Transformasi Model ER untuk Kardinalitas Satu ke Banyak	169
Transformasi Model ER untuk Kardinalitas Banyak ke Banyak.....	171
Implementasi Transformasi ER dengan Entitas Lemah (Weak Entity).....	172
Transformasi Model ER untuk Relasi Khusus IS-A	174
Studi Kasus Transformasi ER ke Tabel Fisik Menggunakan ERDPLUS	175
13 NORMALISASI BASIS DATA.....	179
Pengertian dan Tujuan Normalisasi	179
Tabel Universal.....	179
Proses Normalisasi.....	182
Ketergantungan Fungsional (KF)	184
First Normal Form (1 NF)	185
Second Normal Form (2 NF)	186
Third Normal Form (3 NF)	188
Boyce Codd Normal Form (BC NF)	190
Fourth Normal Form (4 NF).....	191

14	STUDI KASUS ERD DAN NORMALISASI	195
	Studi Kasus Erd	195
	Menentukan Entitas	195
	Menentukan Relasi dengan Matrik Relasi	195
	Gambar Erd Sementara.....	196
	Gambaran Kardinalitas.....	196
	Menentukan Kunci Utama	197
	Mengambarkan Erd Berdasarkan Kunci.....	197
	Menentukan Atribut	198
	Memetakan Atribut.....	198
	Menggambarkan Erd dengan Atribut.....	198
	Studi Kasus Normalisasi (1)	198
	Studi Kasus Normalisasi (2)	200

PENGANTAR BASIS DATA

Ayu Manik Dirgayusari, S.Kom., M.MT.

Institut Bisnis dan Teknologi Indonesia

Sejarah Basis Data

Komputer berevolusi dari masa ke masa dibarengi dengan perkembangan teknologinya, baik teknologi perangkat

Dalam pemakaiannya *software* selalu mengalami peningkatan baik digunakan untuk pengolahan data ataupun sebagai penunjang kinerja komputer. Dalam fungsinya sebagai pengolah data, *software* harus dipastikan dapat mengolah data dengan akurat. Apabila pengolahan data dilakukan dengan baik maka akan menghasilkan informasi yang akurat pula. Sehingga tidak akan ada data dan informasi yang tidak konsisten.

Dalam pengolahan data terutama data digital dikenal istilah yang disebut dengan basis data. Berawal dari istilah pengarsipan atau arsip, basis data lebih banyak dikenal dalam bidang teknologi. Pengetahuan tentang pengarsipan data atau basis data semakin berkembang seiring dengan kebutuhan akan pengaksesan data terutama dalam pengolahan data yang besar. Perkembangan konsep basis data diawali dari awal tahun 1960 hingga tahun 1990-an. Pada awal tahun 1960 adalah titik awal pembentukan konsep basis data dimana sebelumnya pengolahan data masih dilakukan secara manual. Titik awal pembentukan konsep basis data ini disebut dengan tahap 1 dimana data yang diolah berdasarkan prinsip pemrosesan berkas (*file processing*) pada lingkungan komputer. Tahap II yaitu pada akhir

1960 dimana konsep basis data telah banyak mengalami peningkatan yaitu mengubah dan memperbaiki metode penyimpanan pada basis data. Ciri-ciri dari konsep basis data tahap II ini adalah sudah mengenal *Database System/DBS*, mengenal *Database Management System/DBMS*, berbagi informasi secara *online* dan berbasis teks.

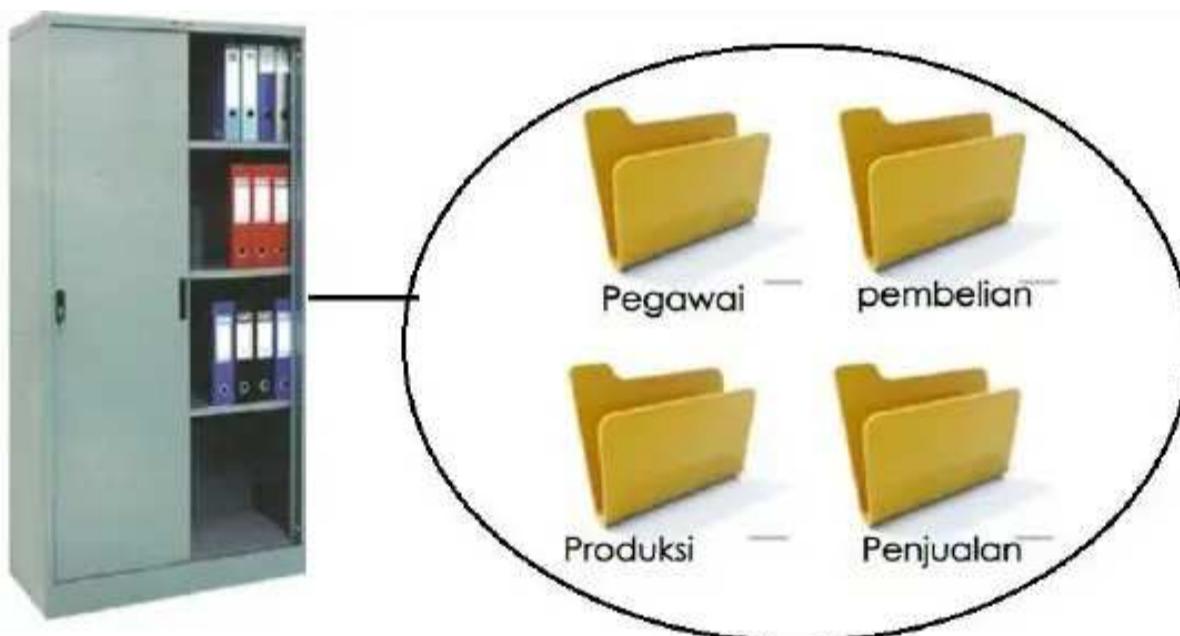
Perkembangan metode penyimpanan data dalam basis data meningkat pesat pada awal 1970 dan dikenal dengan perkembangan tahap III. Peningkatan tidak hanya pada *software* melainkan *hardware* tempat untuk menyimpan data. *Software* yang digunakan yaitu untuk basis data dalam system pakar yaitu membantu dalam pengambilan keputusan (*Decision Support System/DSS*) dan pemrograman berorientasi objek (*Object Oriented Programming/OOP*). Pada perkembangan konsep basis data tahap IV yaitu tahun 1980an, mengalami perkembangan sangat cepat yang dibarengi dengan perkembangan teknologi komputer. Ciri dari konsep basis data tahap IV adalah sistem dengan berbasis *hypertext* yaitu dapat menampilkan informasi dengan mencarinya hanya memasukkan kata kunci dari informasi yang akan dicari. Pada tahun 1990an adalah perkembangan tahap V dari basis data, dimana basis data dikembangkan pada aplikasi-aplikasi sistem kecerdasan buatan (*Artificial Intelligent/AI*), aplikasi multimedia, aplikasi basis data yang berorientasi objek (*Object Oriented Database/OODB*), *online database* dan aplikasi berbasis *fuzzy*. (Sutanta, 2011)

Definisi Basis Data

Istilah basis data sudah tidak asing pada era teknologi informasi seperti ini. Basis data dapat diartikan sebagai tempat penyimpanan atau lemari arsip untuk menyimpan segala jenis data. Basis Data terdiri dari kata Basis dan Data. Basis yang berarti markas atau tempat bersarang atau gudang. Sedangkan Data terdapat beberapa pengertian yaitu:

1. Data merupakan segala bentuk fakta dan angka yang dapat dijadikan bahan untuk menyusun suatu informasi.(Arikunto, 2002)
2. Data merupakan kumpulan kejadian atau peristiwa yang terjadi di dunia nyata yang berupa angka-angka, huruf-huruf, simbol-simbol khusus atau gabungan dari semuanya. (Kuswayatno, 2006)
3. Data adalah representasi dari fakta dunia yang mewakili suatu obyek yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.(Fathansyah, 2018)

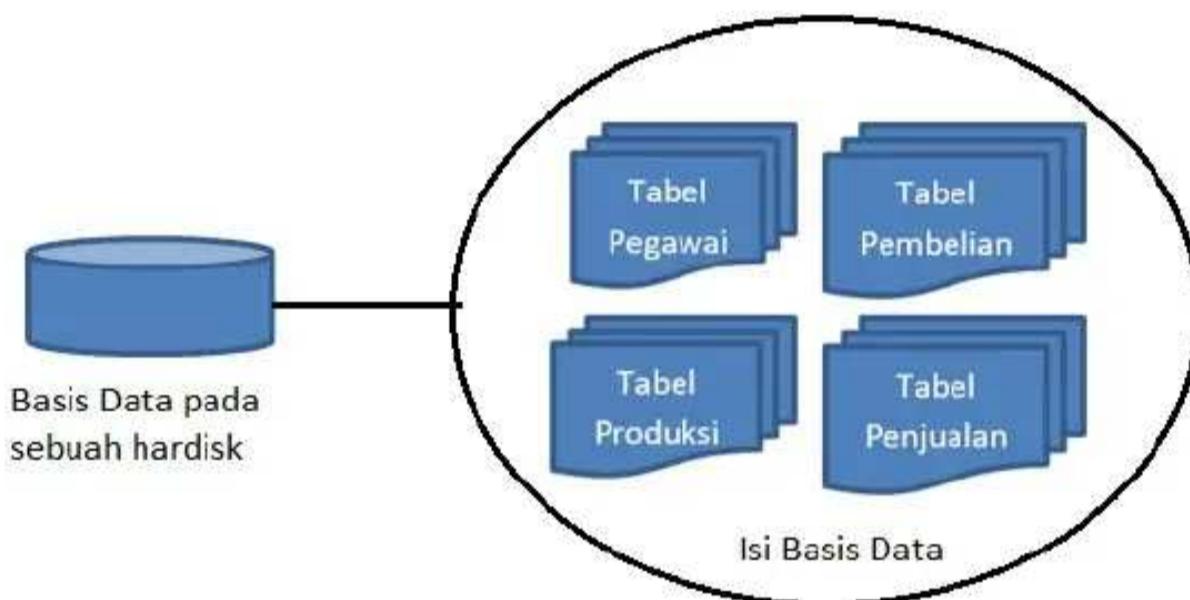
Jadi dapat disimpulkan bahwa Data adalah segala bentuk fakta yang terjadi di dunia nyata yang dapat mewakili suatu objek yang disimpan dalam bentuk angka, huruf, simbol, teks, gambar, suara atau gabungannya.



Gambar 1.1 Arsip Data pada Lemari Arsip

Basis data dapat diilustrasikan sebagai lemari arsip dengan beberapa cara pengaturan datanya. Basis data dan lemari arsip memiliki prinsip kerja dan tujuan yang sama yaitu pengaturan data atau arsip agar lebih mudah dan cepat pada saat dibutuhkan atau diakses. Namun pengaturan data atau arsip pada lemari arsip masih banyak terdapat kelemahan, antara lain data atau arsip yang sama dapat tersimpan lebih dari satu kali sehingga jika data atau arsip tersebut dibutuhkan akan terjadi data yang tidak konsisten atau *inconsistency data*. Jika data yang dibutuhkan tidak konsisten maka informasi yang

dihasilkan juga menjadi tidak akurat. Karena terjadi duplikasi data atau arsip sehingga integritas data tidak dapat dipertanggung jawabkan serta tidak dapat mengakses data yang sama dalam waktun yang bersamaan. Tujuan dari dibuatnya basis data adalah agar pada saat suatu data dibutuhkan, data tersebut dapat diakses dengan mudah dan cepat. Jadi penyimpanan data dengan lemari arsip dirasa masih kurang maksimal pada saat arsip tersebut akan diakses. Perbedaan lain dari pengarsipan dengan lemari arsip dan basis data terletak dari media penyimpanannya.



Gambar 1.2 Arsip Data pada Basis Data

Media penyimpanan pada lemari arsip yaitu sebuah lemari besi dengan beberapa kolom-kolom untuk meletakkan kumpulan berkas dan biasanya berupa berkas cetak pada kertas. Sedangkan suatu basis data yaitu data yang tersimpan dalam suatu media penyimpanan elektronik berupa *magnetic disk* yang terdapat pada komputer. Metode penyimpanannya dilakukan dengan logika tertentu tergantung dari algoritma produsen komputer tersebut. Jadi definisi basis data jika dilihat dari beberapa sudut pandang, antara lain: (Fathansyah, 2018)

1. Himpunan kelompok data (arsip) yang saling berhubungan dan diorganisir sedemikian rupa sehingga dapat diakses kembali dengan cepat dan mudah

2. Kumpulan data yang saling berhubungan, disimpan secara bersama sedemikian rupa dan tanpa ada pengulangan penyimpanan data atau di sebut *redundancy*
3. Kumpulan file atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan tertentu

Jadi dari beberapa definisi tersebut dapat disimpulkan bahwa, **Basis Data** adalah kumpulan data yang saling berelasi atau berhubungan dan dapat diakses kembali dengan cepat dan mudah, tanpa ada *redundancy* yang disimpan dalam suatu media penyimpanan tertentu.

Tujuan Pemanfaatan Basis Data

Basis data memiliki tujuan adalah pengaksesan data menjadi lebih mudah dan cepat sehingga informasi yang di dapatkan juga akan menjadi lebih cepat. Adapun penjabaran pemanfaatan dari basis data adalah : (Fathansyah, 2018)

1. Kecepatan dan Kemudahan (*Speed*)

Penerapan basis data dalam penyimpanan dan pengelolaan data akan lebih cepat dan mudah pada saat pengaksesan data dibandingkan dengan penyimpanan secara manual ataupun penyimpanan data elektronik yang tidak menggunakan relasi data.

2. Efisiensi Ruang Penyimpanan (*Space*)

Data yang dikelola sering kali terjadi pengulangan penyimpanan atau disebut redundansi, sehingga mempengaruhi kapasitas ruang penyimpanan. Semakin banyak data yang disimpan maka memerlukan ruang penyimpanan yang besar pula. Jika menggunakan basis data maka pengulangan data akan diminimalisir sehingga kapasitas ruang penyimpanan akan lebih efisien penggunaannya.

3. Keakuratan (*Accuracy*)

Dalam penerapan basis data, semua data yang ada akan berelasi dengan data lainnya yang memiliki keterikatan atau ketergantungan. Sehingga dalam setiap pengelolaannya akan mempersempit kesalahan serta data pada saat diakses akan menghasilkan informasi yang lebih akurat.

4. Ketersediaan (*Availability*)

Pertumbuhan data semakin lama akan semakin banyak dan akan membutuhkan tempat penyimpanan yang lebih besar. Dari semua data yang ada, tidak semua data akan selalu diakses. Data yang tidak terpakai dapat diatur kembali sehingga dapat mengurangikapasitas pemakaian ruang penyimpanan. Dengan bantuan jaringan komputer, data tidak harus tersimpan di setiap komputer melainkan dapat disimpan pada komputer lain dan dapat diakses oleh komputer manapun. Jadi data apapun pada saat diakses akan selalu tersedia dimanapun data tersebut disimpan.

5. Kelengkapan (*Completeness*)

Agar data yang dikelola senantiasa lengkap baik terhadap kebutuhan pemakai maupun terhadap waktu, dengan melakukan penambahan baris-baris data ataupun melakukan perubahan struktur pada basis data, yakni dengan menambahkan field pada tabel atau menambah tabel baru.

6. Keamanan (*Security*)

Agar data yang bersifat rahasia atau proses yang vital tidak jatuh ke orang/pengguna yang tidak berhak, yakni dengan penggunaan account (username dan password) serta menerapkan pembedaan hak akses setiap pengguna terhadap data yang bisa dibaca atau proses yang bisa dilakukan.

7. Pemakaian Bersama (*Sharability*)

Agar data yang dikelola oleh sistem mendukung lingkungan multiuser (banyak pemakai), dengan

menjaga / menghindari munculnya problem baru seperti **inkonsistensi data** (karena terjadi perubahan data yang dilakukan oleh beberapa user dalam waktu yang bersamaan) atau kondisi **deadlock** (karena ada banyak pemakai yang saling menunggu untuk menggunakan data).

8. Pemusatan Kontrol Data

Pada saat sebuah data dilakukan perubahan, tidak perlu dilakukan perubahan disemua data. Perubahan hanya terjadi di satu tempat pada tempat data yang mengalami perubahan dan data yang telah dirubah dapat diakses kembali oleh semua pengguna (Kusrini, 2007).

9. Kemudahan dalam Pembuatan Aplikasi

Dalam pembuatan sebuah aplikasi yang membutuhkan basis data, tidak perlu melakukan pembuatan basis data dari awal. Sebuah DBMS akan membantu mengontrol tempat penyimpanan datanya, jadi pembuat aplikasi hanya berfokus pada pembuatan interface (Kusrini, 2007).

10. Kebebasan Data

Jika sebuah aplikasi telah selesai dibuat, jika ada perubahan struktur dari basis data maka tidak perlu melakukan perubahan pada keseluruhan dataatau aplikasi. Dengan DBMS perubahan tersebut hanya dilakukan samapi level DBMS (Kusrini, 2007).

11. Userview

Pada basis data dapat diatur hak akses setiap user dan pandangan dari data yang dapat diakses. Contohnya pada basis data akademik, user mahasiswa hanya dapat mengakses data pribadi masing-masing dan tidak dapat mengakses data mahasiswa lain. Beda halnya dengan user staf akademik, mereka dapat mengakses semua data mahasiswa termasuk banyak SKS yang ditempuh, status mahasiswa termasuk tunggakan mahasiswa bukan hanya data pribadi. Beda halnya lagi dengan

bagian keuangan, mereka selain bisa mengakses semua data pribadi mahasiswa mereka juga dapat mengakses tunggakan biaya perkuliahan semua mahasiswa.

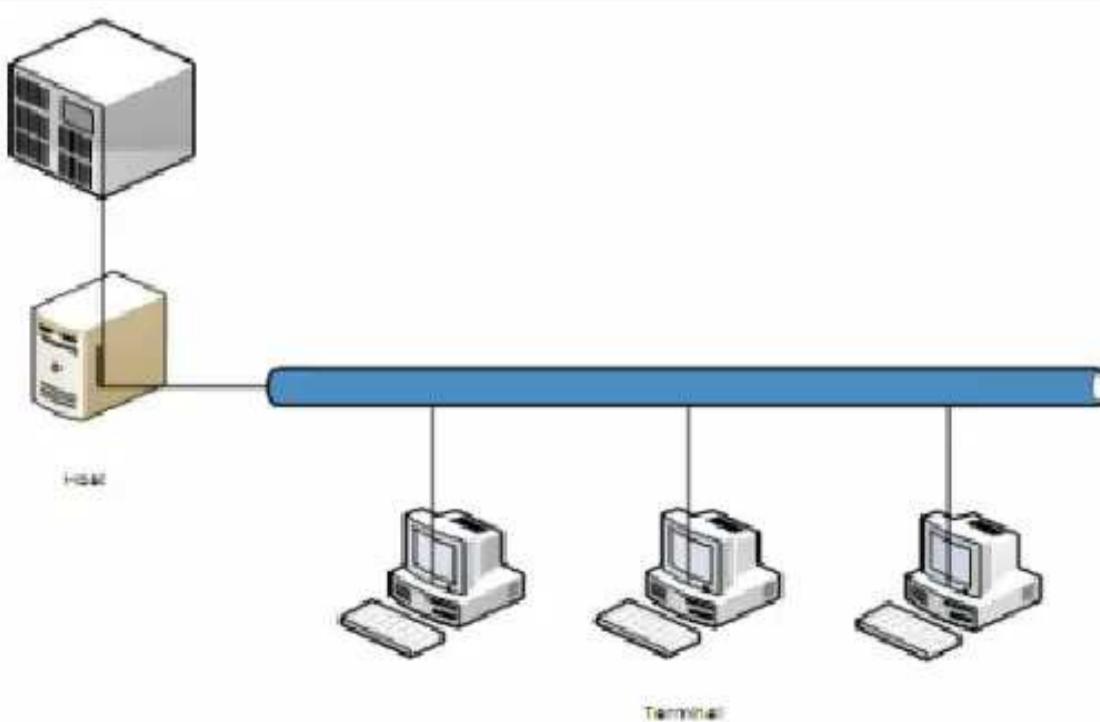
Komponen Sistem Basis Data

Basis data adalah sebuah objek yang pasif sehingga diperlukan sebuah aplikasi atau program untuk mengelolanya. Penggabungan hal tersebut disebut sistem. Sehingga Sistem Basis Data merupakan sistem yang terdiri atas kumpulan table data yang saling berhubungan (basis data) dan *Data Management System* (DBMS) yang memungkinkan user untuk mengakses atau pengelolaan data pada basis data tersebut.(Fathansyah, 2018)

Sistem basis data dapat berkerja dengan baik jika didukung oleh beberapa komponen berikut ini:

1. Perangkat keras (*Hardware*)

Perangkat keras yang umum digunakan pada sebuah sistem basis data adalah komputer *stand-alone* atau lebih dari satu komputer untuk dapat berbagi data dengan menggunakan jaringan komputer. Selain itu diperlukan juga memori sebagai tempat penyimpanan data, contohnya *Harddisk*. Serta perangkat keras untuk komunikasi antara beberapa computer menggunakan jaringan komputer.



Gambar 1.3 Sistem Jaringan Komputer

2. Sistem Operasi (*Operating System/OS*)

Sistem operasi adalah sebuah aplikasi yang dimasukkan/di-instal pada perangkat keras sebagai tempat untuk menjalankan aplikasi-aplikasi pendukung untuk menjalankan sistem komputer. Terdapat beberapa sistem operasi yang dikenal saat ini, seperti: Windows, Linux ataupun Macintosh. Jadi aplikasi pendukung pada sebuah sistem basis data harus menyesuaikan dengan sistem operasi yang digunakan. Misalnya sistem operasi windows, sebaiknya menggunakan DBMS yang dapat berjalan baik pada sistem operasi windows.

3. Basis Data (*Database*)

Basis data adalah inti dari sistem basis data yang dibuat. Dalam basis data menampung semua komponen data dan relasinya. Pada basis data terdapat table, index serta struktur data dan relasinya.

4. *Database Management System (DBMS)*

DBMS adalah software atau aplikasi yang menangani semua akses ke database. DBMS yang akan menentukan bagaimana data ditangani, diorganisir, disimpan, diubah dan diakses kembali. DBMS juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama dan konsistensi data.



Gambar 1.4 Database Management System (DBMS)

5. Pemakai (*User*)

Pemakai atau user merupakan orang atau sistem yang akan mengelola, memakai dan merubah isi basis data. Adapun beberapa jenis pengguna sistem basis data yaitu:

- a. Programmer Aplikasi: orang yang mengkodekan aplikasi dengan bahasa pemrograman
 - b. User mahir: orang yang mampu menggunakan basis data dengan menggunakan DBMS
 - c. User umum / *End User*: orang yang memakai basis data dengan bantuan program aplikasi. Misalnya seorang akunting memasukkan data keuangan menggunakan aplikasi keuangan.
 - d. User khusus: dapat berupa sistem lain
6. Aplikasi pendukung lain
Aplikasi ini merupakan aplikasi pendukung untuk memberikan tampilan aplikasi yang mudah digunakan untuk mengelola basis data dan ini sifatnya opsional.

Penerapan Basis Data

Suatu basis data adalah koleksi data yang bisa mencari secara menyeluruh dan secara sistematis memelihara dan me-retrive informasi. Suatu basis data dapat terkomputerisasi atau tidak terkomputerisasi (Janner, 2007).

Beberapa basis data yang tidak terkomputerisasi adalah buku telepon, lemari penyimpanan surat dan sistem katalog kartu perpustakaan. Untuk meretrive informasi dari masing-masing basis data tersebut, dapat memulainya dengan cara berikut:

1. Untuk melihat nomor telepon teman, kita akan mencarinya dalam buku telepon. Melihat kota dimana mereka tinggal dan mencari-cari nama belakang yang diikuti dengan nama depannya. Jika terdapat lebih dari satu nama, anda perlu memeriksa alamatnya. Buku telepon sangat sederhana dengan struktur yang terbatas. Contoh buku telepon yaitu yang telah sering

dijumpai yaitu yellow pages. Contohnya dapat dilihat pada gambar 1.5.



Gambar 1.5 Buku Telepon

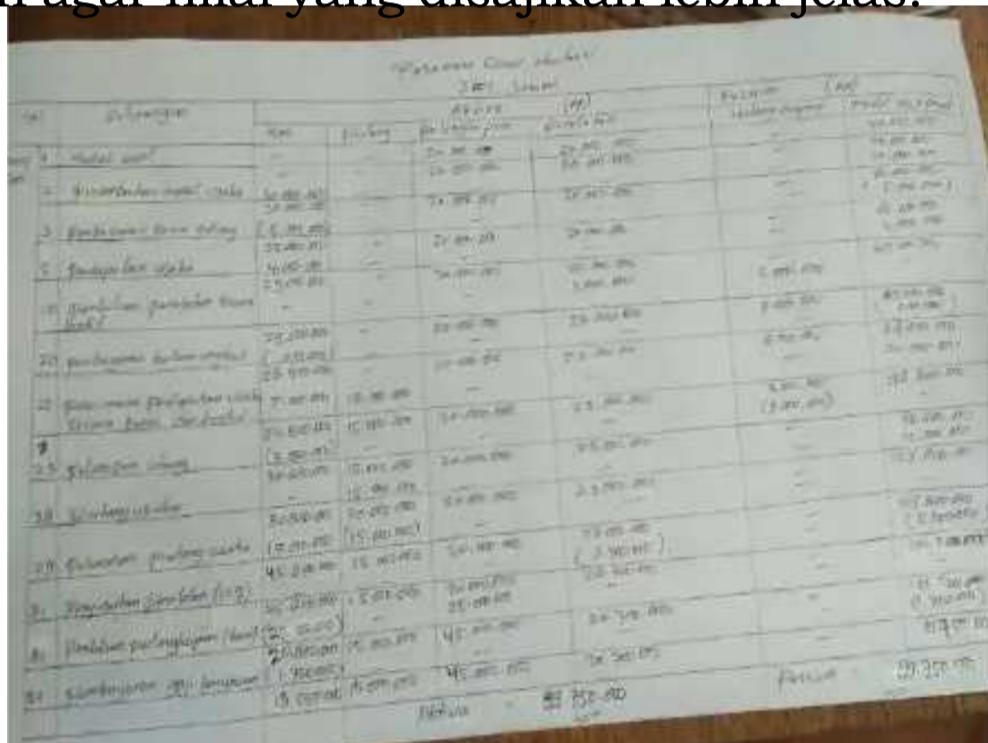
Jika dibuatkan dalam bentuk terkomputerisasi maka akan tercipta sebuah basis data untuk menampung data kontak atau nomor dan nama dari teman atau kenalan kita. Biasanya data yang akan dimasukkan ke dalam database adalah nama, yang mengandung nama depan dan nama belakang. Selain nama, database juga menyimpan data nomor telefon, nomor ponsel, alamat email ataupun alamat rumah. Contohnya dapat dilihat pada gambar 1.6. Jika ingin mencari data nama hanya tinggal mengetikkan nama pada kolom cari, data yang dicari akan dengan mudah ditemukan.



Gambar 1.6 Aplikasi Buku Telepon

Sumber: <http://sourcecodeapplication.blogspot.com/>

2. Untuk memeriksakan sisa keuangan pelanggan, kita perlu mengambilnya dalam lemari penyimpanan surat, menempatkan map pelanggan dan mencabut potongan kertas untuk menyeimbangkan neracanya. Basis data tersebut juga memiliki struktur yang terbatas. Sebagai contoh, melihat record pelanggan beserta alamatnya atau nama dari perwakilan penjualnya. Lagipula untuk melihat semua pelanggan dengan saldo yang lebih dari Rp 1.000.000,00 anda harus membuka seluruh map hingga menemukan kriteria pelanggan yang dicari. Contohnya pada gambar 1.7, Seluruh data di catat manual pada kertas dan apa bila data pada salah satu baris dicari maka harus teliti dalam mencarinya. Selain itu sangat rentan dengan kesalahan, jika terjadi kesalahan maka harus mengubah keseluruhan neraca. Jika terjadi perubahan, sebaiknya mengubah keseluruhan halaman agar nilai yang disajikan lebih jelas.

A photograph of a handwritten financial report on lined paper. The table has four columns: 'No', 'Customer Name', 'Debit', and 'Credit'. The 'Debit' column contains amounts such as 10.000.000, 2.000.000, 1.000.000, etc. The 'Credit' column contains amounts such as 10.000.000, 2.000.000, 1.000.000, etc. There are several rows of data, with some rows having multiple entries under 'Debit' and 'Credit'. The handwriting is in black ink on white paper.

Gambar 1.7 Contoh laporan keuangan tertulis

Jika dibuatkan dalam bentuk terkomputerisasi maka akan digunakan sebuah database untuk menampung data customer, banyak barang yang dibeli, jumlah dan total harga yang dibayarkan. Contoh dapat dilihat pada gambar 1.8.

Buku Besar		
1111 - KAS		
OKTOBER	2015	
TANGGAL	DEBIT	KREDIT
KETERANGAN	SALDO	
01/10/2015	0	
Saldo Awal	0	
01/10/2015	10.000.000	
Membeli mesin	10.000.000	
02/10/2015	2.000.000	
Pendapatan usaha	12.000.000	
06/10/2015		300.000
Bayar	11.700.000	
10/10/2015	1.000.000	
Hutang ke SELBA	12.700.000	
12/10/2015		100.000
Vembayaran hutang ke	12.600.000	
SALDO AKHIR		9.600.000

Gambar 1.8 contoh laporan dengan database

Sumber: <https://carisinyal.com/aplikasi-laporan-keuangan/>

Daftar Pustaka

- Arikunto, S. (2002). Metodologi Penelitian Suatu Pendekatan Proposal. PT. Rineka Cipta.
- Fathansyah. (2018). Basis Data (Edisi Keti). Penerbit INFORMATIKA.
- Janner, S. (2007). Perancangan Basis Data. Penerbit ANDI.
- Kusrini. (2007). Strategi Perancangan dan Pengelolaan Basis Data (A. H. Triyuliana (ed.); First Edition). CV. ANDI OFFSET (PENERBIT ANDI).
- Kuswayatno, L. (2006). Mahir dan Terampil Berkomputer. Penerbit Grafindo Media Pratama.
- Sutanta, E. (2011). Basis Data Dalam Tinjauan Konseptual (First). Penerbit ANDI.
- Sumber: <http://sourcecodeapplication.blogspot.com/>
- Sumber: <https://carisinyal.com/aplikasi-laporan-keuangan/>

Profil Penulis



Ayu Manik Dirgayusari

Lahir dan besar di Denpasar, Bali. Pendidikan Taman Kanak-kanak hingga Sekolah Menengah Atas diselesaikan di Kota Denpasar. Setelah menamatkan sekolah menengah atas, penulis merantau untuk melanjutkan Pendidikan S1 di STIKOM Surabaya yang sekarang berganti nama menjadi Universitas Dinamika dan S2 Manajemen Teknologi di Institut Teknologi Sepuluh Nopember (ITS) di Surabaya. Sejak SMP sudah senang dengan dunia komputer, sehingga pada saat kuliah mengambil jurusan Sistem Komputer konsentrasi Rekayasa Perangkat Lunak dan mendalami bidang Manajemen Teknologi dengan bidang keahlian Manajemen Teknologi Informasi pada saat mengambil S2. Saat ini penulis aktif sebagai Dosen di salah satu kampus swasta di kota Denpasar yaitu di Institut Bisnis dan Teknologi Indonesia atau biasa disebut INSTIKI. Penulis sebagai dosen pengampu mata kuliah Algoritma dan Pemrograman, Human Computer Interaction serta Database. Sebagai seorang dosen, selain aktif melakukan penelitian, penulis juga aktif menulis buku dengan harapan ilmu dan pengalaman yang dimilikinya dapat dibagikan kepada khalayak ramai.

Email penulis: ayu.manik@stiki-indonesia.ac.id

2

LINGKUNGAN BASIS DATA

Nazaruddin Ahmad, M.T.

Universitasi Islam Negeri Ar-Raniry Banda Aceh

Pendahuluan

Basis data memberikan tujuan untuk menyediakan kepada pemakai melalui suatu pandangan abstrak mengenai data dengan tidak memperlihatkan secara lengkap bagaimana data disimpan dan dimanipulasi. Langkah awal dalam merancang basis adalah harus abstrak dan memiliki deskripsi umum dari kebutuhan-kebutuhan informasi suatu organisasi haruslah digambarkan di dalam basis data.

Basis data merupakan suatu sumber yang dapat digunakan secara bersama-sama maka setiap pemakai membutuhkan tingkat pandangan yang berbeda terhadap data di dalam basis data. Untuk memenuhi kebutuhan ini maka dibutuhkan tingkatan arsitektur basis data yang dapat membantu pemakai untuk dapat melihat data di dalam basis data.

Lingkungan Basis Data

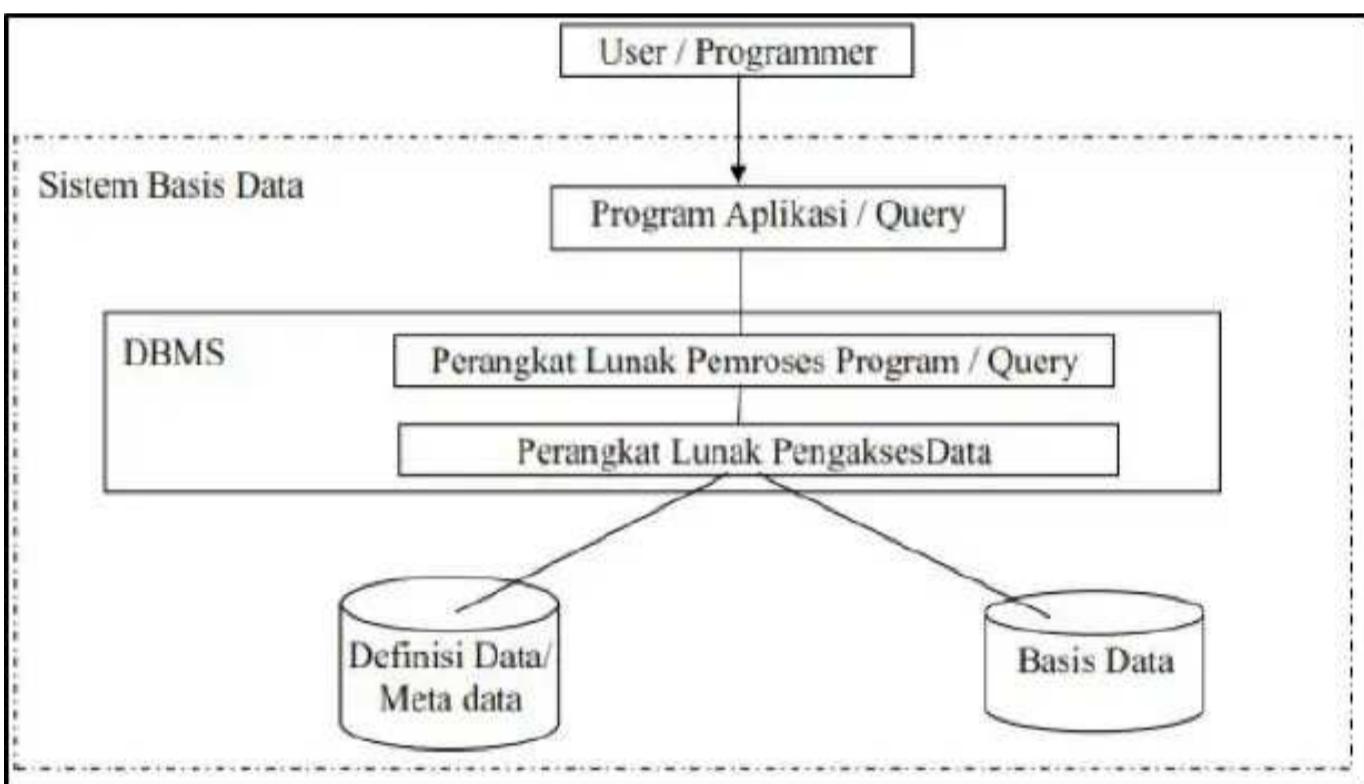
Lingkungan basis data terdiri dari beberapa unsur, yaitu:

1. Komponen Basis Data

Basis data terdiri atas lingkungan pembentuk sistem basis data. Lingkungan pembentuk sistem basis data ini terdiri dari komponen-komponen yang saling terkait satu dengan lainnya, sehingga semua komponen saling berinteraksi dan menghasilkan kerja

basis data yang baik. Komponen-komponen tersebut antara lain:

- a. Perangkat keras (*hardware*)
- b. Sistem operasi (*operating system*)
- c. Basis data (*database*)
- d. Sistem pengelola basis data (*database management system/DBMS*)
- e. Pemakai (*user*)
- f. Aplikasi lain



Gambar 2.1. Lingkungan Sistem Basis Data

a. Perangkat Keras (*hardware*)

Perangkat keras (*hardware*) merupakan seperangkat komputer yang terdiri dari komponen bagian dalam komputer dan komponen bagian luar komputer yang digunakan untuk mengelola basis data.

Perangkat keras yang biasa terdapat dalam sebuah sistem basis data adalah komputer, memori sekunder *on-line* (*harddisk*), memori sekunder *off-line* (*tape/removable disk*) sebagai media penyimpanan *backup* data, dan media perangkat komunikasi.

Komponen perangkat keras (*hardware*) tersebut terdiri dari:

- 1) Komputer (digunakan sebagai *client* dan digunakan secara sistem jaringan).
- 2) Memori sekunder yang *online*. Contohnya: *harddisk*.
- 3) Memori sekunder yang *offline*. Contohnya: *removable disk* untuk keperluan *backup* data.
- 4) Perangkat komunikasi yang digunakan untuk sistem jaringan.

b. Sistem Operasi (*operating system*)

Perangkat lunak sistem operasi berfungsi untuk menghidupkan semua komponen yang terdapat pada sistem komputer. Seluruh sumber daya

komputer dikendalikan oleh sistem operasi. Sistem operasi dasar yang dilakukan pada komputer juga dikendalikan oleh sistem operasi.

Sistem komputer diaktifkan dan difungsikan oleh sistem operasi untuk melakukan operasi input/output, mengelola file, melakukan perintah eksekusi dan lainnya. Terdapat beberapa sistem operasi yang banyak digunakan untuk komputer yang *stand alone* atau untuk komputer *client* yang digunakan di dalam sistem jaringan. Sistem operasi juga dikembangkan untuk penggunaan

pada komputer server di dalam jaringan. Program aplikasi basis data dapat berjalan (*running*) apabila sistem operasinya telah bekerja atau telah aktif terlebih dahulu.

c. Basis Data

Basis data terdiri dari sejumlah objek seperti file/tabel, indeks dan lainnya. Basis data digunakan untuk dapat menyimpan data, basis data juga berisi tentang penjelasan struktur tabel dan berisi penjelasan objek-objek lainnya secara lengkap.

Data di dalam basis data mempunyai sifat terpadu (*integrated*) dan berbagi (*shared*).

- 1) **Terpadu (*integrated*)** yaitu isi-isi data yang terdapat pada basis data memiliki keterhubungan satu sama lain.

Sebagai contoh:

Isi data dosen memiliki atribut nama, alamat, jurusan dan gaji serta atribut-atribut lainnya.

Berkas pelatihan berisi atribut nama, kursus atau pelatihan yang diikuti oleh dosen.

Pada saat melakukan pencatatan pelatihan yang diikuti oleh seorang dosen, data jurusan tempat dosen bekerja tidak perlu lagi dicantumkan pada isi data pelatihan karena data tersebut telah ada pada isian data dosen.

Sehingga dapat terhindar dari pengisian data secara berulang-ulang.

- 2) **Berbagi Data (*shared*)** berarti bahwa data dapat dipakai secara bersama-sama oleh sejumlah pengguna. Atau suatu data dapat diakses oleh sejumlah pengguna dalam waktu bersamaan.

d. Sistem Pengelolaan Basis Data (*Database Management System*)

Perangkat DBMS akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali serta menerapkan mekanisme pengamanan data, pemakaian data bersama, dan pelaksanaan keakuratan atau konsistensi data.

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak yang khusus dan spesifik. Perangkat lunak inilah yang disebut dengan DBMS yang akan menentukan bagaimana data diorganisasikan, disimpan,

diubah dan diambil kembali. DBMS juga menerapkan mekanisme pengamanan data,

pemakaian data secara bersama, pemaksanaan keakuratan atau konsistensi data, dan sebagainya. Pengembangan DBMS dilakukan agar pengelolaan basis data dapat dilakukan secara efektif dan efisien (Budio et al., 2019).

Contoh perangkat lunak yang termasuk DBMS seperti dBase III+, dBase IV, FoxBase, Rbase, MS-Access dan Borland Paradox (untuk kelas sederhana) atau Borland Interbase, MS-SQLServer, CA-Open Ingres, Infomix dan SysBase (untuk kelas kompleks/berat).

e. Pemakai (*User*)

Pemakai (*user*) yang berinteraksi dengan suatu sistem basis data dapat dibedakan kepada beberapa macam yang dilihat dari cara berinteraksi dengan sistem basis data, yaitu:

- 1) Database Administrator (DBA), adalah orang yang bertanggung jawab terhadap pengelolaan basis data. Secara lebih detail, tugas DBA adalah sebagai berikut: mendefinisikan basis data, DBA menentukan isi basis data, menentukan sekuritas basis data.
- 2) Programmer aplikasi adalah orang yang membuat program aplikasi yang menggunakan basis data. Program aplikasi yang dibuat disesuaikan dengan kebutuhan pengguna.
- 3) Pengguna akhir (*end user*), pada pengguna akhir dapat dibedakan menjadi dua jenis yaitu pengguna aplikasi dan pengguna interaktif.
 - a) Pengguna aplikasi merupakan orang yang berinteraksi dengan cara mengoperasikan program aplikasi yang dibuat oleh programmer aplikasi.
 - b) Pengguna interaktif adalah orang yang berinteraksi dengan cara dapat

memberikan perintah-perintah seperti SELECT, INSERT dan sebagainya.

User khusus (*specialized user*) merupakan pemakai yang menulis aplikasi basis data non konvensional, tetapi keperluan-keperluan, seperti untuk aplikasi AI (*Artificial Intelligence*), Sistem Pakar, Pengolahan Citra, dan lain-lain yang bisa saja mengakses basis data dengan/tanpa DBMS yang bersangkutan.

f. Aplikasi lain

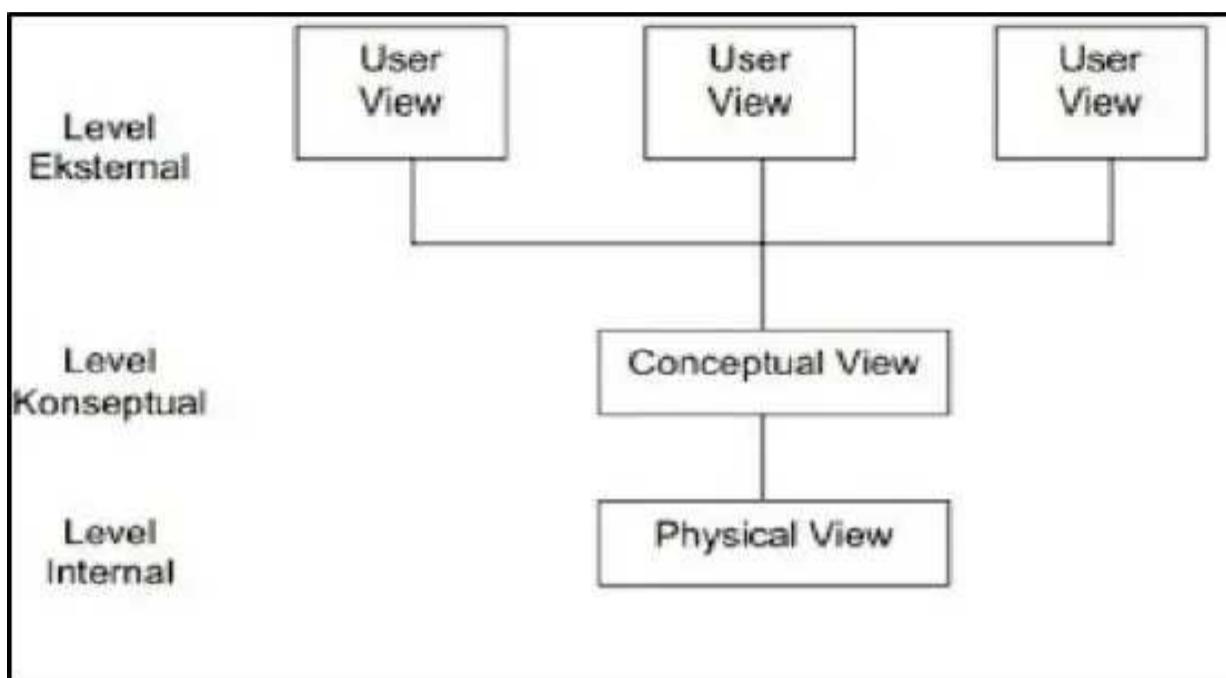
Aplikasi lain adalah perangkat lunak lainnya yang dapat digunakan untuk mengelola basis data. Perangkat lain ini dapat berupa perangkat lunak yang digunakan untuk merancang atau mendesain basis data dalam bentuk diagram-diagram sebelum diubah menjadi tabel-tabel di dalam suatu *database*. Dapat juga berupa perangkat lunak yang digunakan untuk memanipulasi tabel-tabel dan record-record di dalam suatu *database* menggunakan perintah SQL (*structured query language*).

Aplikasi yang mendukung dalam proses pembuatan basis data ini bersifat opsional, boleh ada dan boleh juga tidak, hal ini tergantung kepada pemakai untuk kebutuhannya di dalam basis data. DBMS yang pamakai gunakan lebih berperan dalam pengorganisasian data dalam basis data, sementara bagi pemakai basis data (khususnya yang menjadi *end-user/native user*) dapat dibuatkan/disediakan program khusus/lain untuk melakukan pengisian, pengubahan dan pengambilan data.

2. Abstraksi Data

Salah satu tujuan dari DBMS adalah untuk menyediakan fasilitas atau antar muka (*interface*) dalam melihat data (yang lebih ramah/*user oriented*) kepada pemakai (*user*). Untuk itu, sistem tersebut akan menyembunyikan detail tentang bagaimana data

disimpan dan dipelihara. Abstraksi data merupakan tingkatan atau *level* dalam bagaimana melihat data dalam sistem basis data.



Gambar 2.2 Abstraksi Data

Ada 3 level abstraksi data (Fatansyah, 2018), yaitu:

a. Level fisik (*physical level*)

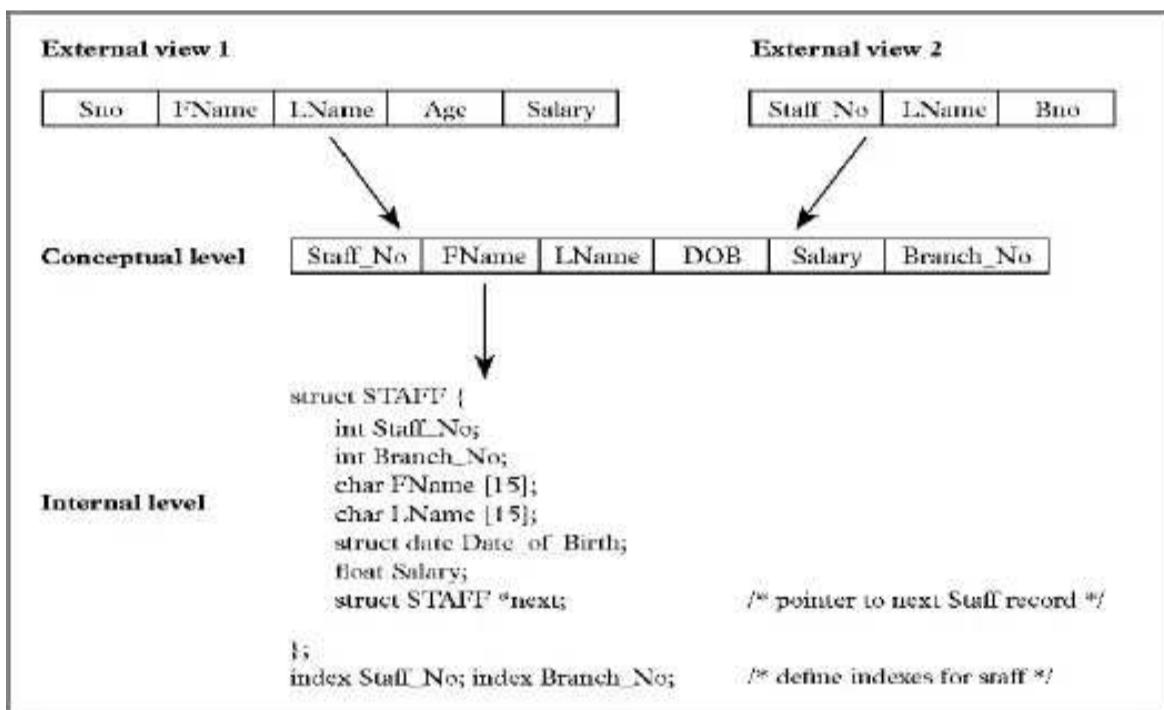
Level ini merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana sesungguhnya suatu data disimpan. Pada level ini pemakai melihat data sebagai gabungan dari struktur dan datanya sendiri. Pemakai juga berkompeten dalam mengetahui bagaimana representasi fisik dari penyimpanan atau pengorganisasian data. Pada level ini pemakai berurusan dengan data sebagai teks, angka atau bahkan melihatnya sebagai himpunan bit data.

b. Level logik/konseptual (*conceptual level*)

Level ini merupakan data apa yang sebenarnya (secara fungsional) disimpan dalam basis data dan hubungannya dengan data yang lain. Pada level ini misalnya mengetahui bahwa data pegawai disimpan atau dipresentasikan dalam beberapa file/tabel.

c. Level penampakan (*view level*)

Level ini merupakan level tertinggi dari abstraksi data yang hanya menunjukkan sebagian dari basis data. Banyak pengguna (*user*) dalam sistem basis data tidak akan terlibat (*concern*) dengan semua data atau informasi dalam basis data yang kemunculannya dimata pemakai diatur oleh aplikasi *end user*. Data yang diperlihatkan juga bisa saja tidak berasal dari hanya sebuah tabel tapi mewakili relasi antar tabel, tapi bagi pemakai menggunakannya terasa sebagai satu kesatuan yang kompak.



Gambar 2.3 Contoh Penggambaran Abstraksi Data

Gambar 2.3 memperlihatkan contoh lapisan-lapisan dalam abstraksi data, di level eksternal (*view level*) membahas tentang setiap pengguna memandang data dengan caranya masing-masing, pada setiap pengguna pasti akan memandang data dengan cara pandangnya masing-masing sesuai dengan kebutuhan masing-masing pengguna. Pada level konseptual menggumpulkan semua data-data yang dibutuhkan oleh setiap pengguna dengan berdasarkan data apa saja yang dibutuhkan oleh basis data yang akan dibuat. Pada level internal/fisik membahas tentang bagaimana data-data yang telah didefinisikan oleh level konseptual dapat diwujudkan ke dalam basis

data disini berbicara tentang tipe data dan bagaimana data tersebut diterjemahkan kedalam basis data.

3. Data Independence

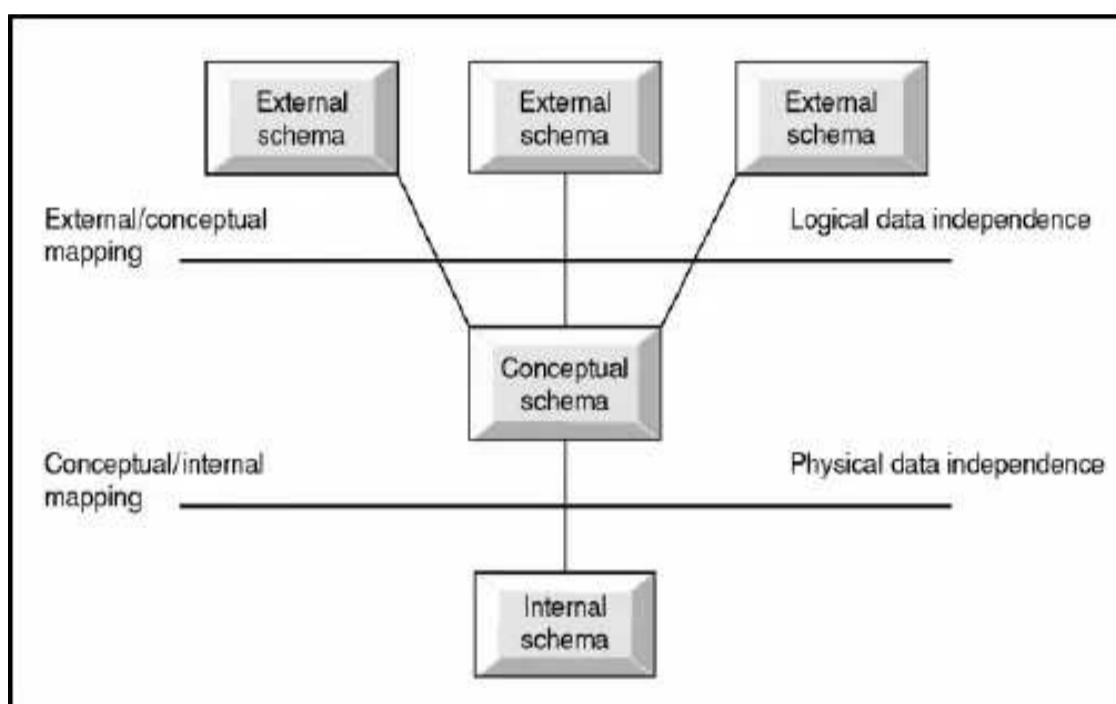
Tiga tingkatan abstraksi data bertujuan untuk memelihara kemandirian data atau data independen yang memiliki arti bahwa jika perubahan yang terjadi pada salah satu tingkatan maka tidak mempengaruhi tingkatan yang lebih rendah. Data independen merupakan jenis transparasi data yang penting untuk DBMS terpusat. Transparansi disini berbeda dengan penggunaan kata transparansi pada bahasa indonesia yang diartikan sebagai terlihat jelas. Tetapi kata transparasi data dapat diartikan bahwa menyembunyikan detail yang ada dengan menggunakan lapisan-lapisan yang ada.

Data independen terdapat dua tingkatan dalam basis data yaitu:

- a. Tingkat pertama adalah struktur logika dari data atau *the logic structure of the data*, tingkat pertama ini berada diantara level eksternal (*view*) dengan level konseptual (*logic*) dengan nama ***Logical data independence***. Logical data independence berfungsi jika ada perubahan di level konseptual (*logic*) yang dilakukan oleh DBA maka perubahan tersebut tidak akan mempengaruhi level eksternal (*view*). Dengan kata lain *logical data independence* menunjukkan kekebalan level eksternal (*view*) terhadap perubahan pada level konseptual (*logic*).
- b. Tingkat kedua dalam data independen adalah struktur fisik data atau *the physical structure of the data*, tingkatan ini berada diantara level konseptual (*logic*) dengan level internal atau fisik. Tujuan tingkatan kedua ini adalah menyembunyikan detail dari struktur penyimpanan dari aplikasi pengguna. Dalam tingkat kedua ini terdapat jenis data independence yaitu ***physical data independence***.

berfungsi jika ada perubahan level internal/fisik oleh DBA maka perubahan tersebut tidak akan mempengaruhi level konseptual. Dengan kata lain physical data independence menunjukkan kekebalan pada level internal atau fisik terhadap perubahan yang dilakukan pada level internal atau fisik.

Kedua tingkatan pada data independence terlihat pada gambar 2.4.



Gambar 2.4 Lapisan Data Independence

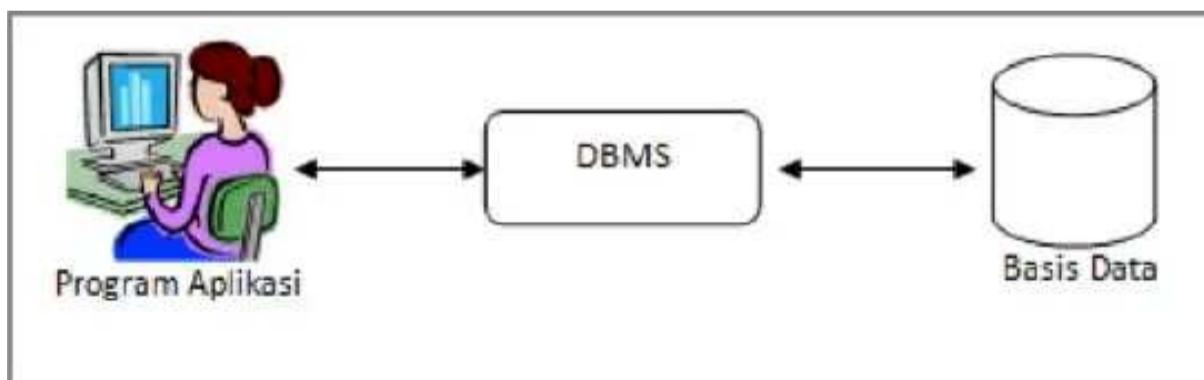
Data independence berfungsi untuk perubahan-perubahan seperti dibawah ini:

- DBA dapat mengubah isi, lokasi, perwujudan dalam organisasi basis data tanpa harus mengganggu program-program aplikasi yang sudah ada.
- Pabrik/agen peralatan/sofware pengolahan data dapat memperkenalkan produk-produk baru tanpa harus mengganggu program-program aplikasi yang sudah ada.
- Mempermudah dalam pengembangan-pengembangan pada program-program aplikasi
- DBA diberikan kontrol terpusat demi keamanan dan integritas data dengan memperhatikan perubahan-perubahan kebutuhan pengguna.

4. Bahasa Basis Data

Untuk dapat berinteraksi dengan data yang tersimpan maka pengguna basis data harus memahami perintah-perintah yang digunakan untuk dapat memanggil data yang tersimpan tersebut. Data yang dipanggil tidak saja berasal dari satu tabel yang ada di dalam *database*, tetapi data tersebut dapat dipanggil dari beberapa tabel secara sekaligus. Perintah-perintah tersebut disusun dalam bentuk bahasa *query* yang terdapat didalam tools *database management system* (DBMS) sehingga memudahkan pemakai untuk dapat melakukan definisi data, membuat, melakukan akses, memelihara data, menyimpan data, memperbaiki data, menghapus data dan menampilkan data (Setiyadi et al., 2020). Fungsi bahasa *query* ini adalah pengguna akan bisa mendapatkan data spesifik yang dibutuhkan dan dapat melakukan pembaharuan informasi pada data-data yang terdapat di dalam basis data (Mardiono et al., 2019).

Cara berinteraksi antara pemakai dengan basis data diatur menggunakan suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat DBMS. Bahasa ini terdiri dari sejumlah perintah (*statement*) yang diformulasikan dan dapat diberikan oleh pemakai dan dikenali oleh DBMS untuk melaksanakan suatu aksi tertentu (Fatansyah, 2018).



Gambar 2.4 Lapisan Data Independence

Basis data menyediakan *Data-Definition Language* (DDL) untuk menentukan skema basis data dan *Data-Manipulation Language* (DML) untuk mengekspresikan permintaan dan pembaruan basis data (Putri, 2020).

a. Data Definition Language (DDL)

Data Definition Language (DDL) adalah struktur bahasa basis data yang menggambarkan skema basis data secara keseluruhan. Dengan menggunakan struktur bahasa DDL maka kita akan dapat membuat tabel, membuat indeks, menentukan struktur tabel, dan lain-lain. Hasil dari kompilasi perintah DDL disebut dengan Kamus Data (*Data Dictionary*) yang menjelaskan data sesungguhnya. Contoh bahasa DDL :

Untuk membuat database, perintahnya yaitu:

```
create database (nama_database);
```

Untuk membuat tabel, perintahnya yaitu:

```
create table dosen
```

```
(nip varchar(18) primary key,  
 nama varchar(25) not null,  
 alamat varchar(30) not null);
```

create merupakan perintah *Data Definition Language* (DDL).

b. Data Manipulation Language (DML)

Data Manipulation Language (DML) adalah struktur bahasa basis data yang digunakan untuk melakukan manipulasi dan pengambilan data dari suatu basis data. Struktur bahasa basis data ini memudahkan pemakai untuk mengakses data yang terdapat pada suatu basis data. Jenis-jenis akses yang bisa dilakukan dengan struktur bahasa DML ini antara lain:

- 1) Penambahan data baru ke suatu basis data (*insert*)
- 2) Penghapusan data dari suatu basis data (*delete*)
- 3) pengubahan data dari suatu basis data (*update*)

Ada 2 jenis DML, yaitu:

- 1) Prosedural, mensyaratkan pemakai menentukan data apa yang diinginkan serta bagaimana cara mendapatkannya.
- 2) Non Prosedural, membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

Contoh bahasa DML :

Query untuk memasukkan/menyimpan data pada tabel:

```
insert into [nama_tabel] (field1,  
field2) values (....,...);
```

Query untuk memperbaiki data pada tabel:

```
update [nama_tabel] SET  
nama_field=value;
```

Query untuk menghapus data pada tabel:

```
delete from [nama_tabel] where  
nama_field=value;
```

Query untuk menampilkan data dari tabel dosen:

```
select * from dosen where nip =  
'198006042014011002'
```

Daftar Pustaka

- Budio, S., Fadlan, A. H., & Sari, P. S. (2019). Manajemen Data Base. *Jurnal Menata: Jurnal Manajemen Pendidikan Islam*, 2(1), 65–76.
- Fatansyah. (2018). Basis Data (Revisi Ketiga). Penerbit INFORMATIKA.
- Mardiono, I., Fil'aini, R., & Didin, F. S. (2019). Perancangan sistem basis data offline dokumen akreditasi program studi. *Jurnal Optimasi Sistem Industri*, 12(2), 101–107. <https://doi.org/10.31315/opsi.v12i2.3153>
- Putri, R. A. (2020). Diktat Sistem Basis Data. UIN Sumatera Utara - Medan.
- Setiyadi, D., Henderi, H., & Arifin, R. W. (2020). Fungsi Date dalam Data Manipulation Language Dengan Bahasa Query Menggunakan SQL Server 2008. *INFORMATICS FOR EDUCATORS AND PROFESSIONAL: Journal of Informatics*, 4(2), 163. <https://doi.org/10.51211/itbi.v4i2.1329>

Profil Penulis



Nazaruddin Ahmad

Penulis berasal dari kota Banda Aceh, Provinsi Aceh, lahir di Banda Aceh 5 Juni 1982, merupakan anak keempat dari lima bersaudara dari pasangan Drs. H. Ahmad Habib Lubis dan Hj. Rostina. Saat ini penulis bekerja sebagai dosen tetap pada Prodi Teknologi Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Ar-Raniry Banda Aceh sejak tahun 2014. Penulis menyelesaikan pendidikan Strata Satu (S1) Teknik Informatika di Universitas Jabal Ghafur Sigli – Aceh Tahun 2008, kemudian melanjutkan pendidikan Strata Dua (S2) di Universitas Atma Jaya Yogyakarta (UAJY) pada program studi magister Teknik Informatika dan mendapatkan gelar Magister Teknik pada tahun 2013. Penulis saat ini mengajar mata kuliah Pengantar Teknologi Informasi, Basis Data, Manajemen Proyek Teknologi Informasi, Jaringan Syaraf Tiruan, Data Mining. Penulis aktif dalam menulis berbentuk *Bookchapter* yang berkolaborasi dengan dosen-dosen dari seluruh Indonesia dengan harapan dapat memberikan kontribusi untuk pengembangan ilmu dalam bidang Informatika. Penulis juga terlibat dalam pengembangan jurnal program studi teknologi informasi yaitu JINTECH: Journal of Information Technology.

Email penulis: nazar.ahmad@ar-raniry.ac.id

3

MODEL BASIS DATA RELASIONAL

Bagus Tri Mahardika, M.MSI.

Universitas Darma Persada

Model Data Relasional

Model Relasional atau Basis Data Relasional adalah sekumpulan relasi yang mempunyai nama berbeda dan ternaliasasi, atau merupakan suatu model bagaimana menunjukkan suatu cara untuk mengelola data secara fisik dalam memori sekunder dan bagaimana bentuk relasi dari keseluruhan data dalam sistem yang ditinjau.

Relasi tabel adalah hubungan logik antar data dalam basis data dengan cara memvisualisasikan ke dalam bentuk tabel dua dimensi yang terdiri dari sejumlah baris dan kolom yang menunjukkan atribut-atribut

Kelebihan

1. Dapat mengakomodasi berbagai kebutuhan pengelolaan basis data yang ada di dunia nyata (*real world*)
2. Pencarian data dari suatu tabel atau banyak tabel dapat dilakukan dengan cepat
3. Merupakan model yang paling sederhana sehingga mudah untuk dipahami.

Karakteristik Model Basisdata Relasi

1. Semua entry / elemen data pada suatu baris dan kolom tertentu harus mempunyai nilai tunggal (single value)

2. Setiap kolom berisi data tentang atribut dari suatu entity.
3. Semua entry / elemen pada suatu kolom tertentu harus mempunyai domain / jenis yang sama
4. Masing - masing kolom dalam suatu relasi mempunyai nama unik
5. Pada suatu tabel yang sama tidak ada dua baris yang identik

Istilah Model Basisdata Relasi

Di dalam penerapan model basis data relasi terdapat beberapa istilah penting dalam konsep pengorganisasian database yang selalu digunakan, yaitu:

1. ENTITAS:

Suatu obyek yang mewakilkan sesuatu dalam dunia nyata dan dapat dibedakan antara satu dengan lainnya (unique). Setiap entitas memiliki beberapa atribut yang mendeskripsikan karakteristik. Entitas dapat berupa data fisik, abstrak atau konsep, kejadian / transaksi.

2. RELASI:

Dapat diartikan sebagai data yang saling berhubungan satu dengan yang lainnya. Relasi atau hubungan adalah kejadian atau transaksi yang terjadi di antara dua entitas yang tersimpan didalam database, relasi juga bisa disebut tabel dengan baris-baris dan kolom yang menjadi penyusunnya. Elemen relasi adalah baris-baris (tupel) dalam tabel tersebut. Baris atau tupel ini serupa dengan record dalam file.

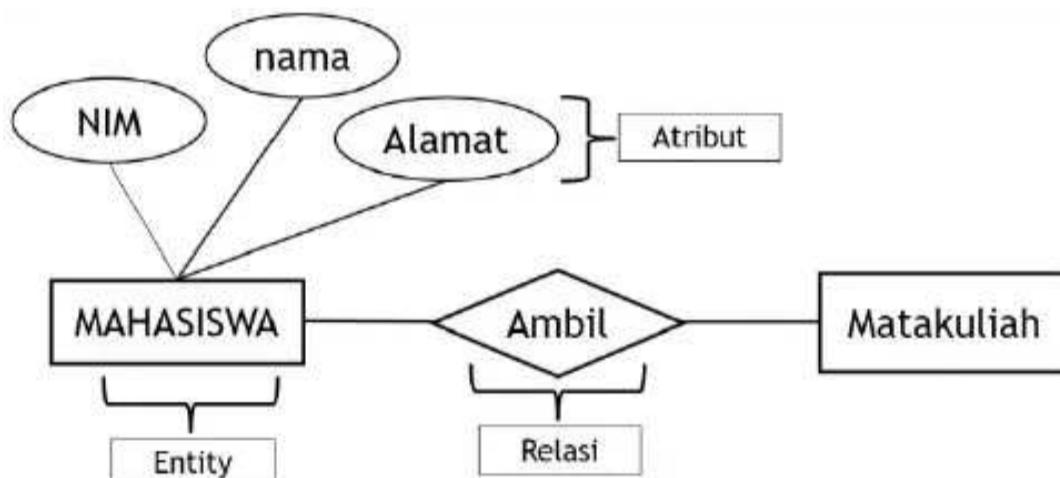
3. FIELD / KOLOM / ATRIBUT:

Merupakan kolom bernama dari suatu relasi, setiap relasi pasti memiliki beberapa atribut didalamnya untuk merepresentasikan data-data yang ada didalamnya. atau dapat dikatakan sebagai satuan data terkecil yang tidak dapat dipecah lagi menjadi unit lain yang bermakna. Atribut (Field) adalah elemen, data field, atau data item yang digunakan

untuk menerangkan suatu atribut dari entitas yang mempunyai nilai tertentu. Kumpulan dari field membentuk suatu record.

Atribut memiliki beberapa jenis, antara lain:

- a. **Atribut Value:** Data actual yang disimpan pada suatu atribut didalam suatu entity
- b. **Atribut Key:** Atribut yang digunakan untuk menentukan suatu entity secara unik & berbeda
- c. **Atribut Simple:** Atribut yang hanya memiliki nilai tunggal
- d. **Atribut Multivalue:** Atribut yang memiliki sekelompok nilai untuk setiap instant entity
- e. **Atribut Composite:** Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil dan mempunyai arti tertentu
- f. **Atribut Derivatif:** Suatu atribut yang berasal atau dihasilkan dari atribut yang lain.



4. RECORD / BARIS / TUPLE:

Merupakan baris pada sebuah relasi, bisa disebut juga sebagai record dalam file atau kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap. Satu record mewakili satu data atau informasi tentang seseorang. Record / Tuple (Tupel) merupakan baris dari suatu relasi.

5. CARDINALITY / KARDINALITAS:

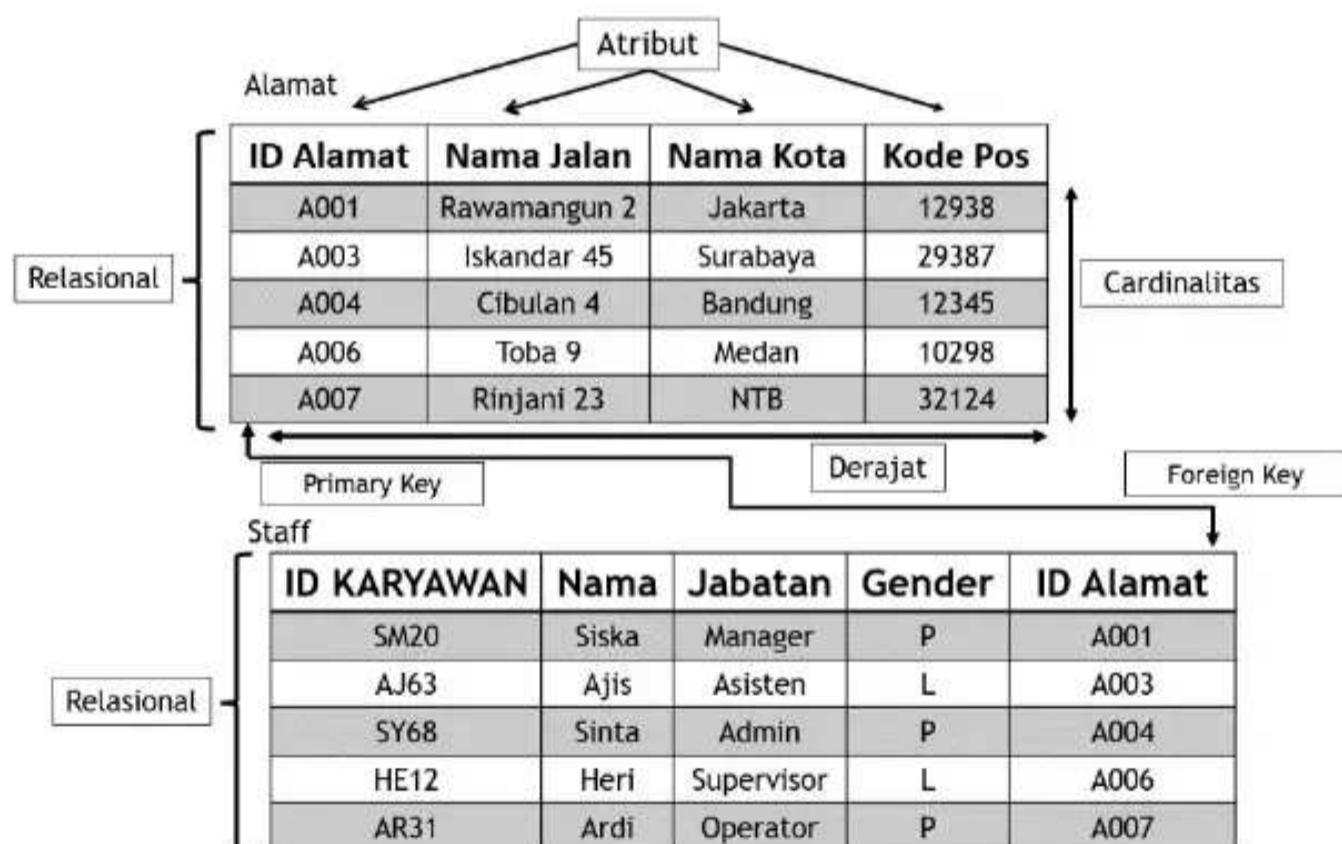
Merupakan jumlah tuple dari suatu relasi atau ukuran keunikan data ada kolom tertentu dari suatu tabel database. Pengukuran kardinalitas didasarkan pada perbandingan jumlah row yang unik pada kolom terhadap jumlah keseluruhan row. Semakin rendah nilai perbandingannya maka data semakin tidak unik, demikian juga sebaliknya.

6. DOMAIN:

Adalah himpunan yang terdapat dalam suatu atribut. Setiap atribut dalam basisdata relasional didefinisikan terhadap suatu domain. Atau dapat juga dikatakan suatu himpunan nilai yang memungkinkan untuk suatu atribut.

7. DEGREE / DERAJAT:

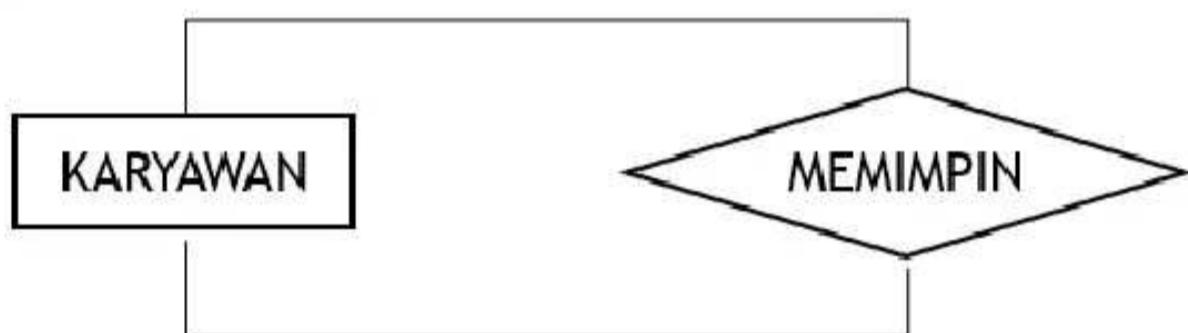
Merupakan jumlah atribut-atribut yang ada dalam suatu relasi



Domain Atribut			
Atribut	Nama Domain	Arti	Definisi
ID Alamat	Nomer Id Alamat	Kumpulan nomer untuk Id alamat	Panjang karakter 4
Nama Jalan	Nama Jalan	Kumpulan nama jalan	Panjang karakter 30
Nama Kota	Nama kota	Kumpulan nama kota	Panjang karakter 25
Kode pos	Kode pos	Kumpulan kode pos	Panjang karakter numerik 6
Jabatan	Jabatan	Jabatan pada perusahaan	Panjang karakter ; 25
Gender	Jenis Gender	Jenis kelamin seseorang	Panjang karakter 1 ; (L / P)

8. UNARY RELATION:

Adalah relasi yang terjadi dari sebuah himpunan entitas ke himpunan entitas yang sama, atau sering disebut dengan relasi tunggal.



9. BINARY RELATION:

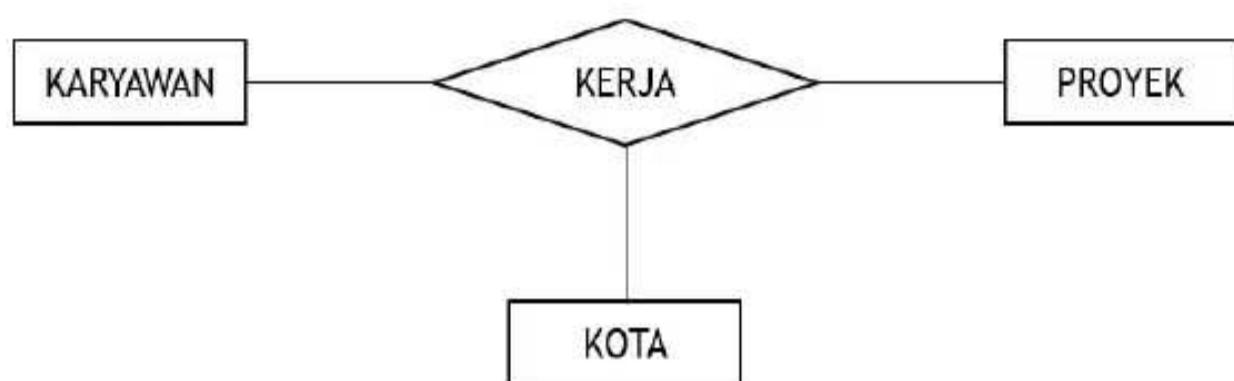
Adalah relasi yang terjadi antara 2 himpunan entitas yang berbeda. Relasi ini merupakan relasi umum yang digunakan.



10. TERNARY RELATION:

Merupakan relasi dari 3 entitas atau lebih. Relasi ini untuk menghubungkan dari 3 entitas yang dimasukkan ke relasi multi entitas. Ternary / N-ary relation menunjukkan secara lebih jelas bahwa beberapa entitas berpartisipasi dalam sebuah relasi tunggal. Bentuk relasi semacam ini

se bisa mungkin dihindari karena akan mengaburkan derajat relasi yang ada dan akan menyebabkan perencanaan database semakin kompleks



Kardinalitas (Derajat Relasi)

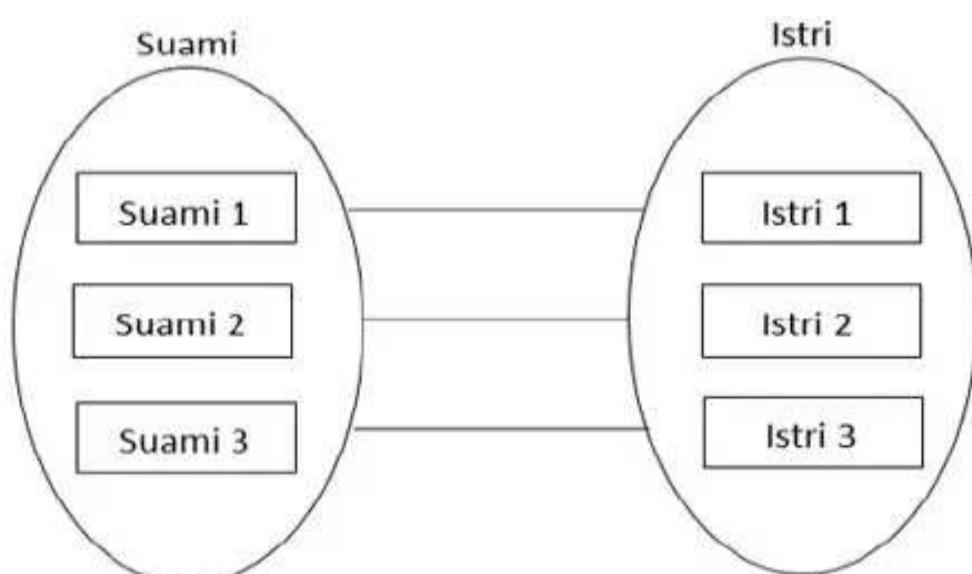
Kardinalitas relasi menunjukkan atau menggambarkan banyaknya jumlah maksimum entitas yang dapat berrelasi dengan entitas pada himpunan entitas lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*), dan banyak ke banyak (*many to many*).

1. Satu ke satu (*one to one*)

Hubungan satu ke satu (*one to one*) berarti setiap himpunan entitas hanya boleh berhubungan dengan satu himpunan entitas lainnya.

Relasi ini mempunyai notasi 1: 1.

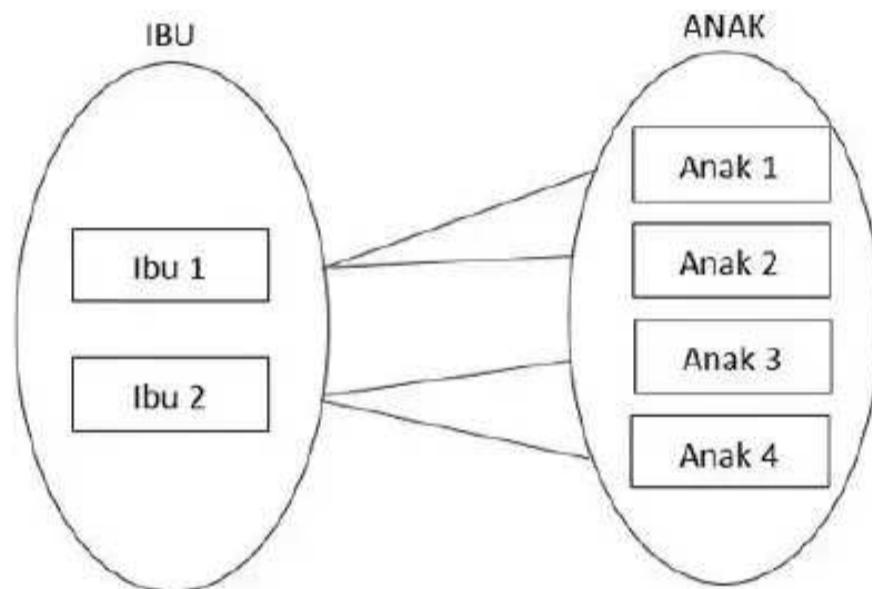
Contoh, satu himpunan entitas suami hanya berhubungan tepat dengan satu himpunan entitas istri.



2. Hubungan satu ke banyak (one to many)

Hubungan satu ke banyak (one to many) berarti satu dari setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya.

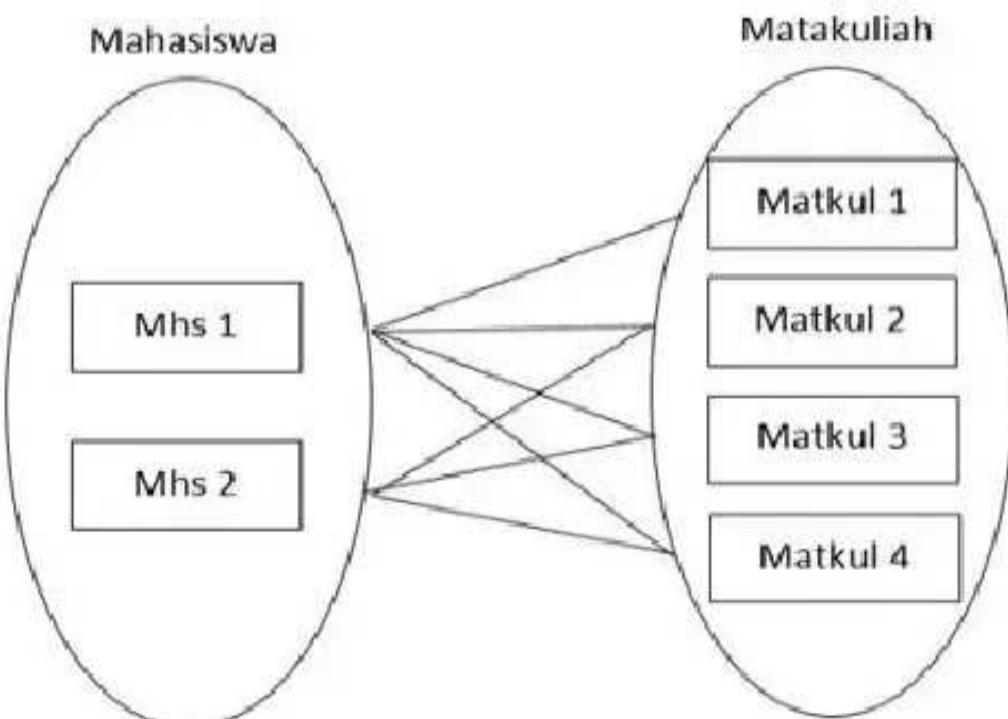
Relasi ini mempunyai notasi 1: m



Pada gambar diatas dapat dilihat bahwa satu himpunan ibu memiliki banyak hubungan ke himpunan entitas anak.

3. Hubungan banyak ke banyak (Many to many)

Hubungan banyak ke banyak (many to many) berarti setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya.



Pada gambar diatas dapat dilihat bahwa satu himpunan mahasiswa memiliki banyak hubungan ke himpunan entitas matakuliah dan satu dari

himpunan matakuliah memiliki banyak hubungan ke himpunan entitas mahasiswa.

Jenis-Jenis “Key” Dalam Relasi Database

KEY / KUNCI adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris pada data dalam suatu tabel secara unik, artinya jika suatu atribut dijadikan sebuah key maka tidak boleh ada baris data yang mempunyai nilai yang sama untuk atribut tersebut.

Fungsi atribut dapat dikatakan sebagai index atau kunci utama ketika kita akan mencari data dalam sebuah database. Key memiliki beberapa jenis antara lain:

1. SUPER KEY;

Satu atau lebih atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Bisa jadi ada lebih dari satu kumpulan atribut yang bersifat super key dalam sebuah tabel. Contoh super key: Nim, Nama, alamat.

2. CANDIDATE KEY;

Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris dalam sebuah tabel secara unik. Sebuah candidate key tidak boleh berisi atribut atau kumpulan atribut yang telah menjadi superkey yang lain, contoh: nim, nama (jika tidak ada nilai yang berulang didalamnya).

3. PRIMARY KEY:

Merupakan candidate key yang telah dipilih untuk mengidentifikasi setiap record secara unik. Primary key harus merupakan atribut yang benar-benar unik dan tidak boleh ada nilai NULL. Primary key adalah suatu nilai dalam basis data yang digunakan untuk mengidentifikasi suatu baris dalam tabel. Contoh: NIM.

4. ALTERNATE KEY;

Alternate key adalah primary key yang tidak terpilih.

Misal dalam suatu entitas terdapat dua atribut yang bisa dijadikan sebagai primary key. Sementara yang

boleh dijadikan primary key hanya satu, maka kita harus memilih salah satu. Atribut yang dipilih, disebut primary key. Sedangkan atribut yang tidak dipilih disebut dengan alternate key.

Contoh misalkan ada NIM dan No_KTP dalam sebuah tabel, maka kedua atribut tersebut dapat dijadikan sebagai primary key, namun hanya ada satu primary key dalam sebuah tabel jadi harus dipilih salah satu saja.

5. *FOREIGN KEY*:

Jika sebuah primary key terhubung ke tabel lain, maka keberadaan primary key pada tabel lain tersebut adalah sebagai foreign key.

Foreign key adalah atribut dalam suatu relasi yang digunakan untuk menunjukkan ke suatu baris pada relasi yang lain, jadi foreign key ini digunakan untuk membuat sebuah relasi yang terjadi antara tabel A dengan tabel B, dimana ketika tabel A membuat sebuah relasi dengan tabel B maka ditabel B primary key tabel A akan menjadi foreign key di tabel B.

6. *COMPOSITE KEY*:

Composite key adalah key yang terdiri dari dua atau lebih atribut yang secara unik mengidentifikasi suatu entitas.

Setiap atribut yang membentuk key senyawa adalah key sederhana. Composite key terjadi karena dalam suatu tabel tidak ditemukannya sebuah primary key, jadi supaya tidak melakukan pembuatan primary key baru maka dibuatlah sebuah composite key, yaitu biasanya merupakan gabungan dari dua buah foreign key, sehingga tidak ada data yang sama yang akan dimunculkan.

Integritas Referensial

Integritas referensial adalah seperangkat aturan yang mengatur hubungan antara kunci primer dengan kunci tamu milik tabel-tabel yang berada dalam suatu basis data relasional untuk menjaga konsistensi data. Tujuan

integritas referensial sendiri adalah untuk menjamin dan memastikan agar dalam suatu tabel yang menunjukkan ke suatu pengenal unik pada suatu baris ditabel lain benar-benar menunjuk pada nilai yang memang ada. Sehingga kejadian ambiguitas data ketika integrasi data tidak terjadi.

Berdasarkan operasi yang dilakukan, integritas referensial dapat dibedakan sebagai berikut:

1. Penambahan (*Insert*)
2. Penghapusan (*Delete*)
3. Peremajaan (*Update*)

Integritas referensial membuat ketiga operasi di atas dapat dilaksanakan pada tabel yang memiliki relasi. Sehingga proses penghapusan ataupun peremajaan suatu kolom juga akan terjadi pada kolom tabel lain yang mempunyai referensi dengannya.

Dalam Bahasa Data Definition Language SQL (DDL), kunci primer, kunci kandidat, dan kunci tamu, dapat dispesifikasikan sebagai bagian dari pernyataan SQL create table. Kunci kandidat merupakan kunci yang secara unik dapat digunakan untuk mengidentifikasi suatu baris dalam tabel. berikut salah satu contoh DDL dari pembuatan tabel mata_kuliah.

Create tabel mata_kuliah
(kode_mk char (6) not null,
nama_mata_kuliah varchar (25),
nip char (9),
primary key (kode_mk),
foreign key (nip) reference dosen on delete cascade)

Mata kuliah**Batasan Integritas**

1. Domain Atribut;

Setiap nilai yang disimpan dalam kolom sebuah relasi harus memiliki jangkauan nilai yang sama.

2. Aturan integritas;

Aturan yang menjamin setiap atribut primary key bernilai valid (tak null dan bukan null)

3. Integritas referensial;

Garis yang menghubungkan antara suatu tabel dengan tabel lain.

Relasi yang Berstruktur Baik

Relasi yang mengandung redundansi yang minimal dan mengijinkan pengguna untuk menyisipkan, memodifikasi, serta menghapus baris-baris tanpa menimbulkan kesalahan.

Basisdata yang Baik

Pembentukan basis data yang baik akan memberikan sejumlah keuntungan:

1. Tabel tabel dan relasi lebih kompak
2. Struktur masing-masing tabel lebih efisien dan sistematis
3. Kebutuhan ruang penyimpanan data lebih efisien
4. Redudansi data yang optimal akan meningkatkan integritas data

5. Tidak ada ambiguitas data disemua tabel

Integritas data (kesatuan data) merupakan keutuhan dan kesatuan data. Sehingga data dalam basis data sehingga data tersebut dapat menjadi sumber informasi yang dapat digunakan. Munculnya istilah integritas terutama disebabkan oleh adanya konsep basis data relasional dan adanya normalisasi. Pemisahan data kedalam tabel-tabel yang mempunyai relasi membuat integritas keseluruhan data menjadi sangat penting. Relasi antara satu tabel dengan tabel yang lain harus benar-benar terjadi sehingga keutuhan data dapat terjaga.

Datftar Pustaka

Aryanto. 2010. Pengolahan Database MySQL. Yogyakarta: CV. Budi Utama

Connolly, T., & Begg, C. (2010). Database Systems: A Practical Approach to Design, Implementation, and Management (5th ed.). United States: Pearson.

Fathansyah. (2015). Basis Data. Bandung: Informatika Bandung

Kadir Abdul, 2008, Belajar database menggunakan MySQL, Andi Offset, Yogyakarta.

Profil Penulis

Bagus Tri Mahardika



Penulis lahir di Jakarta, pada tanggal 26 Maret 1986. Beralamat di Jalan Taman Malaka Selatan Blok A3 no 2, Jakarta Timur. Pendidikan S1 yaitu Sarjana Komputer, Universitas Bina Nusantara dan melanjutkan S2 pada Magister Manajemen Sistem Informasi, Universitas Bina Nusantara (MMSI).

Email Penulis: bagusunsada@gmail.com

4

REALASIONAL KEY

*(SUPER KEY, CANDIDAT KEY,
PRIMARY KEY, ALTERNATIF)*

Musyrifah, S.Pd., M.Pd.

Universitas Sulawesi Barat

Relational key adalah atribut yang dijadikan sebagai key atau atribut yang dapat mengidentifikasi sebuah table. Relational Keys juga dapat mengidentifikasi satu atau sekelompok kolom yang nilainya dapat membedakan secara unik tuple-tuple tersebut.

Key, adalah satu atau gabungan beberapa atribut yang dapat membedakan semua baris data (row) dalam tabel secara unik. Artinya adalah apabila suatu field/atribut dijadikan key, maka tidak boleh ada dua atau lebih baris data dengan nilai yang sama untuk field/atribut tersebut. elemen record yang dipakai untuk menemukan record tersebut pada waktu akses, atau bisa juga digunakan untuk mengidentifikasi suatu entity atau record atau baris

Sehubungan dengan pernyataan tersebut, maka kita dapat membedakan 4 (Empat) macam key yang dapat diterapkan pada suatu tabel:

Tabel 1. Anggota

Kode Anggota	Nama
A01	Surya
A02	Putri
A03	Syahrul

Tabel 2. Pengembalian

Kode Kembali	Kode Pinjam
KM01	PJ01
KM02	PJ02

Tabel 3. Buku

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

Tabel 4. Peminjaman

Kode Pinjam	Tgl Pinjam	Kode Buku	Kode Anggota	Jumlah	Tgl Kembali
PJ01	10-01-2019	B01	A01	1	13-01-2019
PJ02	10-01-2019	B02	A01	1	13-01-2019
PJ03	10-01-2019	B03	A01	1	13-01-2019
PJ04	12-01-2019	B02	A02	1	14-01-2019
PJ05	12-01-2019	B03	A02	1	14-01-2019

Tabel 5. Mahasiswa (MHS)

NPM	Nama	Alamat
10296832	Nurhidayati	Jakarta
31296500	Budi	Bogor
41296525	Pipit	Depok
21196353	Andi	Tangerang

Tabel 6. Data Mahasiswa

PMBID	NIM	Nama	Tempat Lahir	Tanggal Lahir	Alamat
20196832	19020015	Nurhidayati	Kanang	1997-08-02	Polewali
20196500	18023994	Budi	Kanang	1999-04-06	Polewali
20196525	18011109	Pipit	Lembang	2000-10-11	Majene
20180075	18023993	Andi	Pangaliali	1993-08-25	Majene

Super Key

Super Key adalah satu atribut atau kumpulan atribut yang secara unik mengidentifikasi sebuah baris di dalam relasi atau himpunan dari satu atau lebih entitas yang dapat digunakan untuk mengidentifikasi secara unik

sebuah entitas dalam set entitas. Kunci kandidat adalah konsep yang terkait erat dengan *super key* dikurangi menjadi jumlah minimum kolom yang diperlukan untuk mengidentifikasi setiap baris secara unik. Pada tabel diatas masing-masing terdapat lebih dari satu *superkey*, yaitu:

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

Sebagai contoh, untuk Tabel Mahasiswa (MHS) di atas, super key-nya adalah:

1. NPM
2. Nama (dengan syarat tidak ada nama yang sama)
3. Alamat (dengan syarat tidak ada alamat yang sama)
4. NPM + Nama
5. NPM + Alamat
6. Nama + Alamat
7. NPM + Nama + Alamat

Contoh selanjutnya untuk Tabel 6. Data Mahasiswa di atas:

Sesuai pengertian dari super key yaitu satu atribut atau beberapa atribut yang mengidentifikasi sebuah baris jadi bisa di gambarkan seperti dibawah ini sesuai data dari Tabel 6.

Tabel 6. Data Mahasiswa

Record/Baris 2							Record/Baris 1						
PMBID	NIM	Nama	Tempat Lahir	Tanggal Lahir	Alamat								
20196832	19020015	Nurhidayati	Kanang	1997-08-02	Polewali								
20196500	18023994	Budi	Kanang	1999-04-06	Polewali								
20196525	18011109	Pipit	Lembang	2000-10-11	Majene								
20180075	18023993	Andi	Pangaliali	1993-08-25	Majene								

Data warna orange menunjukkan data satu record atau baris dari beberapa atribut atau kolom begitupun data warna kuning dan seterusnya sampai terdapat ada 4 record atau baris yang ada ditabel 6. Dari data tersebut baris tersebut dapat dikatakan super key karena tidak ada data yang sama, walaupun data tempat lahir dan alamat ada yang sama di beberapa atribut tetapi dalam satu baris ada data yang unik atau tidak sama dengan data yang lain misalkan data **Alamat** dan **Tempat Lahir** sama tetapi data

PMBID, NIM, Nama dan **Tanggal Lahir** tidak sama jadi tetap dikatakan unik atau super key dari kumpulan beberapa atribut.

Candidate Key

Candidate key adalah atribut yang menjadi determinan yang dapat dijadikan identitas baris pada sebuah relasi. Biasanya super key minimum. *Candidate key* ini berperan sebagai untuk mengidentifikasi adanya kejadian yang spesial pada tabel. Pada candidate key ini juga memiliki syarat sebuah kunci yang dinamakan candidate key adalah unik identifier, serta juga *candidate key* ini non duplikat. Yang di mana maksudnya tidak ada kunci yang memiliki ciri khas yang sama dengan *candidate key*. Pada masing-masing tabel terdapat lebih *candidate key* atau bukan *candidate key*, yaitu:

1. Tabel anggota:
 - a. Kolom kode anggota: candidate key
 - b. Kolom kode anggota dan kolom nama anggota: bukan candidate key
2. Tabel buku:
 - a. Kolom kode buku: candidate key
 - b. Kolom kombinasi kode buku, judul, stok buku: bukan candidate key
3. Tabel Peminjaman:
 - a. Kolom kode pinjam, kode buku, kode anggota: candidate key
 - b. Kolom kombinasi kode pinjam, kode buku, kode anggota, jumlah: bukan candidate key
4. Tabel pengembalian
 - a. Kolom kode kembali, kode pinjam: candidate key
 - b. Kolom kombinasi kode kembali, kode pinjam: bukan candidate key

Sebagai contoh, untuk Tabel 5. Mahasiswa (MHS) di atas, Candidate key-nya adalah:

1. NPM

2. Nama

3. Alamat

Contoh selanjutnya untuk Tabel 6. Data Mahasiswa di atas:

Sesuai pengertian dari candidate key yaitu super key yang minimal. Dari sekian banyak super key yang ada di Tabel 6 seperti **PMBID**, **NIM**, **Nama** dan **Tanggal Lahir** maka akan dipilih kunci super minimalnya seperti yang ditunjukkan di Tabel 6 dibawah ini.

Tabel 6. Data Mahasiswa

PMBID	NIM	Nama	Tempat Lahir	Tanggal Lahir	Alamat
20196832	19020015	Nurhidayati	Kanang	1997-08-02	Polewali
20196500	18023994	Budi	Kanang	1999-04-06	Polewali
20196525	18011109	Pipit	Lembang	2000-10-11	Majene
20180075	18023993	Andi	Pangaliali	1993-08-25	Majene

Beberapa *super key* yang telah di warnai di atas dari beberapa atribut maka akan dipilih *candidate key* yang kemungkinan datanya tidak akan sama. Untuk **Nama** dan **Tanggal Lahir** bisa saja beberapa orang memiliki **Nama** atau **Tanggal Lahir** yang sama tetapi **Nim** dan **PMBID** hanya dimiliki satu mahasiswa jadi super key minimalnya atau *candidate key* adalah **PMBID** dan **NIM**.

Primary Key

Primary key adalah candidate key yang dipilih untuk mengidentifikasi baris data secara unik dalam relasi. *Primary Key* atau Kunci Utama adalah atribut merupakan *candidate key* yang telah dipilih untuk mengidentifikasi setiap record secara unik. Jika dalam suatu tabel hanya terdapat satu *candidate key* misalnya tabel anggota dan

tabel buku), maka key tersebut menjadi *primary key*. Tetapi jika terdapat lebih dari satu *candidate key* (misal tabel peminjaman dan tabel pengembalian), maka salah satu *candidate key* tersebut dapat dijadikan *primary key*. *Primary key* masing-masing tabel diatas adalah:

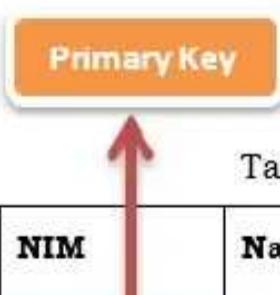
1. Tabel anggota : kode anggota
2. Tabel buku : kode buku
3. Tabel peminjaman : kode pinjam
4. Tabel pegembalian : kode kembali

Candidate key yang dipilih untuk mengidentifikasi tuple secara unik dalam relasi. Maka, *primary key* yang dipilih pada Tabel 5. Mahasiswa (MHS) di atas adalah NPM (unik, tidak ada NPM yang sama).

Contoh selanjutnya untuk Tabel 6. Data Mahasiswa di atas:

Sesuai pengertian dari primary key yaitu candidate key yang telah dipilih untuk mengidentifikasi setiap record yang unik. Maka dari candidate key yang dipilih di Tabel 6 yaitu atribut **PMBID** dan **NIM** dipilih salah satunya yang akan menjadi *primary key* karena sifat *primary key* adalah tidak ada nilai null, harus unik dan sifatnya minimal. Sifat minimalnya ini dari mana yaitu dari candidate key yang dipilih salah satunya. Jadi untuk *primary key* adalah **NIM**.

Primary Key



Tabel 6. Data Mahasiswa

PMBID	NIM	Nama	Tempat Lahir	Tanggal Lahir	Alamat
20196832	19020015	Nurhidayati	Kanang	1997-08-02	Polewali
20196500	18023994	Budi	Kanang	1999-04-06	Polewali
20196525	18011109	Pipit	Lembang	2000-10-11	Majene
20180075	18023993	Andi	Pangaliali	1993-08-25	Majene

Alternative Key

Alternative Key adalah *candidate key* yang tidak terpilih sebagai *primary key* atau atribut untuk menggantikan kunci utama. *Alternate Key* adalah *primary key* yang tidak terpilih. Misal dalam suatu entitas terdapat dua atribut yang bisa dijadikan sebagai *primary key*. Pada tabel pengembalian jika memilih kode kembali sebagai *primary key*, maka kode pinjam dapat dijadikan *alternate key*.

Candidate key yang tidak dipilih sebagai *primary key*. Maka, *candidate key* yang dipilih pada Tabel 5. Mahasiswa (MHS) di atas adalah Nama dan Alamat.

Contoh selanjutnya untuk Tabel 6. Data Mahasiswa di atas:

Sesuai pengertian dari *Alternative key* yaitu *candidate key* yang tidak terpilih sebagai kunci utama atau *primary key*. Jadi jika yang dipilih menjadi primary key adalah atribut **NIM** maka yang tidak dipilih dan otomatis menjadi alternative key yaitu atribut **PMBID**.

Alternative Key



Tabel 6. Data Mahasiswa

PMBID	NIM	Nama	Tempat Lahir	Tanggal Lahir	Alamat
20196832	19020015	Nurhidayati	Kanang	1997-08-02	Polewali
20196500	18023994	Budi	Kanang	1999-04-06	Polewali
20196525	18011109	Pipit	Lembang	2000-10-11	Majene
20180075	18023993	Andi	Pangaliali	1993-08-25	Majene

Contoh Penggunaan Relasi

Tabel Mahasiswa

- Primery-key:nim
- Foreign-key: tidak ada

Kode SQL:

```
ALTER TABLE namataber ADD PRIMARY KEY ("namafield");
```

Tabel Matakuliah:

1. Primary-key: nim
2. Foreign-key: tidak ada

Tabel Nilai:

1. Foreign-key: nim
2. Foreign-key: kode_mk

Kode SQL:

```
ALTER TABLE namataber ADD UNIQUE Cnamafield");
```

Catatan: Sebuah tabel dapat terdiri dari beberapa field yang UNIQUE. Contoh: Basis data sistem akademik yang terdiri dari 3 tabel yaitu: Mahasiswa, Matakuliah dan Transaksi Nilai. Seperti yang ditunjukkan oleh Gambar 1 dibawah ini.

Tb_mhs

Field Name	Type Data	Len
Nim *	Char	16
Nama	Char	50
Alamat	Varchar	255
JK	Char	20

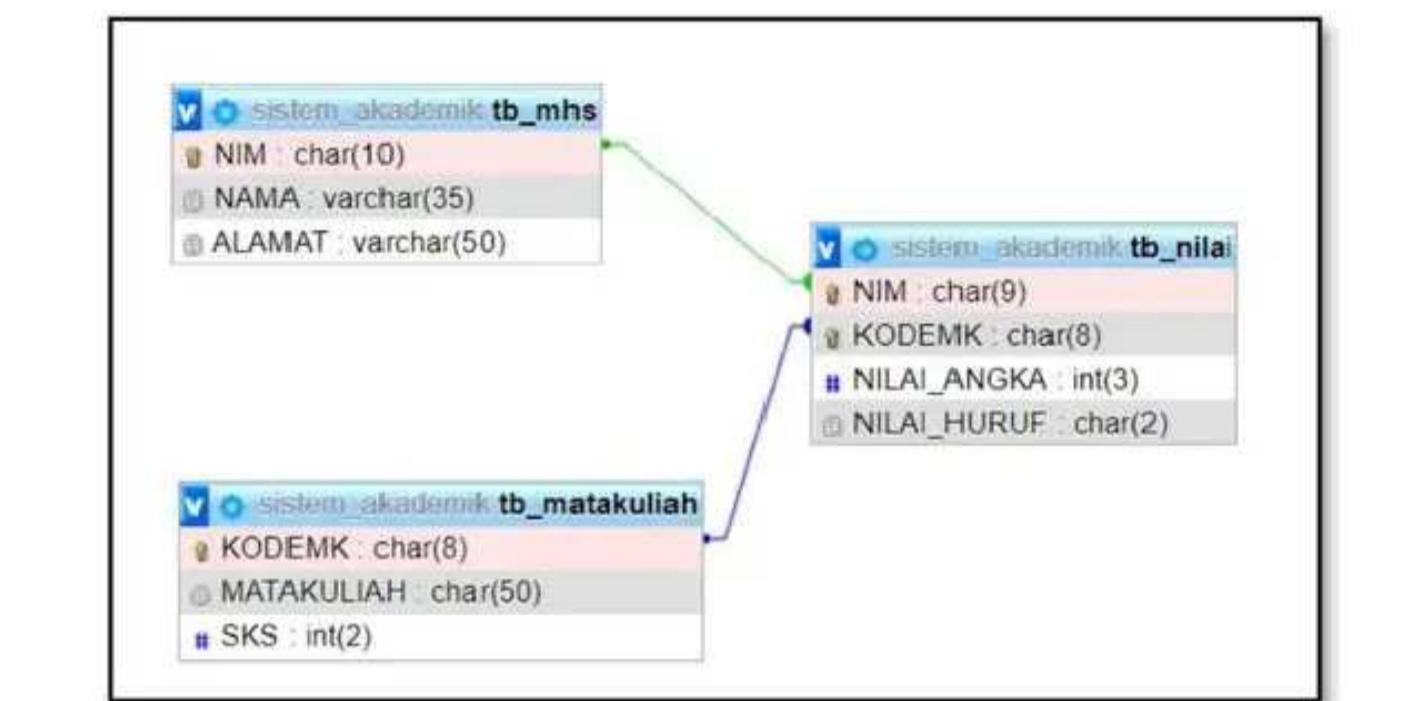
Tb_mk

Field Name	Type Data	Len
Kode_MK *	Char	10
Nama_MK	Char	100
Jumlah_SKS	Int	10

Tb_Nilai

Field Name	Type Data	Len
Nim **	Char	16
Kode_MK **	Char	10
Nilai_angka	Int	10
Nilai_huruf	Char	50

Gambar 1.
Field Tabel Mahasiswa, Matakuliah dan Transaksi Nilai

Hasil:

Gambar 2. Contoh Relasi Tabel Data Base

Daftar Pustaka

Penulisan daftar pustaka menggunakan format APA Edisi-7 atau 6. Contoh:

Efitra. (2022). Pengantar Praktikum Basis Data. Jambi: Guide.

Yanto, Robi. (2016). Managemen Basis Data Menggunakan MySQL. Yogyakarta: Deepublish.

Relational Key dan Functional Dependency. (2018). Laboratory of Enterprise Application. <http://lea.si.fti.unand.ac.id/>

Fransisca. (2018). The Relational Model. <https://sis.binus.ac.id/2018/03/21/the-relational-model/>

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

Pratama, Wahyu. (2015). Basis Data 1. http://wahyu_pratama.staff.gunadarma.ac.id

Profil Penulis



Musyrifah

Lahir 14 November 1993 di Kanang, Polewali Mandar, Sulawesi Barat. Memiliki ketertarikan yang tinggi di bidang pendidikan, terutama pendidikan S-1 di Universitas Negeri Makassar jurusan Pendidikan Teknik Informatika. Kemudian melanjutkan pendidikan S-2 di kampus yang sama dengan jurusan Pendidikan Teknologi Kejuruan sambil menjadi tenaga pengajar di SMK YAPPMI Makassar.

Tahun 2017, penulis menjadi pemateri seminar nasional dengan judul penelitian Pengembangan Model Pembelajaran Merakit Komputer berbasis Teaching Factory di SMK. Setelah lulus S-2 pada tahun 2017, penulis kemudian aktif menjadi tenaga pengajar di SMKN 1 Majene. Kemudian tahun 2018 diterima sebagai dosen tetap di Universitas Sulawesi Barat pada Prodi Teknik Informatika. Selama menjadi dosen, aktif menjadi anggota tim MBKM pada kegiatan KMMI Pemasaran Digital dan Program Kedaireka Pengembangan Aplikasi Smart Tourism berjasama dengan YOY. Selama berkarir sebagai dosen, telah menerbitkan beberapa penelitian.

Penulis dapat dihubungi melalui email:

musyrifah65@gmail.com

5

BAHASA PADA MODEL RELASIONAL

Husna Gemasih, S.Inf., M.Cs.

Prodi Teknik Informatika Fakultas Teknik

Universitas Gajah Putih

Pendahuluan

Bahasa yang digunakan dalam model relasional disebut bahasa query. Bahasa query adalah bahasa yang memberikan pengguna akses ke informasi dalam database. Secara umum, level bahasa ini lebih tinggi daripada bahasa pemrograman standar. Bahasa pada model relasional terbagi menjadi Bahasa query formal dan Bahasa query komersial.

Bahasa query formal adalah bahasa query yang diterjemahkan dengan menggunakan symbol-simbol matematika. Bahasa ini dibagi menjadi prosedural dan non prosedural. Prosedural adalah pengguna menentukan data yang diperlukan dan cara mengambilnya, contoh; aljabar relasional yaitu dimana query diwakili dengan menerapkan operator tertentu ke table/ relasi.

Non prosedural adalah pengguna menentukan data yang diperlukan tanpa menentukan cara mengambil data, contoh; kalkulus relasional yaitu dimana query menjelaskan kumpulan tupel yang diinginkan dengan menentukan predikat tupel yang diharapkan. Kalkulus relasional dibagi menjadi kalkulus relasional tupel dan kalkulus relasional domain.

Bahasa query komersial adalah sebuah bahasa query yang dirancang oleh programmer sendiri untuk program aplikasi agar lebih mudah digunakan oleh pengguna (*user-friendly*). Contoh QUEL, QBE dan SQL.

Aljabar Relasional

Aljabar relasional adalah bahasa query prosedural yang berisi kumpulan operasi pada relasi, dimana setiap operasi menggunakan satu atau lebih relasi untuk membuat relasi baru dari hasil operasi tersebut. Aljabar relasional menyediakan seperangkat operator untuk memanipulasi data; selection, projection, union, set-difference, cartesian-produk, rename dan menyediakan operator tambahan; set intersection, natural join, theta join, division.

1. Selection (σ)

Operasi selection digunakan untuk memilih record atau untuk mencari record yang memenuhi predikat atau kondisi tertentu. Hasilnya kemudian ditulis ke relasi baru sebagai hasil dari operasi select. Operator perbandingan yang digunakan pada operasi select adalah; $=$, \neq , $<$, \leq , $>$, \geq . Beberapa predikat dapat digabungkan dengan konektor AND (\wedge) dan OR (\vee) atau negasi (\sim) untuk membentuk predikat majemuk.

Sintaks: $\sigma_P^{(T_1)}$

Kumpulan semua record/ baris dalam T_1 yang memenuhi kondisi P dimana; P adalah predikat pada atribut T_1 , T_1 adalah table atau relasi.

Contoh selection (σ)

nim	nama	tempat_lahir	tanggal_lahir	alamat	jurusan
1902001	Ani Alatas	Takengon	12/02/2002	Bebesen	Teknik Informatika
1902002	Tina Susilo	Aceh Tengah	30/08/2002	Bebesen	Teknik Informatika
1903001	Budi Sudrajat	Jakarta	24/03/2002	Kebaya-kan	Ekonomi
1904002	Tono	Pasar Pagi	11/07/2002	Paya Tumpi	Agroteknologi
1903002	Sisi Hasanah	Aceh Tengah	26/03/2002	Kebaya-kan	Ekonomi

Kasus 1: tampilkan daftar mahasiswa yang mempunyai alamat di 'Kebayakan'.

Aljabar relasional: $\sigma_{\text{alamat}='Kebayakan'}(\text{mahasiswa})$

Hasil:

nim	nama	tempat_lahir	tanggal_lahir	alamat	jurusan
1903001	Budi Sudrajat	Jakarta	24/03/2002	Kebayakan	Ekonomi
1903002	Sisi Hasanah	Aceh Tengah	26/03/2002	Kebayakan	Ekonomi

Kasus 2: tampilkan daftar mahasiswa yang mempunyai alamat di 'Bebesen' dan tempat lahirnya di 'Aceh Tengah'.

Aljabar relasional: $\sigma_{\text{alamat}='Bebesen'} \wedge \sigma_{\text{tempat_lahir}='Aceh Tengah}(\text{mahasiswa})$

Hasil:

nim	nama	tempat_lahir	tanggal_lahir	alamat	jurusan
1902001	Ani Alatas	Takengon	12/02/2002	Bebesen	Teknik Informatika

2. Projection (π)

Operasi proyeksi adalah operasi menampilkan kolom tertentu dan merupakan operasi unary yang mengirimkan hubungan argumen dengan kolom tertentu.

Sintaks: $\pi^{(T1)}_S$

Dimana: S adalah daftar (list) yang berisi satu atau lebih field yg ada di T1, T1 adalah tabel atau relasi

Kasus 1: tampilkan nim, nama, alamat dari relasi mahasiswa.

Aljabar relasional: $\pi \text{ nim, nama, alamat}^{(\text{mahasiswa})}$

Hasil:

nim	nama	alamat
1902001	Ani Alatas	Bebesen
1902002	Tina Susilo	Bebesen
1903001	Budi Sudrajat	Kebayakan
1904002	Tono	Paya Tumpi
1903002	Sisi Hasanah	Kebayakan

Kasus 2: tampilkan nim, nama, alamat, jurusan dimana jurusannya 'Teknik Informatika' dari relasi mahasiswa.

Aljabar relasional: $\pi \text{ nim, nama, alamat, jurusan} (\text{ojurusan}=\text{'Teknik Informatika'})^{(\text{mahasiswa})}$

Hasil:

nim	nama	alamat	jurusan
1902001	Ani Alatas	Bebesen	Teknik Informatika
1902002	Tina Susilo	Bebesen	Teknik Informatika

3. Union (U)

Operasi union adalah operasi yang membuat gabungan tabel, ketika kedua tabel memiliki atribut yang sama. Operasi ini memungkinkan untuk menggabungkan data dari dua baris yang serupa.

Sintaks: $T1 \cup T2$

Dimana: $T1$ dan $T2$ adalah tabel atau relasi.

Kasus 1:

$T1$

A	B
a	b
c	d
e	f

$T2$

A	B
g	h
a	b
e	f

Aljabar relasional: $T1 \cup T2$

Hasil:

A	B
a	b
c	d
e	f
g	h

Kasus 2: Tampilkan nim (dari tabel mahasiswa) union dari nim (dari tabel perkuliahan)

Table mahasiswa

nim	nama
1902001	Ani Alatas
1902002	Tina Susilo
1903001	Budi Sudrajat
1904002	Tono
1903002	Sisi Hasanah

Tabel perkuliahan

nim	nama
1902001	Ani Alatas
1903001	Budi Sudrajat
1903002	Afika Harahap
1903003	Susi Susanti
1904002	Tono

Aljabar relasional: $\pi_{\text{nim}}(\text{mahasiswa}) \cup \pi_{\text{nim}}(\text{perkuliahhan})$

Hasil:

nim
1902001
1902002
1903001
1904002
1903002
1903002
1903003

4. Set-difference (-)

Set-difference adalah operasi yang mendapatkan record yang ada di satu tabel tetapi tidak di tabel lain.

Sintaks: $T_1 - T_2$

Dimana: T_1 dan T_2 adalah tabel atau relasi.

Kasus 1:

T_1

A	B
a	b
c	d
e	f

T2

A	B
g	h
a	b
e	f

Aljabar relasional: T1 - T2

Hasil:

A	B
c	d

Kasus 2: Tampilkan nim (dari tabel mahasiswa) set difference dari nim (dari tabel perkuliahan)

Table mahasiswa

nim	nama
1902001	Ani Alatas
1902002	Tina Susilo
1903001	Budi Sudrajat
1904002	Tono
1903002	Sisi Hasanah

Tabel perkuliahan

nim	nama
1902001	Ani Alatas
1903001	Budi Sudrajat
1903002	Afika Harahap
1903003	Susi Susanti
1904002	Tono

Aljabar relasional: π nim (mahasiswa) - nim (perkuliahan)

Hasil:

nim
1902002
1903002

5. Cartesian-product (X, disebut juga cross product)

Operasi cartesian product adalah operasi yang digunakan untuk merelasikan semua record yang berasal dari dua tabel. Operasi cartesian product bisa digabungkan dengan operasi lainnya seperti select dan project.

Sintaks: T1 X T2

Tabel dosen

nidn	nama_dosen
1001	Hendri
1002	Ira
1003	Richa

Tabel matakuliah

kd_mk	nama_mk	skls
mk01	AI	3
mk02	IMK	3
mk03	Statistik	2

Tabel mengajar

nidn	kd_mk	ruang
1001	mk01	A
1001	mk02	A
1002	mk01	A
1002	mk02	B
1003	mk01	B
1003	mk03	B

Kasus 1: tampilkan kd_mk, nama_mk, sks (dari tabel matakuliah), kelas (dari tabel mengajar) dimana kelas yang diajar adalah kelas B.

Aljabar relasional: $\pi_{\text{kd_mk}, \text{nama_mk}, \text{skls}, \text{ruang}} (\sigma_{\text{ruang}=\text{B}} \wedge \text{mengajar. kd_mk} = \text{matakuliah. kd_mk})$ (mengajar x matakuliah))

Hasil:

kd_mk	nama_mk	skls	ruang
mk01	AI	3	B
mk03	Statistik	2	B

6. Rename (ρ)

Operasi rename adalah memberi nama baru T1 dengan X, sehingga seolah-olah dimiliki 2 relasi (E1 dan X) yang isinya sama persis. Operasi rename digunakan untuk menyalin table lama kedalam table baru.

Sintaks: $\rho_x(T1)$

Kasus: salinlah table baru dengan nama matakuliah_baru dari table matakuliah, dimana sksnya adalah '3'.

Aljabar relasional: $\rho_{\text{matakuliah_baru} (\sigma \text{skls}='3') (\text{matakuliah})}$

Hasil:

kd_mk	nama_mk	skls
mk01	AI	3
mk02	IMK	3

7. Set Intersection

Set intersection merupakan operator tambahan, karena dapat disimpulkan dari operator dasar berikut:

$$A \cap B = A - (A - B), \text{ atau } A \cap B = B - (B - A)$$

Operasi set intersection digunakan untuk mengambil nilai yang ada pada satu tabel relasional dan juga ada pada tabel relasional lainnya. Simbol \cap digunakan untuk menunjukkan operasi persimpangan yang telah ditetapkan.

Kasus: Tampilkan nidn (dari tabel dosen) Set Intersection dengan nidn (dari tabel mengajar).

Aljabar relasional: $\pi_{\text{nidn}}(\text{dosen}) \cap \pi_{\text{nidn}}(\text{mengajar})$

Hasil:

nidn
1001
1002
1003

8. Natural Join

Natural Join berfungsi untuk menggabungkan operasi selection dan cartesian product menjadi 1 operasi. Ikon " \bowtie " digunakan untuk menampilkan aktivitas natural join. Natural join hanya membuat tuple

dengan nilai yang sama pada 2 atribut dengan nama yang sama dalam dua tabel relasi yang berbeda.

Kasus: Tampilkan seluruh data yang ada pada tabel Matakuliah dan tabel Mengajar.

Aljabar relasional: matakuliah \bowtie
mengajar.kd_mk=matakuliah.kd_mk mengajar

Hasil:

kd_mk	Nama_mk	SkS	nidn	ruang
mk01	AI	3	1001	A
mk02	IMK	3	1001	A
mk01	AI	3	1002	A
mk02	IMK	3	1002	B
mk01	AI	3	1003	B
mk03	Statistik	2	1003	B

Kalkulus Relasional

1. Kalkulus Relasional Tupel

Kalkulus relasional tupel adalah operasi yang menggambarkan informasi yang diinginkan tanpa memberikan prosedur rinci untuk memperoleh informasi tersebut. Kalkulus tupel relasional adalah dasar dari bahasa query QUEL.

Sintaks: $\{t \mid P(t)\}$

Semua tupel dari t sedemikian rupa sehingga predikat P benar untuk t. Mengikuti notasi sebelumnya, kita menggunakan $t[A]$ untuk merepresentasikan nilai tuple t pada atribut A dan $t \in r$ untuk menunjukkan bahwa tuple t berada pada relasi r.

Dalam kalkulus relasional dengan 2 simbol signifikan;

- a. “terdapat beberapa (there exists)”

Sintaks: $\exists t \in r (Q(t))$

yaitu terdapat tupel dari t anggota relasi r sedemikian rupa sehingga predikat $Q(t)$ benar.

- b. “untuk seluruh (for all)”

Sintaks: $\forall t \in r (Q(t))$

yaitu untuk semua anggota tupel t dari relasi r sedemikian rupa sehingga predikat $Q(t)$ benar.

Kasus: tampilkan nidn, nama dan gaji pokok pada relasi $Dosen$

yang mempunyai atribut jenis kelamin ‘Pria’ dan memiliki atribut gaji pokok adalah lebih besar dari 1000000.

Kalkulus relasional tupel: `SELECT dosen.nidn, dosen.nama, dosen.gaji_pokok FROM Dosen WHERE Dosen.jkelamin='Pria' AND Dosen.gajipokok>1000000`

2. Kalkulus Relasional Domain

Kalkulus Relasional Domain menggunakan variabel domain yang mendapatkan nilai dari domain atribut, bukan nilai dari semua tupel. Kalkulus relasional domain adalah dasar dari bahasa query QBE.

Sintaks:

$$\{ < x^1, x^2, \dots, x^n > \mid P(x^1, x^2, \dots, x^n) \}$$

dimana x_1, x_2, \dots, x_n merepresentasikan variabel domain. P adalah formula yang terdiri dari atom. Sebuah atom dalam kalkulus relasi domain dapat memiliki salah satu bentuk berikut:

- a. $< x_1, x_2, \dots, x_n > \in r$, dimana r adalah relasi dengan n atribut dan x_1, x_2, \dots, x_n adalah variabel domain atau konstanta domain.
- b. $x \theta y$, dimana x dan y adalah variabel domain dan θ adalah operator relasional ($\leq, <, =, \neq, >, \geq$). Kita membutuhkan atribut x dan y mempunyai domain yang bisa diperbandingkan dengan θ .

- c. $x \Theta c$, dimana x adalah variabel domain, Θ adalah operator relasional, dan c adalah sebuah konstanta.

Dalam kalkulus relasional domain, serta dalam kalkulus relasi tupel, kita juga menggunakan notasi “terdapat” beberapa (there exists)” dilambangkan dengan “ \exists ” dan “untuk seluruh (for all)” dilambangkan dengan “ \forall ”.

Kasus: Tampilkan nidn, nama, gaji_pokok dari dosen, dimana jenis Kelaminnya adalah pria dan gaji pokoknya lebih besar 1200000.

Kalkulus relasional domai: { nidn | \exists nama_d | \exists gajipokok (Dosen (nidn, nama, gaji_pokok) AND jkelamin='Pria' AND gajipokok > 1000000) }

Daftar Pustaka

Silberschatz, A & Gehrke, J. (2003). Sistem Manajemen Database. Yogyakarta: Penerbit Andi.

Kadir, A. (2009). Dasar Perancangan dan Implementasi Database Relasional. Yogyakarta: Penerbit Andi.

Rizarulham. (2009). Aljabar Relasional, Bahasa pada Model Data Relasional. <https://rizarulham.wordpress.com/2009/10/16/aljabar-relasional-bahasa-pada-model-data-relasional/>

Profil Penulis



Husna Gemasih

Dilahirkan tanggal 15 Februari 1990 di Dedalu Takengon Kabupaten Aceh Tengah Provinsi Aceh, anak kedua dari tiga bersaudara. Lulus S1 di Program Studi Teknik Informatika Fakultas Teknik Universitas Gajah Putih tahun 2012, lulus S2 di Program Studi Ilmu Komputer dan elektronika Fakultas MIPA Universitas Gadjah Mada tahun 2016. Penulis bergabung sebagai dosen tetap di Program Studi Teknik Informatika Universitas Gajah Putih sampai dengan saat ini.

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2008 silam. Penulis memiliki kepakaran dibidang Dana Mining. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Penulis aktif meneliti dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini.

Email Penulis: gemasihhusna@gmail.com

6

PENGENALAN SQL

BESERTA CONTOHNYA

Asmawati S, S.Kom., M.Pd.

Universitas Sulawesi Barat

Pendahuluan

Bagi yang baru pertama mendengar SQL pastinya akan bingung dengan kata “MySQL” dan “SQL”. Mungkin Anda akan berpikir dan bertanya, ini satu atau dua aplikasi yang berbeda? Untuk menjawab itu kita akan paparkan lebih jelasnya.

Untuk Anda yang telah mempelajari atau membaca tentang Basis Data, mungkin telah mendapatkan penjelasannya, tetapi disini saya akan menjelaskannya kembali. *Structured Query Language* (SQL) merupakan komponen bahasa *relational database system*. SQL merupakan bahasa baku (ANSI/SQL), *non procedural*, dan berorientasi himpunan (*set-oriented language*). SQL dapat digunakan baik secara interaktif atau ditempelkan (*embedded*) pada sebuah program aplikasi. SQL dapat digunakan untuk mendefinisikan struktur data, memodifikasi data pada basis data, menspesifikasi batasan kemanan (security), hingga pemeliharaan kinerja basis data.

So, SQL adalah bahasa permintaan yang melekat pada satu database atau SMDB tertentu, sedangkan MySQL merupakan database servernya. Sebagai sutu Bahasa permintaan, SQL tidak hanya melekat pada MySQL server saja, tetapi juga diakui oleh SMDB lainnya, seperti

MsQL, PostgreSQL, Interbase, dan Oracle. Selain itu, SQL juga didukung oleh database bukan server seperti Ms. Access maupun Paradox.

Secara umum, bahasa SQL memiliki beberapa bagian penting, yaitu:

1. *Data Definition Language* (DDL)

DDL menyediakan perintah-perintah untuk mendefinisikan skema relasi, menghapus relasi, serta memodifikasi skema relasi.

2. *Data Manipulation Language* (DML)

DML mencakup bahasa SQL untuk menyisipkan rekaman pada relasi, menghapus rekaman pada relasi, serta memodifikasi rekaman pada relasi.

3. *Data Control Language* (DCL)

DCL adalah sub bahasa SQL yang berfungsi untuk melakukan pengontrolan data dan server databasenya, seperti manipulasi user dan hak akses (privileges). Yang termasuk perintah dalam DCL ada dua, yaitu GRANT dan REVOKE.

Sejarah SQL

1. Tahun 1974, D. Chamberlin (IBM San Jose Laboratory) mendefinisikan bahasa yang disebut *Structured English Query Language* (SEQUEL)
2. Pada tahun 1975 lahirlah SEQUEL/2 versi revisi dari SEQUEL, tetapi karena alasannya hukum Namanya diubah menjadi SQL.
3. IBM secara berurut memproduksi *prototype* DBMS yang disebut *system R*, berdasarkan SEQUEL/2. Inti dari SQL adalah SQUARE (*Specifying Queries as Relational Expression*) yang mendahului proyek system R.
4. Dan pada akhir tahun 70-an, oracle muncul dan mungkin merupakan RDBMS komersil pertama yang berbasis SQL.

5. Tahun 1989, ANSI dan ISO mempublikasi standar awal untuk SQL dan pada tahun yang sama ISO juga mempublikasikan tambahan yang mendefenisikan *Integrity Enhancement Feature*.
6. Tahun 1992, revisi penting pada standar ISO yang pertama dikenal sebagai SQL2 atau SQL/92. Selanjutnya pada tahun 1999, SQL3 dikeluarkan dengan dukungan untuk manajemen data berorientasi objek.

Apa Pentingnya SQL

1. SQL telah menjadi bagian dari arsitektur aplikasi, contohnya pada arsitektur aplikasi system IBM
2. Merupakan pilihan yang strategis untuk organisasi besar dan berpengaruh
3. ~~SQL mempengaruhi pembuatan standar lainnya sebagai definitional tool. Contohnya adalah:~~
 - a. *Standard ISO Information Resource Directory System (IRDS)*
 - b. *Standard Remote Data Access (RDA)*

Penulisan Perintah SQL

Penulisan structure SQL terdiri dari *Reserved word* dan *user defined word*.

1. *Reserved word* adalah bagian yang telah ditetapkan pada SQL. Penulisannya harus sesuai dan tidak bisa di pisahkan
2. *User defined word* dibuat oleh *user* dan merepresentasikan nama-nama dari berbagai objek basis data, seperti relasi kolom, dan baris.
3. Mayoritas perintah SQL bersifat case insensitive, kecuali untuk data literal karakter
4. Mudah dibaca dengan pengaturan baris dan spasi
5. Menggunakan bentuk notasi *Backus Naur Form (BNF)*

Data Definition Language (DDL)

Data Definition Language (DDL) merupakan perintah SQL yang digunakan untuk melakukan definisi awal suatu basis data dan tabel pada konsep RDBMS. Secara sederhana, penulisan perintah SQL pada kelompok ini terdiri dari *Create*, *Alter*, dan *Drop*. Berikut merupakan pembahasan untuk perintah SQL tersebut.

1. *Create*

Perintah SQL ini digunakan untuk membuat suatu basis data dan tabel pendukung didalam pembangunan basis data tersebut. Tabel merupakan objek yang sangat penting dalam suatu basis data.

Sintak untuk membuat database baru:

```
create database databasename;
```

Sintak untuk membuat Tabel baru:

```
create tabel table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .... );
```

Contoh:

- Membuat database baru:

```
create database testDB;
```

- Membuat Tabel baru:

```
create tabel siswa (
    nis NOT NULL,
    nama varchar(255) NOT NULL,
    alamat varchar(255),
    umur int, PRIMARY KEY (nis)
) ;
```

2. *Alter*

Perintah SQL ini digunakan untuk mengubah struktur table yang terdapat didalam basis data. Hal ini dapat saja terjadi, jika ingin melakukan penambahan atau penghapusan suatu field atau atribut key (*Primary/Foreign*), tanpa mendefinisikan struktur baru pada table. Melalui perintah SQL ini jika sudah terisikan record pada table tersebut, maka mengenai hilangnya record yang terdapat pada suatu table tidak akan terjadi. Berikut ini merupakan aturan umum penulisan yang digunakan pada *alter table*.

Sintak menambah kolom baru:

```
alter table table_name ADD column_name  
datatype;
```

Sintak menghapus kolom:

```
alter table table_name drop column  
column_name;
```

Sintak merubah kolom:

```
alter table table_name alter column  
column_name datatype;
```

Contoh

Menambahkan kolom alamat pada table mahasiswa dengan tipe data varchar panjang field 25

```
alter table mahasiswa add alamat  
varchar(25);
```

Menghapus kolom alamat pada table mahasiswa:

```
alter table mahasiswa drop column alamat;
```

3. *Drop*

Perintah SQL ini digunakan untuk menghapus table yang terdapat didalam basis data. Berikut ini merupakan aturan umum penulisan yang digunakan pada *drop tabel*.

Sintak menghapus database:

```
DROP DATABASE databasename;
```

Sintak menghapus table:

```
DROP TABLE table_name;
```

Contoh

Menghapus database testDB:

```
DROP DATABASE testDB;
```

Menghapus table mahasiswa:

```
DROP TABLE mahasiswa;
```

Data Manipulation Language (DML)

SQL yang digunakan untuk melakukan pengolahan record atau memanipulasi data pada table dalam suatu basis data. Secara sederhana penulisan perintah SQL pada kelompok ini terdiri dari insert, select, update, dan delete. Berikut ini adalah deskripsi mengenai kelompok perintah DML tersebut.

1. *Insert*

Perintah SQL ini digunakan untuk melakukan entry atau penambahan suatu record pada table dalam basis data. Berikut ini merupakan aturan umum penulisan yang digunakan pada insert suatu table.

Sintak pertama:

```
insert into table_name (column1, column2,  
column3, ...) values (value1, value2,  
value3, ...);
```

Sintak kedua:

```
insert into table_name VALUES (value1,  
value2, value3, ...);
```

Contoh

Penerapan sintak pertama:

Menambahkan satu record pada tabel testing

Nis	nama	alamat	umur
D0219234	Anita	Jatinegara	19
D0220123	Kaco	UmbulHarjo	24

```
INSERT INTO `testing` (`Nis`, `nama`, `alamat`, `umur`) VALUES ('D0217521', 'Bambang', 'Klaten', '18');
```

Hasilnya

Nis	nama	alamat	umur
D0217521	Bambang	Klaten	18
D0219234	Anita	Jatinegara	19
D0220123	Kaco	UmbulHarjo	24

2. Select

Perintah SQL ini digunakan untuk memilih record yang akan ditampilkan berdasarkan data pada table dalam basis data. Berikut ini merupakan aturan umum dan variasi penulisan yang digunakan pada select.

Sintak untuk menampilkan salah satu kolom yang ada ditabel:

```
SELECT column1, column2, ... FROM  
table_name;
```

Sintak untuk menampilkan semua kolom yang ada ditabel:

```
SELECT * FROM table_name;
```

Contoh

Menampilkan CustomerName dan City pada table customer:

SELECT CustomerName, City FROM Customers;

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå
Blauer See Delikatessen	Mannheim
Blondel père et fils	Strasbourg
Bólido Comidas preparadas	Madrid

menampilkan seluruh kolom yang ada di table Customer:

SELECT * FROM customer;

CustomerName	ContactName	Address	City	PostalCode
Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023
Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP
Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22
Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306
Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000

3. Update

Perintah SQL ini untuk mengubah data dalam suatu table pada field tertentu dengan record baru berdasarkan suatu field sebagai kriteria pengubahan record-nya.

Sintak perintah update:

**UPDATE table_name SET column1 = value1,
column2 = value2, ... WHERE condition;**

Contoh

Merubah ContactName= 'Maria Anders' menjadi 'Alfred Schmidt' dan City= 'Frankfurt'

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	9-958 22	Sweden

```
UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;
```

Hasilnya:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	9-958 22	Sweden

Merubah dua record sekaligus yaitu merubah yang Country='Mexico' dirubah ContactName menjadi 'Juan'

```
UPDATE Customers SET ContactName='Juan' WHERE Country='Mexico' ;
```

Hasilnya:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

4. Delete

Perintah SQL ini digunakan untuk menghapus data dalam suatu table, berdasarkan suatu field sebagai kriteria penghapusan record-nya. Berikut ini merupakan aturan umum penulisan yang digunakan pada delete, yaitu:

Sintak perintah untuk menghapus salah satu record:

```
DELETE FROM table_name WHERE condition;
```

Sintak untuk menghapus seluruh record pada tabel:

```
DELETE FROM table_name;
```

Contoh

Menghapus CustomerName= ‘Alfreds Futterkiste’ dari tabel customer

```
DELETE FROM Customers WHERE  
CustomerName='Alfreds Futterkiste';
```

Menghapus seluruh record yang ada di tabel mahasiswa

```
DELETE FROM mahasiswa;
```

Data Control Language (DCL)

Data Control Language (DCL) merupakan perintah SQL yang digunakan untuk melakukan pengaturan hak akses suatu objek data para pengguna basis data. Secara sederhana, penulisan perintah SQL pada kelompok ini terdiri dari grant, revoke.

1. Grant

Perintah SQL ini digunakan oleh seorang administrator basis data untuk memberikan hak aksesnya kepada pengguna tertentu, agar dapat mengakses suatu table dalam basis data. Hak akses tersebut adalah *insert, delete, update, dan select*. Berikut ini merupakan aturan umum penulisan yang digunakan pada Grant.

Sintak:

```
GRANT hak_akses ON nama_tabel TO  
pengguna_tertentu;
```

Contoh

- a. Pemberian hak akses INSERT dan DELETE pada tabel mahasiswa kepada Budi

```
GRANT INSERT, DELETE ON mahasiswa TO  
Budi;
```

- b. Pemberian hak akses UPDATE pada tabel mahasiswa kepada Raisa

```
GRANT UPDATE ON mahasiswa TO Raisa;
```

2. Revoke

Perintah SQL ini digunakan oleh seorang administrator basis data, untuk membatalkan / menghentikan hak akses yang telah diberikan kepada pengguna tertentu, agar tidak dapat mengakses table dalam basis data. Berikut ini merupakan aturan umum penulisan yang digunakan pada revoke.

Sintak:

```
REVOKE hak_akses ON nama_tabel FROM  
pengguna_tertentu;
```

Contoh

- a. Membatalkan hak akses INSERT dan DELETE pada tabel mahasiswa kepada Budi

```
REVOKE INSERT, DELETE ON mahasiswa FROM  
Budi;
```

- b. Membatalkan hak akses UPDATE pada tabel mahasiswa kepada Raisa **REVOKE UPDATE ON mahasiswa FROM Raisa;**

Perintah SQL Menggunakan Operator Dasar

Setelah pembahasan sebelumnya kita telah menjelaskan jenis perintah SQL, yaitu DDL, DML, dan DCL, pada bagian ini akan dipaparkan beberapa perintah SQL dengan penggabungan klausa, operator, dan fungsi. Berikut ini merupakan hal umum untuk perintah SQL.

1. Where

Where ini digunakan untuk melakukan seleksi pada record, yang sesuai dengan syarat suatu kriteria pada suatu kondisi perintah SQL. Pada table dibawah ini disajikan operator matematika yang digunakan oleh klausa where.

Tabel Operator Matematika

No.	Operator	Arti
1	=	Sama dengan
2	<>	Tidak sama dengan
3	<	Lebih kecil
4	<=	Lebih kecil sama dengan
5	>	Lebih besar
6	>=	Lebih besar sama dengan

Sintak

perintah Where dengan Select:

```
SELECT column1, column2,...FROM table_name  
WHERE condition;
```

Contoh

Menampilkan Country='Mexico' dari tabel customer

```
SELECT * FROM Customers WHERE  
Country='Mexico';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Aria Trujillo Emparedados y helados	Aria Trujillo	Ave. de la Constitución 3222	México D.F.	05031	Mexico
2	Antonio Moreno Tequilería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
11	Centro comercial HogarQuinto	François Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
50	Pendes Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cañón 321	México D.F.	05033	Mexico

** Penggunaan WHERE juga bisa digunakan dalam perintah UPDATE dan DELETE.

2. Operator Logika *And, Or* dan *Not*

Operator logika and dan or digunakan untuk menggabungkan seleksi pada record yang syaratnya lebih dari satu kondisi. Pada operator logika, not digunakan untuk negasi dari kondisi tersebut.

Sintak operator:

a. AND

```
SELECT column1, column2,...FROM  
table_name WHERE condition1 AND  
condition2 AND condition3 ...;
```

b. OR

```
SELECT column1, column2,...FROM  
table_name WHERE condition1 OR  
condition2 OR condition3 ...;
```

c. NOT

```
SELECT column1, column2,...FROM  
table_name WHERE NOT condition;
```

Contoh

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Oberk Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	993 82	Sweden

- a. Menampilkan customer city='Berlin' dan Country='Germany'

```
SELECT * FROM Customers WHERE  
Country='Germany' AND City='Berlin';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Oberk Str. 57	Berlin	12209	Germany

- b. Menampilkan customer city='London' atau Country='Mexico'

```
SELECT * FROM Customers WHERE  
Country='Mexico' AND City='London';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

- c. Menampilkan costomer yang country nya bukan jerman

```
SELECT * FROM Customers WHERE NOT
Country='Germany' ;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2322	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05021	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Järfälla	S-958 22	Sweden

3. Between dan Not Between

Between digunakan untuk mengolah data suatu nilai dalam range tertentu. Not between merupakan negasinya, yaitu mengolah suatu nilai diluar dari range yang telah ditentukan.

Sintak / perintah

```
Between: SELECT column_name(s) FROM
table_name WHERE column_name BETWEEN value1
AND value2;
```

Contoh

- a. Menampilkan semua record dari tabel produk dengan harga 10000 sampai dengan 20000

```
SELECT * FROM produk WHERE harga BETWEEN
10000 AND 20000;
```

- b. Menampilkan semua record dari tabel produk dengan harga tidak antara 10000 sampai dengan 20000

```
SELECT * FROM produk WHERE harga NOT
BETWEEN 10000 AND 20000;
```

- c. Menampilkan semua mahasiswa dari tabel mahasiswa antara nama ‘Budi’ dan Raisa diurutkan berdasarkan nama:

```
SELECT * FROM mahasiswa WHERE nama NOT
BETWEEN 'Budi' AND 'Raisa' ORDER BY
nama;
```

4. *Like* dan *Not Like*

Like digunakan untuk mencari suatu teks yang sesuai berdasarkan kata depan (prefix), kata tengah (infix), kata akhir (suffix). *Not like* merupakan pernyataan negasinya, yaitu mencari suatu teks yang tidak sesuai dengan kriteria *like*. Pada table 2 disajikan mengenai pola yang digunakan oleh operator *like* dan *not like* untuk mencari kesesuaian kata. Pada operator ini terdapat penggunaan tanda persen (%) untuk mewakili satu atau lebih kecocokan karakter, dan tanda underscore (_) untuk mewakili satu saja kecocokan karakter yang dicari.

Tabel 2. Pola penggunaan Percent (%) dan Underscore (_)

NO	Pola	Penjelasan
1	a%	Cara untuk mencari kesesuaian pada suatu kata yang berawalan 1 atau beberapa huruf a atau A
2	%a%	Cara untuk mencari kesesuaian pada suatu kata yang didalamnya mengandung 1 atau beberapa huruf a atau A
3	%a	Cara untuk mencari kesesuaian pada suatu kata yang berakhiran 1 atau beberapa huruf a atau A
4	a_	Cara untuk mencari kesesuaian pada suatu kata yang berawalan hanya 1 huruf a atau A yang sama
5	_a	Cara untuk mencari kesesuaian pada suatu kata yang berawalan hanya 1 huruf a atau A yang sama
6	_a	Cara untuk mencari kesesuaian pada suatu kata yang berakhiran hanya 1 huruf a atau A yang sama

Sintak perintah

Like: `SELECT column1,column2,... FROM
table_name WHERE columnN LIKE pattern;`

Contoh

Menampilkan nama costumer yang berawalan huruf a:

```
SELECT * FROM Customers WHERE  
CustomerName LIKE 'a%';
```

CustomerName	ContactName	Address	City
Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.
Around the Horn	Thomas Hardy	120 Hanover Sq.	London

Daftar Pustaka

- B. Nugroho, *Panduan Lengkap Menguasai Perintah SQL*. Jakarta: Mediakita, 2008.
- Indrajani, *Sistem Basis Data dalam paket Five in One*. Jakarta: PT. Elex Media Komputindo, 2009.
- Hoffer.J.A., Prescott.M.B, McFadden.F.R *Modern Database Management*. . Prentice Hall Intern, 2005.

Profil Penulis



Asmawati S

Penulis mulai berkecimpung dan mengenal ilmu komputer dimulai pada tahun 2002 silam. Hal tersebut membuat penulis memilih untuk melanjutkan Pendidikan pada jenjang Diploma 1 dengan memilih Jurusan Manajemen Informatika dan lulus pada tahun 2003 di AMIK Rezky Makassar. Penulis kemudian melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S1 di prodi Sistem Informasi pada tahun 2012. Tiga tahun kemudian, penulis menyelesaikan studi S2 di prodi PENDIDIKAN TEKNOLOGI KEJURUAN PROGRAM PASCA SARJANA UNM MAKASSAR dan selesai tahun 2017.

Penulis memiliki kepakaran dibidang Web Development dan Sistem Informasi. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh Kemenristek DIKTI. Selain peneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini.

Email Penulis: asmawati.s@unsulbar.ac.id

7

ADVENCED SQL (EMBEDDED DAN DYNAMIC)

Asep Abdul Sofyan, S.Kom., M.Kom.

Universitas Islam Syekh Yusuf Tangerang

Advenced SQL

Seiring dengan perkembangan teknologi komputer yang berkembang dari berbagai macam segi, baik itu teknologi yang terkait dengan bahasa pemrograman, dan implementasi terhadap framework yang mengimplementasi bahasa tersebut. Berkembang pula database dan dengan berbagai implementasi yang dapat dilakukan didalam pengembangan perangkat lunak, kita belum membicarakan berkembang pula berbagai jenis database yang dapat digunakan di beberapa konteks sebuah aplikasi yang berbeda, misalnya kita berbicara terkait sebuah database yang baik digunakan dengan kaitannya sebuah sosial media tentu berbeda dengan sebuah database yang baik dan tepat digunakan dengan sebuah aplikasi berbasis transaksi.

Pembahasan kita akan berkutat pada sebuah implementasi terhadap sebuah sistem aplikasi dengan memperhatikan dari berbagai hal. Ketika kita berbicara sebuah aplikasi dan kaitannya dengan implementasi database yang digunakan maka kita harus memandang dari berbagai segi, misalnya terkait besaran datanya, besaran transaksinya, kecepatan prosesnya dan lain sebagainya.

Ketika kita mengeksekusi sebuah sintak sql dalam konteksnya implementasi data sederhana atau dengan data yang cakupannya masih belumlah luas, tentu akan berbeda dengan eksekusi sintak sql dalam sebuah aplikasi e-commerce yang sedang live dan digunakan secara masive.

Mari kita ambil contoh sederhana, misalnya kita mengeksekusi sintaks sederhana dalam menampilkan sebuah data produk dari dalam database yang sedang digunakan, “select * from produk” dengan kaitannya sebuah data sampel, atau implementasi sederhana, memilih eksekusi sebuah “*” yaitu semua kolom tentu tidak akan berpengaruh besar dalam result yang ditampilkan jika hanya terdapat 10 kolom, namun akanlah sangat berpengaruh jika tabel tersebut mempunyai kolom yang lumayan sangat banyak jumlahnya.

Aspek lainnya dari sebuah ekseskusi query pemilihan produk tersebut kita juga perlu mempertimbangkan berapa banyak data yang akan ditampilkan, dengan kaitannya sql yang di ekseskusi diatas, tentu diasumsikan akan menampilkan data dalam jumlah besar, tentu akan menjadi hal yang berbeda jika di eksekusi didalam sebuah database table dengan jumlah row yang sangat besar.

Dalam implementasi sql tingkat lanjut, diharuskan tidak hanya berpandangan dari error atau tidaknya sebuah eksekusi sintaks sql tersebut, namun haruslah juga mempertimbangkan dari berbagai macam aspek pertimbangan lainnya, seperti bagaimana efisiensinya, kecepatannya, model datanya dan lain sebagainya.

Adapun pembahasan kita selanjutnya yaitu membahas tentang embeded dan dinamic sql yang juga merupakan beberapa hal dari sekian banyak penerapan tingkat lanjut daripada SQL itu sendiri dalam kaitannya pembahasan tingkat lanjut, mengenai sql.

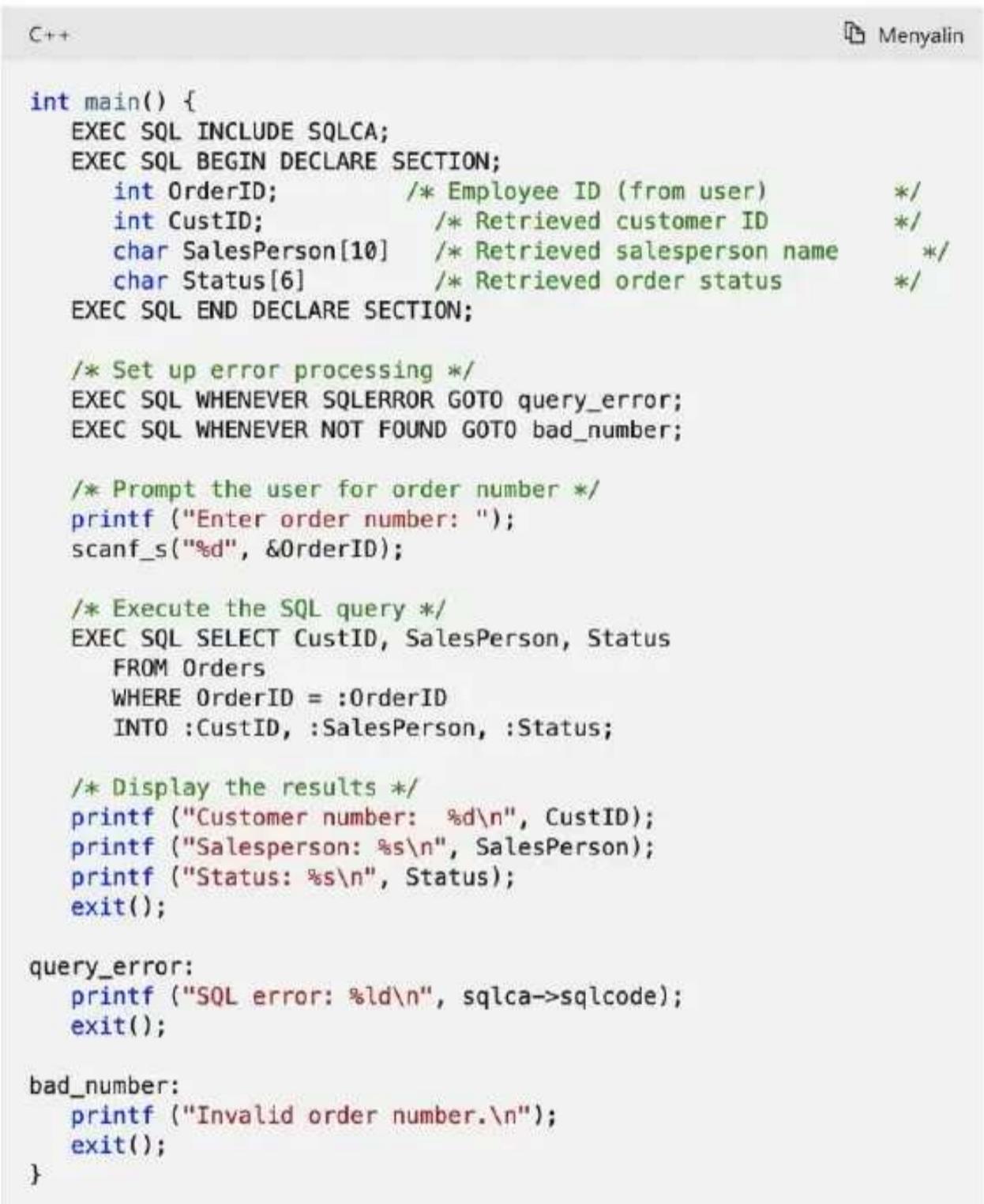
Embedded SQL

Pembahasan tentang embedded SQL sebenarnya adalah pembahasan yang memang sudah jarang sekali dibahas, karena pembahasan mengenai ini, hanya spesifik kepada beberapa domain bahasa pemrograman tertentu saja, terutama sekali bahasa pemrograman utama, dan merupakan bahasa pemrograman yang sudah sangat populer seperti C/C++.

Namun akan dijelaskan nanti beberapa pendekatan baru yang juga merupakan pendekan dari embedded SQL dengan pendekatan yang sama namun mengadopsi pendekatan embeded SQL.

Apa yang dimaksud dengan embedded sql ialah, sebuah metode atau mekanisme menggabungkan kekuatan dari pada sebuah bahasa pemrograman dan kemampuan manipulasi daripada bahasa SQL dalam satu kesatuan. Secara praktik embedded SQL statement ditulis dalam kode yang sama(inline), atau dalam preserved code bersama dengan bahasa pemrograman yang digunakan (*host programming*).

Perbedaan mendasar daripada embedded SQL adalah yang paling mencolok bahwa perintah SQL ditulis langsung dalam satu kesatuan bahasa yang digunakan, misalkan C/C++, bukan ditempatkan dalam sebuah variabel misalkan dalam variabel String, embedded SQL kemdian diolah oleh SQL preprocessor yang kemdian digantikan oleh host programmingnya dengan memanggil library yang digunakan, untuk lebih memahami konsepnya perhatikan contoh daripada embedded SQL berikut.



```

C++ Menyalin

int main() {
    EXEC SQL INCLUDE SQLCA;
    EXEC SQL BEGIN DECLARE SECTION;
        int OrderID;          /* Employee ID (from user) */
        int CustID;           /* Retrieved customer ID */
        char SalesPerson[10]   /* Retrieved salesperson name */
        char Status[6];        /* Retrieved order status */
    EXEC SQL END DECLARE SECTION;

    /* Set up error processing */
    EXEC SQL WHENEVER SQLERROR GOTO query_error;
    EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

    /* Prompt the user for order number */
    printf ("Enter order number: ");
    scanf_s("%d", &OrderID);

    /* Execute the SQL query */
    EXEC SQL SELECT CustID, SalesPerson, Status
        FROM Orders
        WHERE OrderID = :OrderID
        INTO :CustID, :SalesPerson, :Status;

    /* Display the results */
    printf ("Customer number: %d\n", CustID);
    printf ("Salesperson: %s\n", SalesPerson);
    printf ("Status: %s\n", Status);
    exit();

query_error:
    printf ("SQL error: %ld\n", sqlca->sqlcode);
    exit();

bad_number:
    printf ("Invalid order number.\n");
    exit();
}

```

Dengan memperhatikan secara seksama dapat dilihat bahwa sintaksis SQL tidak ditulis dalam sebuah variabel namun menjadi satu kesatuan dalam bahasa pemrograman yang digunakan (host programming) dalam contoh menggunakan bahasa pemrograman C++.

Penjelasan daripada potongan bahasa pemrograman diatas yang mengimplementasikan embedded SQL adalah:

1. Variabel-variabel yang digunakan (*Host Variabel*) di deklarasikan di dalam sebuah lingkup yang di apit oleh sintaksis dengan kata kunci BEGIN DECLARE SECTION dan END DECLARE SECTION, jika diperhatikan sebuah variabel akan dimulai dengan tanda titik dua/colon (:) ketika variabel tersebut

berada didalam sebuah sintaksis embedded SQL. Tanda titik dua, memungkinkan precompiler membedakan antara variabel dalam bahasa induk pemrogramannya dan objek databasenya, misalnya tabel dan kolom yang memiliki nama yang sama.

2. Tipe Data yang di dukung oleh DBMS dan induk bahasa pemrograman yang digunakan bisa jadi berbeda satu sama lainnya. Dan ini ber efek pada variabel induk bahasa pemrogramaman karena memainkan dua peran, disatu sisi variabel induk adalah variabel bahasa pemrograman induknya yang dideklarasikan dan digunakan untuk memanipulasi dalam statemen bahasa pemrograman. Sedangkan disisi lainnya juga digunakan dalam embedded SQL yang berfungsi untuk menerima data hasil eksekusi. Namun apabila tidak ada tipe yang sesuai yang berhubungan dengan tipe data dalam DBMSnya, maka DBMS otomatis akan mengkonversikan tipedatanya. Namun, karena setiap DBMS memiliki aturan dan keistimewaannya sendiri yang terkait dengan proses konversi, tipe variabel yang digunakan induk bahasa pemrogramannya harus dipilih dengan hati-hati.
3. Penanganan Error, DBMS akan mengirimkan run-time error yang terjadi kepada program aplikasi melalui Area Komunikasi SQL (SQLCA), dalam sampel code dapat diperhatikan pada statemen pertama adalah INCLUDE SQLCA. Kode sintak diatas memberitahukan kepada precompiler untuk memasukan struktur SQLCA didalam program, yang mana ini dibutuhkan untuk aplikasi dalam memproses laporan kesalahan (error report) yang diberikan oleh DBMS, dan untuk sintak WHENEVER...GOTO memberitahukan precompiler untuk meletakkan penanganan kesalahan yang terjadi untuk diproses pada cabang kode program yang telah diberi label jika kesalahan terjadi.
4. Singleton SELECT, sintaksis atau statemen yang digunakan untuk mengembalikan data bersifat Singleton SELECT, yaitu kelas yang hanya di inisiasi

sekali saja. Dalam sampel mengembalikan hanya satu row data, namun perlu di perhatikan di dalam kode sampel tidak mendeklarasikan cursor.

Pengimplementasian embedded SQL tidak bisa dilakukan dalam semua bahasa pemrograman dan juga database, keduanya perlu dicek ketersediaan fitur implementasinya namun sejauh ini database populer seperti MS SQL Server, ORACLE, SYBASE, PostgreSQL mendukung untuk implementasi embedded SQL.

Adapun bahasa pemrograman yang digunakan sebagai induk dari embedded SQL (host programming) juga tidak semua bisa mengimplementasikan embedded SQL sejauh ini hanya beberapa bahasa pemrograman yang sudah berumur panjang yang mengimplementasikannya, seperti C/C++, Fortran,

Bisa dikatakan untuk mengimplementasikan embedded SQL masih terbilang terbatas, dengan hanya pemrograman tertentu dan dengan database tertentu pula, namun begitu apakah embedded SQL bagaimanakah kelebihan dan kekurangan tentu hal ini perlu diperhatikan dan perlu dilakukan kajian terhadap aplikasi dan sistem yang akan dibuat terlebih mengingat minimnya resource dan support yang dapat dilakukan untuk pengimplementasianya.

Kelebihan dan kekurangan dari embedded SQL tentu bisa dilihat dari arsitekturnya dan penerapannya, yaitu dimana embedded SQL lebih cepat dan efisien, itu memungkinkan karena statemen(sintaks,perintah SQL) embedded SQL dicompile pada saat kompilasi induk programmingnya, namun sebaliknya Dynamic (Interactive) SQL dimana statemen dicompile pada saat run time, atau pada saat perintah di kirimkan kepada DBMS, dan tentu saja pada saat yang sama pulan berlaku beberapa pengecekan dan validasi sebelum akhirnya di jalankan oleh mesin eksekusi DBMS itu sendiri, dari situ dapat dilihat bahwa embedded SQL lebih unggul dalam memperingkas waktu eksekusi.

Selain daripada kelebihan yang telah dijelaskan perlu diketahui bahwa eksekusi perintah melalui mekanisme

embedded SQL akan lebih aman, karena bahaya terbesar dari sekuritas SQL itu sendiri, yaitu SQL Injection amat sangat sulit dilakukan, karena akan sangat sulit menyisipkan sintaksis atau statement yang dapat di exploit jika itu tergabung dengan bahasa pemrogramannya dan sudah terkompilasi, menjadi bahasa mesin.

Berbeda dengan pendekatan Dinamic SQL yang memungkinkan bagi seseorang untuk menyusupkan sebagian kecil sintaksis atau statemen bahkan karakter untuk menjadikan perintah SQL vulnerable melakukan hal-hal diluar kendali fungsi awalnya.

Namun begitu kekurangan dari embedded SQL juga tidak luput dari perhatian, karena sudah terbungkus dan terkompile dengan bahasa pemrograman ketika aplikasi dikompilasi, maka akan sangat sulit juga untuk melakukan perubahan2 dikemudian hari, tentu karena sifatnya yang bisa dibilang hard-coded didalam aplikasi, berbeda dengan Dinamic SQL, lebih tepatnya bahwa embedded SQL tidak fleksibel dalam melakukan perubahan-perubahan yang nantinya perlu dilakukan didalam penerapan aplikasi.

Dalam pembahasana embedded SQL, sudah dibandingkan pula dengan Dinamic SQL embedded SQL juga disebut dengan Static SQL. Pembahasan berikutnya akan dijelaskan yang dimaksud dengan Dinamic SQL

(SQL Dinamis) yang umumnya banyak diterapkan dan di implementasikan dalam berbagai macam penerapan aplikasi dalam integrasinya dengan database yang digunakan, selain daripada fleksibilitas dan diketahui bahwa dengan perkembangan teknologi yang hari demi hari, memungkinkan percepatan ataupun performa yang mungkin dihasilkan dari sebuah aplikasi dan database yang digunakan dapat dihasilkan dan dimaksimalkan dari berbagai sudut dan implementasi baik, dengan menerapkan dukungan implementasi tambahan dengan skema “Scale Up” ataupun dengan “Scale Right” yang bisa dilakukan dengan dukungan “development operation” sebagai pelengkap disisi development aplikasi ataupun implementasi dalam sisi arsitektur.

Dinamic SQL

Embedded SQL sudah dibahas, dapat diketahui pula sisi kekurangan dan kelebihan yang dapat dihasilkan daripada implementasi yang dapat dilakukan, namun sudah diketahui bahwa *Dinamic SQL* (*SQL Dinamis*) adalah merupakan implementasi yang banyak dilakukan dalam penerapan pengembangan aplikasi yang dilakukan pada saat ini, bukan dengan tanpa sebab, perkembangan bahasa pemrograman bisa diketahui memiliki peranan yang sangat besar didalamnya, dan database yang digunakan umumnya juga umumnya berkembang, maka dinamisasi didalamnya menjadi peran penting yang juga harus diperhitungkan.

Performa memang menjadi hal utama, namun bukan hal yang satu-satunya diperhitungkan dalam pengembangan aplikasi, karena bisa dilakukan optimasinya dari sisi lain yang bisa memperkecil beban development aplikasi, bisa dibayangkan dengan penerapan *embedded SQL* yang mana sintaksis SQL di lakukan bersama dalam bahasa pemrograman induknya tentu menjadikan developmen aplikasi tambahan dalam segi development waktu dan maintenance nantinya, karena beban ketidak dinamisannya perlu diperhitungkan.

Penerapan *SQL dinamis* juga perlu diperhitungkan sejauh mana dinamisasi dapat dilakuakn dalam aplikasi jika dinamisasi eksekusi juga dilakuakn di semua objek

database pertimbangkan pula sisi keamanan yang perlu dilakukan, umumnya penerapan *SQL dinamis* dimasukan dilengkapi dengan penanganan dari sisi keamananannya.

SQL Dinamis adalah penerapan dinamisasi atau fleksibilitas dalam pengeksekusian sintaksis atau statemen SQL dalam sebuah aplikasi. Misalnya secara sederhana bahwasanya dalam sebuah aplikasi kita meletakan sintaksis sql dalam sebuah variabel dan dengan value criteria dinamis seiring dengan penggunaan aplikasinya itu juga merupakan implementasi sederhana yang disebut *SQL Dinamis*.

Namun pada penerapan sebenarnya dalam *SQL dinamis*, atau disbut juga *Interactive SQL* (*SQL Interaktif*) dimana

salah satu penerapannya menggunakan statemen *EXECUTE*, *EXECUTE IMMEDIAT* dan *PREPARE*, perlu diketahui juga penerapan dalam objektifitasnya dengan masing-masing database juga berbeda. Mengikuti aturan-aturan lebih lanjut terkait database yang digunakan tentu hal ini akan merujuk kepada manual book, atau panduan daripada database yang digunakan.

Sebagai contoh sederhana akan diberikan sampel sederhana eksekusi perintah SQL dengan sederhana dan dengan menerapkan SQL dinamis, dalam contoh akan digunakan database MS SQL Server, dan database yang digunakan adalah *adventureworks*.

Misalnya akan dijalankan perintah untuk menampilkan data karyawan dalam table “Employee” maka sintaks SQL yang akan dieksekusi adalah sebagai berikut.

```
select [EmployeeID],LoginID from HumanResources.Employee  
where MaritalStatus='M' and Gender='F'
```

Sintaksis tersebut dapat di sematkan dalam variabel aplikasi bahasa pemrograman dalam sebuah variabel misalnya adalah String(tekst) yang mana program akan mengirimkan variabel tersebut kepada mesin database yang kemudian akan melakukan tugas awalnya dari mulai mengkonversikan, lalu validasi optimasi dan lain sebagainya sehingga akhirnya di eksekusi oleh mesin diharapkan. Terkadang kriteria didalam sintaksis SQL didalamnya pun juga masih di dinamisasi seiring dengan input user yang dilakukan dalam tampilan form yang ada didalam pengguna akhir perangkat lunak, atau *software*.

Namun dalam penerapan SQL Dinamis yang seharusnya dilakukan maka akan ditambahkan beberapa variabel tambahan dalam statemen atau sintaks tersebut. Maka perlu disematkan *EXECUTE* dan disesuaikan dengan *SP_EXECUTESQL* dengan parameter dasar berikut:

“ EXECUTE SP_EXECUTESQL <SQL>, <Parameter Definitions>, <Parameter Values>

Kenapa perlu mengimplementasikan SQL Dinamis, misalnya adalah ditemukan didalam analisa lapangan bahwa situasi yang dihadapkan adalah mengharuskan sebuah aplikasi dapat bekerja dengan menjalankan perintah statemen SQL yang bersifat dinamis, misalnya dengan nama tabel dalam database yang terkadang gabungan antara nama objek tahun, atau susunan fieldnya juga berubah sesuai dengan hari. Maka dengan permasalahan diatas perlu dinamisasi statemen SQL didalamnya maka implemetasi yang seharusnya dengan pendekatan database MS SQL Server penyesusaiannya

adalah seperti berikut:

```
Declare @sql nvarchar(500)='select [EmployeeID],LoginID from HumanResources.Employee  
where MaritalStatus=@MS and Gender=@G';  
Declare @ParamDef nVarchar(50) = '@MS Varchar(5), @G Varchar(5)';  
  
execute sp_executesql @sql, @ParamDef, @MS='M', @G='F';
```

Dapat dilihat bahwa dari mulai query dan parameter yang ada dialamnya merupakan variabel pula yang harus di jadikan input dalam pengeksekusiannya.

Perlu diperhatikan dengan sangat seksama bahwa, hal seperti ini akan sangat memiliki permasalahan dalam segi keamanan, karena eksekusi yang teramat dinamis memerlukan tambahan logic pengamanan eksekusi didalamnya misalnya didalam eksekusi perintah dinamisasi diatas mengharuskan klausul where dan dengan beberapa parameter tambahan yang berfungsi secara logis untuk memvalidasi skema filter statemen SQL dengan kata lain yang lebih sederhana agar sintaksis yang di eksekusi tidak “polosan”.

Pendekatan dan Teknik Lain

Perkembangan teknologi dalam bidang bahasa pemrograman, baik implementasi sederhana dalam internal bahasa pemrograman tersebut ataupun framework yang juga mengimplementasikannya secara lebih lanjut.

Perkembangan selanjutnya juga perlu diperhatikan bahwa dalam penerapan aplikasi amat sangat bergantung dengan penyimpanan data, dan data bergantung pada database yang digunakan, dalam hal permasalahan ini di implementasikanlah teknologi ORM yang berfungsi sebagai jalan tengah, dalam eksekusi perintah SQL dimana tidak lagi bergantung pada databasenya, tapi hanya dengan bahasa ORMnya saja.

Secara sederhana statemen SQL nya tidak lagi dengan SQL secara DBMS namun sudah lebih kepada library apa yang digunakan dan bahasa yang ada di dalamnya, lalu library tersebut yang akan berkomunikasi dengan database yang digunakan. Contohnya adalah dengan menerapkan teknologi HIBERNATE, atau dengan teknologi yang sama namun berbasis microsoft yaitu Entity Framework. Dimana dengan HIBERNATE bisa menggunakan bahasa HQL dan Entity Framework dengan menggunakan LINQ.

Yang menarik adalah dengan menggunakan LINQ (Language Integrated Query) perintah SQL dapat di eksekusi didalam bahasa induknya misalnya adalah C# dalam platform yang sama yaitu Microsoft .Net, maka ini tidak jauh berbeda dengan pendekatan embedded SQL. Menarik bukan.?

Dengan adanya bahasa-bahasa tersebut baik itu HQL (Hibernate Query Language) ataupun LINQ memberikan warna dalam pendekatan-pendekatan lain dalam implementasi sebuah aplikasi dalam menggunakan piranti database.

Hal yang juga menarik yang seharusnya tidak luput dari pengamatan bahwa penggantian database menjadi lebih mudah dilakukan dengan menggunakan Hibernate,

karena bahasa yang digunakan tetaplah sama yaitu HQL (tentu dalam aplikasi yang dibuat mengharuskan hanya menggunakan HQL) namun bisa dengan mengganti2 database yang mungkin dalam keadaan tertentu perlu dilakukan. Dengan di implementasikan HQL aplikasi akan

lebih general dalam mengelola data yang tersimpan dalam database dalam hal bahasa SQL yang digunakan.

Dan perlu diperhatikan juga, saat ini ada banyak sekali teknologi-teknologi semisal yang juga diterapkan dalam bahasa pemrograman, terutama sekali dalam framework lanjutan dari bahasa pemrograman yang menjadi induk ataupun host dari framework tersebut. Maka pada dasarnya semua bahasa mengadopsi logis yang sama, namun memiliki kalimat yang berbeda dalam penggunaanya, umumnya bisa dikatakan “mirip” secara gaya bahasa.

Daftar Pustaka

Pohan, Husni I. (2002). SQL + Tutorial Plus Studi Kasus Dengan Oracle dan Sybase. Bandung, Indonesia: Informatika Bandung.

Referensi online:

<https://docs.microsoft.com/id-id/sql/odbc/reference/embedded-sql-example?view=sql-server-2017>

https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc37776.1252/html/connpb/Using_EMBEDDED_SQL_with_Microsoft_SQL_Server.htm

<https://www.geeksforgeeks.org/difference-static-dynamic-sql/>

<https://www.tutorialspoint.com/difference-between-static-sql-and-dynamic-sql>

https://en.wikipedia.org/wiki/Embedded_SQL

<https://www.rastertek.com/oratut05.html>

<https://www.quora.com/What-is-embedded-SQL-Does-PHP-support-it>

<https://www.quora.com/What-is-embedded-SQL>

<http://suherman.asia/w2/dynamic-sql.html>

Profil Penulis



Asep Abdul Sofyan

Penulis dilahirkan di kota Tangerang, tertarik dengan computer Ketika era 1990an dengan mengikuti kursus-kursus komputer baik itu DOS, microsof office, dan dilanjutkan Ketika menyelesaikan studi pada perguruan tinggi profesi dengan mengambil jurusan Informatika Komputer. disela-sela bekerja penulis juga menjadi salah satu tim konsultan lepas dalam pemgembangan aplikasi pada salah satu lembaga negara, saat ini penulis masih aktif sebagai salah satu dosen tetap di perguruan tinggi swasta di wilayah Tangerang Banten, pada program studi Teknik informatika, Penulis menyelesaikan Studi S1 di Program Studi Sistem Informasi STMIK PGRI Tangerang dan pada tahun 2014 penulis menyelesaikan Studi Strata 2 (Magister) di prodi Program Pascasarjana Ilmu Komputer Universitas Budiluhur Jakarta.

Email Penulis : asep.abdulsofyan@unis.ac.id

Email Pribadi : asep.abdul.sofyan@gmail.com

8

RDBMS (*RELATIONAL DATABASE MANAGEMENT SYSTEM*)

Djamaludin, S.Kom., M.Kom.

Universitas Islam Syekh Yusuf Tangerang

RDBMS

RDBMS kependekan dari *Relational Database Management System*, merupakan suatu konsep yang digunakan dalam mengolah suatu database yang terintegrasi dengan table-tabel satu dengan lainnya dalam database melalui kunci yang dimiliki. sebelum lebih lengkap mempelajari konsep tentang *Relational Database Management System* (RDBMS), alangkah baiknya terlebih dahulu kita pelajari apa itu; konsep, Relational, Database, Managamen dan System,

1. Konsep

Konsep adalah

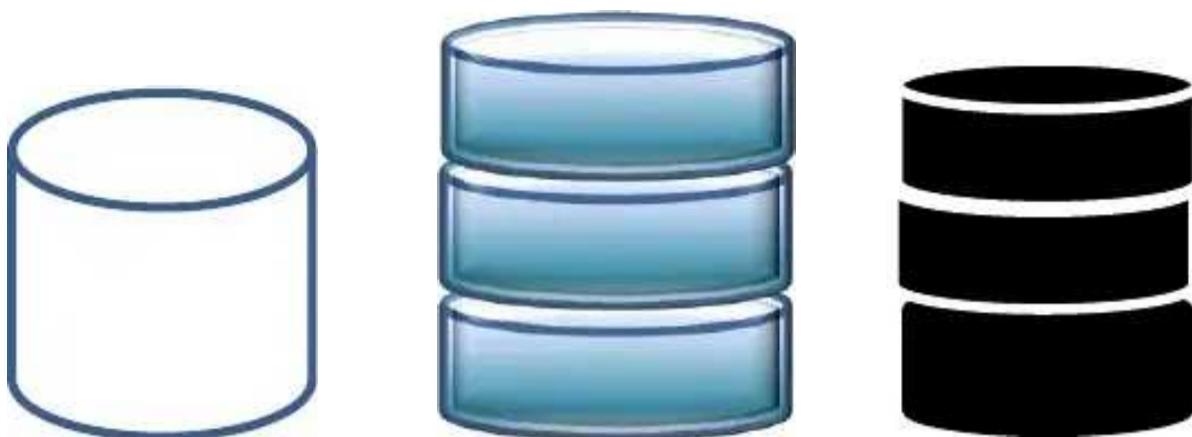
2. Pengertian Relational

Relational adalah suatu hubungan, dalam pengertian database ini Relational adalah hubungan antar table.

3. Pengertian Database (Basis Bata)

Database adalah sekumpulan data dan informasi yang tersimpan didalam suatu media komputer secara bersama-sama dan salin berelasi ataupun berdiri sendiri, data-data tersebut disimpan dalam bentuk

table yang berisi baris (*Row*) dan Kolom (*Coloum*) dalam mengkasesnya dibutuhkan sebuah system yaitu Database Management System (DBMS).



Gb. Simbol-simbol *database*

a. Operasi Basis Data

Berikut merupakan operasi dasar basisdata:

Create database : Perintah yang digunakan untuk membuat basisdata dengan nama yang diberikan

Drop database : Perintah yang digunakan untuk menghapus basisdata dengan perintah yang diberikan

Create table : Perintah yang digunakan untuk membuat suatu table dalam basis data

Drop table : Perintah yang digunakan untuk menghapus table dalam basisdata

Insert : Perintah yang digunakan untuk memasukan data (*record*) pada suatu table

Delete : Perintah yang digunakan untuk menghapus data (*record*)

b. Pemanfatan Basis Data

Pemanfataan basisdata yaitu:

1) Salah satu komponen penting dalam suatu system informasi, karena merupakan dasar dalam menyediakan informasi

2) Menentukan kualitas informasi: akurat, tepat waktu dan relevan

3) Mengurangi duplikasi data (data redundancy)

4) Hubungan data dapat ditingkatkan

5) Manipulasi terhadap data dengan cepat dan mudah

6) Efisiensi penggunaan ruang penimpanan

c. Penerapan Basisdata

Tidak ada system informasi yang bisa dibangun tanpa adanya basisdata, sehingga bisa dikatakan bahwa posisi basisdata pada sebuah system informasi adalah sangat penting

d. Kriteria Basisdata

1) Berorientasi pada data bukan berorientasi pada program

2) Dapat digunakan oleh beberapa program aplikasi tanpa mengubah basisdatanya

3) dapat berkembang dengan mudah, baik volume maupun strukturnya

4) dapat digunakan dengan cara berbeda-beda

5) kerangkapan data minimal

e. Siklus Hidup Basis Data

Siklus hidup basis data bermaksud mendukung perancangan basis data dimulai dari perancangan skema logika basis data, pengalokasian data melewati jaringan computer, serta pendefinisikan skema spesifik system basis data local. Setelah itu proses implementasi dan perawatan (maintenance) bermaksud menjaga keandalan sistem basis data yang telah direncanakan sebelumnya. Jadi, siklus hidup basis data antara lain: menentukan kebutuhan informasi, perancangan logika, perancangan fisik, dan implementasi (Toerey, et al, 2011).

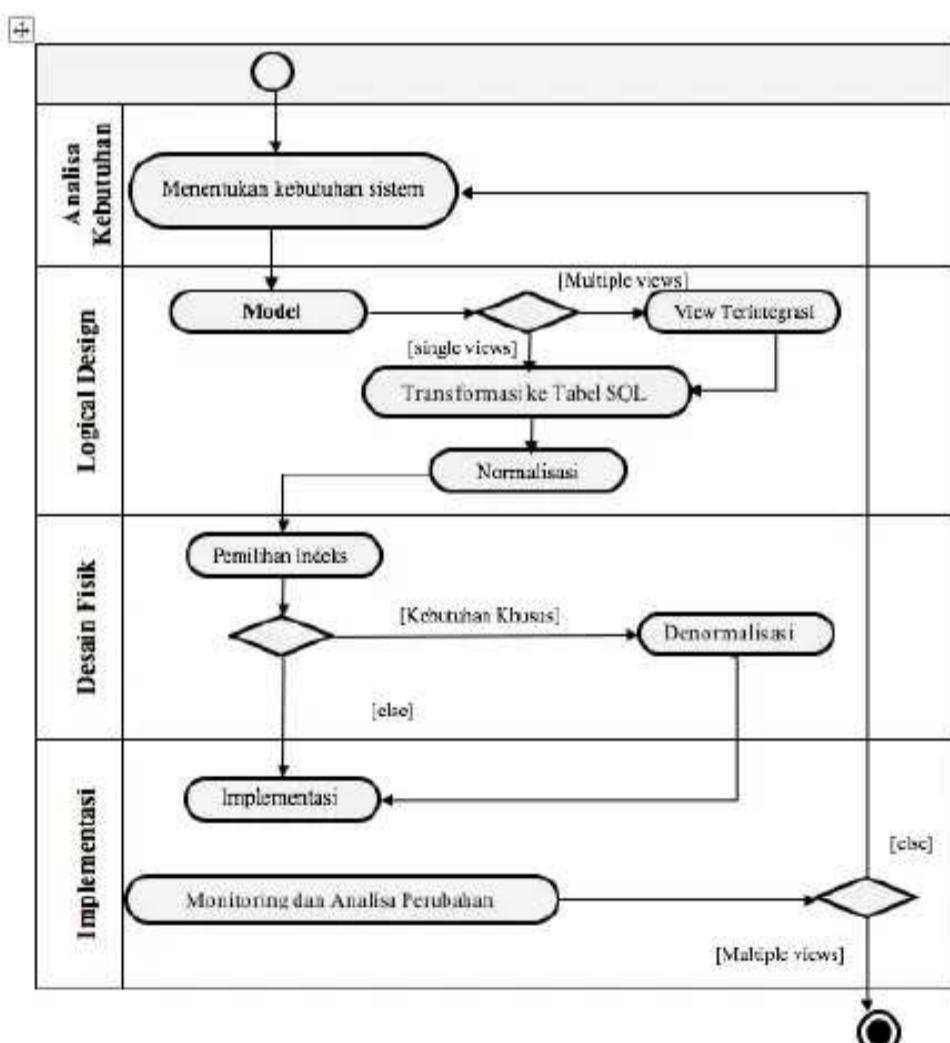
1) Analisa Kebutuhan (*requirement analysis*). Tahap ini dilakukan dengan mewawancarai baik pensuplai dan pengguna data serta

menghasilkan spesifikasi formal kebutuhan. Spesifikasi-spesifikasi tersebut adalah: data

yang dibutuhkan untuk proses, hubungan antar data, dan platform perangkat lunak yang dibutuhkan.

- 2) *Logical Design.* Dimulai dari perancangan skema global dengan konsep model data hingga konversi dari model menjadi tabel-tabel. Model harus bisa menunjukkan hubungan-hubungan antar data dan biasanya dikembangkan dengan *Entity Relationship* (ER) atau UML. Tahap ini sangat penting sebelum masuk ketahap berikutnya. Tahaan yang biasanya dilalui oleh perancang basis data difase ini adalah: *Conceptual data Modeling, View integration, transformation, Transformasi ke table SQL* dan normalisasi table.
- 3) *Physical design.* Tahap ini mulai merealisasikan rancangan yang telah dibuat pada tahap sebelumnya ke system dengan melibatkan pemilihan indeks, partisi, kluster terhadap data. Sebisa mungkin tahapan ini membuat rancangan pada tahap logic menjadi nyata dalam suatu system. Pada tahap fisik ini dilakukan juga proses optimasi agar dihasilkan system yang cepat dan mampu menangani komputasi untuk data real yang besar. Terkadang diperlukan proses denormalisasi untuk mengurangi proses *join* yang membutuhkan proses yang banyak. Waktu yang dibutuhkan untuk tiap query dicatat untuk mengetahui kinerjanya. Penting untuk memperhatikan integritas data agar terhindar dari kesalahan saat mengolah data. Terutama untuk system Online Analytical Processing (OLAP). OLAP merupakan system yang berfokus ke Analisa data yang berada di database yang berukuran besar, berbeda dengan system transaksi yang hanya sebatas keluaran dan masukan dengan komputasi sederhana

- 4) *Database implementation, Monitoring, and Modification.* Ini merupakan tahap terakhir yang tidak kalah pentingnya dengan tahap sebelumnya, implementasi menyiapkan tabel-tabel yang akan digunakan oleh pengguna baik dengan menggunakan Date Definition Language atau step diolah dengan Data Manipulation Language (DML). Ketika database mulai beroperasi, tahap monitoring dimulai untuk memantau apakah performa dan kebutuhan system terpenuhi. Jika tidak memenuhi maka perlu dilakukan modifikasi agar sesuai dengan kebutuhan system, perhatikan gambar Siklus Hidup Basis Data pada halaman berikut ini;



Gb. Siklus Hidup Basis Data
(Rahmadya Trias Handayanto, ST, M.Kom., Ph.D dan
Herlawati, S.Si.,M.M.,M.Kom;
Pemrograman Basis Data di Matlab dengan MySQL dan
Microsoft Access, Penerbit Informatika)

4. Pengertian Sistem Managemen basis data

Pengertian sistem manajemen basis data adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses sumber data dengan cara praktis dan efisien.

5. Pengertian System

Sistem berasal dari bahasa Latin (*systēma*) dan bahasa Yunani (*sustēma*) adalah suatu kesatuan yang terdiri atas komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi, atau energi untuk mencapai suatu tujuan. Istilah ini sering digunakan untuk menggambarkan suatu set entitas yang berinteraksi, di mana suatu model matematika sering kali bisa dibuat.

Sistem juga merupakan kesatuan bagian-bagian yang saling berhubungan yang berada dalam suatu wilayah serta memiliki item-item penggerak, contoh umum misalnya seperti negara. Negara merupakan suatu kumpulan dari beberapa elemen kesatuan lain seperti provinsi yang saling berhubungan sehingga membentuk suatu negara di mana yang berperan sebagai penggeraknya yaitu rakyat yang berada di negara tersebut.

Kata "sistem" banyak sekali digunakan dalam percakapan sehari-hari, dalam forum diskusi maupun dokumen ilmiah. Kata ini digunakan untuk banyak hal, dan pada banyak bidang pula, sehingga maknanya menjadi beragam. Dalam pengertian yang paling umum, sebuah sistem adalah sekumpulan benda yang memiliki hubungan di antara mereka.

a. Elemen dalam Sistem

Pada prinsipnya, setiap sistem selalu terdiri atas empat elemen:

- 1) Objek, yang dapat berupa bagian, elemen, ataupun variabel. Ia dapat benda fisik,

abstrak, ataupun keduanya sekaligus; tergantung kepada sifat sistem tersebut.

2) Atribut, yang menentukan kualitas atau sifat kepemilikan sistem dan objeknya.

3) Hubungan internal, di antara objek-objek di dalamnya.

4) Lingkungan, tempat di mana sistem berada.

b. Elemen Sistem

Ada beberapa elemen yang membentuk sebuah sistem, yaitu: tujuan, masukan, proses, keluaran, batas, mekanisme pengendalian dan umpan balik serta lingkungan. Berikut penjelasan mengenai elemen-elemen yang membentuk sebuah sistem:

1) Tujuan

Setiap sistem memiliki tujuan (*Goal*), entah hanya satu atau mungkin banyak. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Tanpa tujuan, sistem menjadi tak terarah dan tak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem yang lain berbeda.

2) Masukan

Masukan (*input*) sistem adalah segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan yang diproses. Masukan dapat berupa hal-hal yang berwujud (tampak secara fisik) maupun yang tidak tampak. Contoh masukan yang berwujud adalah bahan mentah, sedangkan contoh yang tidak berwujud adalah informasi (misalnya permintaan jasa pelanggan).

3) Proses

Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna dan lebih bernilai, misalnya berupa informasi dan

produk, tetapi juga bisa berupa hal-hal yang tidak berguna, misalnya saja sisa pembuangan atau limbah. Pada pabrik kimia, proses dapat berupa bahan mentah. Pada rumah sakit, proses dapat berupa aktivitas pembedahan pasien.

4) Keluaran

Keluaran (*output*) merupakan hasil dari pemrosesan. Pada sistem informasi, keluaran bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

5) Batas

Yang disebut batas (*boundary*) sistem adalah pemisah antara sistem dan daerah di luar sistem (*lingkungan*). Batas sistem menentukan konfigurasi, ruang lingkup, atau kemampuan sistem. Sebagai contoh, tim sepak bola mempunyai aturan permainan dan keterbatasan kemampuan pemain. Pertumbuhan sebuah toko kelontong dipengaruhi oleh pembelian pelanggan, gerakan pesaing dan keterbatasan dana dari bank. Tentu saja batas sebuah sistem dapat dikurangi atau dimodifikasi sehingga akan mengubah perilaku sistem. Sebagai contoh, dengan menjual saham ke publik, sebuah perusahaan dapat mengurangi keterbatasan dana.

6) Mekanisme Pengendalian dan Umpan Balik

Mekanisme pengendalian (*control mechanism*) diwujudkan dengan menggunakan umpan balik (*feedback*), yang mencuplik keluaran. Umpan balik ini digunakan untuk mengendalikan baik masukan maupun proses. Tujuannya adalah untuk mengatur agar sistem berjalan sesuai dengan tujuan.

7) Lingkungan

Lingkungan adalah segala sesuatu yang berada di luar sistem. Lingkungan bisa berpengaruh terhadap operasi sistem dalam arti bisa merugikan atau menguntungkan sistem itu sendiri. Lingkungan yang merugikan tentu saja harus ditahan dan dikendalikan supaya tidak mengganggu kelangsungan operasi sistem, sedangkan yang menguntungkan tetap harus terus dijaga, karena akan memacu terhadap kelangsungan hidup sistem.

c. Jenis Sistem

Ada berbagai tipe sistem berdasarkan kategori:

- 1) Atas dasar keterbukaan:
 - a) sistem terbuka, di mana pihak luar dapat mempengaruhinya.
 - b) sistem tertutup.
- 2) Atas dasar komponen:
 - a) Sistem fisik, dengan komponen materi dan energi.
 - b) Sistem non-fisik atau konsep, berisikan ide-ide.

Dari penjelasan mengenai istilah-istilah, *Relational, Data Base, Management system basis data*, bisa disimpulkan bahwa *Relational Database Management System* (RDBMS) dalam bahasa indonesia disebut dengan sistem pengelola basis data relasional, merupakan perangkat lunak (*software*) pengelola basis data yang didasari model relasional atau jenis basis data relasional, sistem perangkat lunak ini mampu menentukan (*define*), membuat (*create*), memelihara (*construct*) juga mengontrol (*access control*) data pada basis data yang saling berelasi, bisa juga bahwa RDBMS adalah suatu model untuk memproses suatu basis data yang dapat diintegrasikan, sehingga diperlukan suatu konsep untuk

RDBMS (RELATIONAL DATABASE MANAGEMENT SYSTEM)

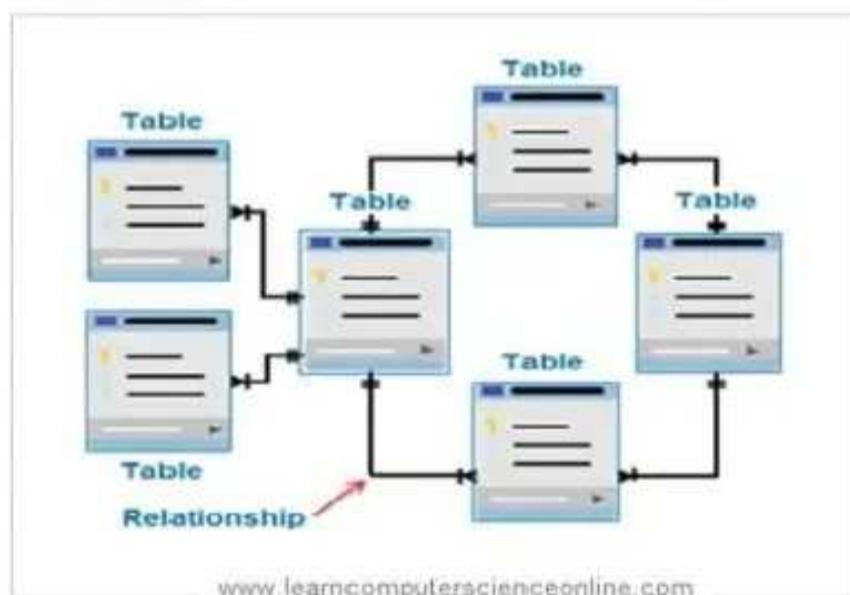
menggabungkan satu tabel dengan tabel lainnya menggunakan kunci yang dimilikinya.

Konsep RDBMS sendiri adalah sistem yang mendukung hubungan atau hubungan antar tabel dalam suatu basis data, RDBMS Merupakan produk yang menyajikan tampilan data sebagai kumpulan baris dan kolom, meskipun tidak hanya didasarkan pada teori basis data relasional.

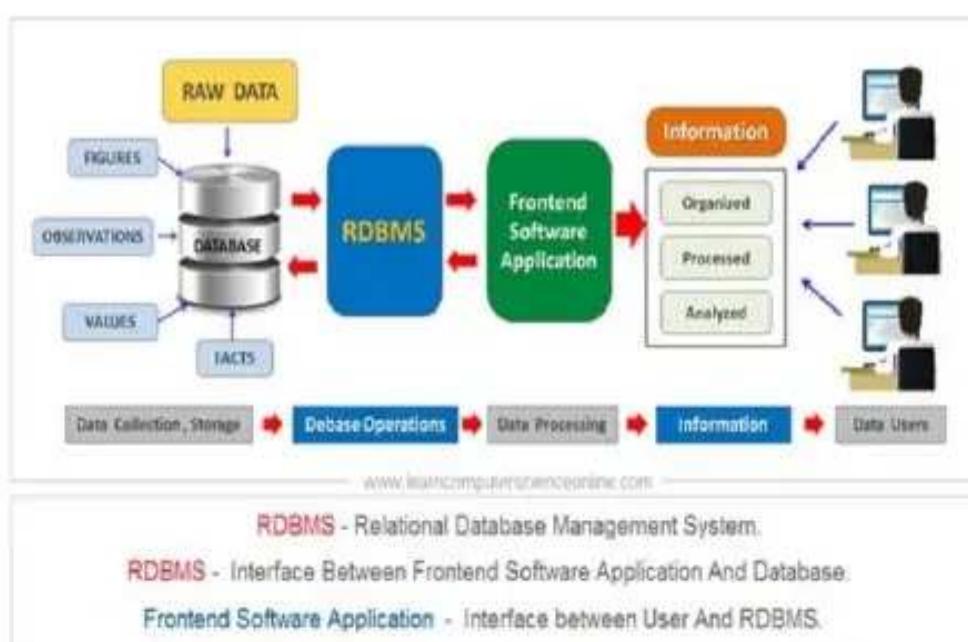
Bahasa Query maupun syntax digunakan sebagai penghubung pengguna model basis data relasional ini bervariasi menurut ketentuan vendor masing-masing, namun yang menjadi defacto dan populer adalah SQL. SQL merupakan bahasa standar ANSI 1986 dan ISO 1987.

Model database Relasional

RDBMS - Relational Database Model



Model RDBMS



Sejarah Sistem Pengelolaan Basis Data Relasional (Relational Database Management System/RDBMS)

Awalnya pada Tahun 1970 Edgar F Codd pada penelitiannya di IBM memperkenalkan istilah Database relational, namun, pada awal implementasinya banyak model relasional yang tidak mengikuti seluruh elemen-elemen yang terdapat dalam 12 hukum Codd, alasan ini menjadikan terminologinya berkembang untuk mendeskripsikan sebuah tipikal sistem basis data yang lebih luas. Dalam cakupan minimum dari sistem tersebut adalah mampu memenuhi kriteria penyajian data pada pengguna dalam bentuk relasional atau penampilan dalam bentuk tabular, sebagai koleksi dari tabel dimana setiap tabel berisi sekumpulan baris dan kolom dan penyedian operator relasional dalam memanipulasi data yang berbentuk tabular.

Sistem Basis Data yang pertama kali yang secara relatif memenuhi terapan sebuah model relasional adalah Pusat Studi Ilmiah IB, Inggris, Peterlee; IS1 (1970-1972) dan implementasi mengikutinya PRTV (1973-1979). Sistem pertama kali yang didistribusikan secara komersial sebagai RDBMS adalah Multics Relational Data Store (Store atau store) pada tahun 1978. kemudian adalah Berkeley Ingres QUEL dan IBM BS12. (Wikipedia)

Perangkat Lunak RDBMS

Berikut ini merupakan Contoh maupun Jenis dari sistem pengelola basis data yang mendukung model relasional antara lain seperti:

1. Perangkat Lunak RDBMS bersifat komersial antara lain: FileMaker Pro, Dataphor, DB2, FrontBase, 4th Dimension, Informix, Daffodil database, Sand Analytic Server (Nucleus), SmallSQL, Progress 4GL, VMDS, Sybase Adaptive Server Anywhere (Watcom SQL) Netezza, NonStop SQL, Oracle, Teradata, VistaDB, Sybase Adaptive Server Enterprise, Sybase Adaptive Server IQ, InterBase, Mimer SQL, Microsoft Access, Microsoft SQL Server, Microsoft Visual FoxPro, dsb.

- Perangkat Lunak RDBMS bersifat open source atau gratis antara lain: SQLite, HSQLDB, H2, Cloudscape, Ingres, Firebird, MySQL, MaxDB, Derby, tdbengine, MonetDB, PostgreSQL, dsb.

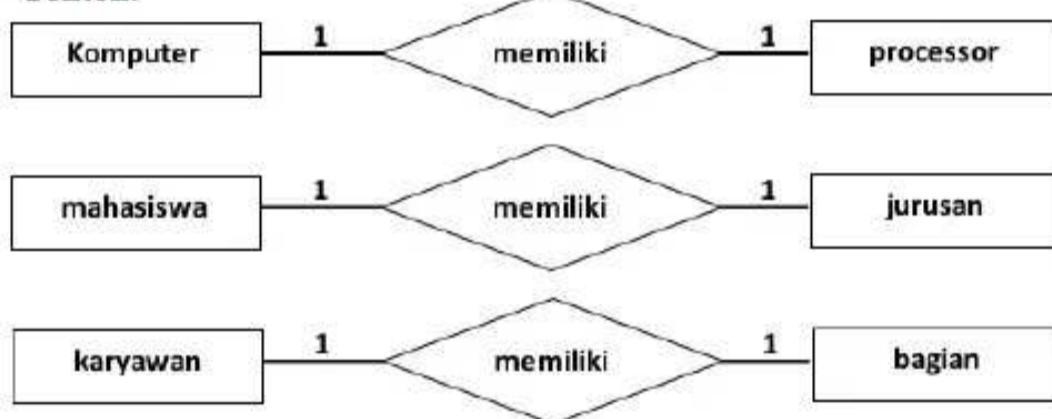
Jenis Relasi Tabel

Ada tiga jenis relasi basis data yang dikelola sistem dbms, diantaranya:

- One-To-One (Satu ke Satu)** adalah dimana setiap sebuah entitas pada tabel dapat berrelasi dengan satu entitas tabel lainnya pada basis data, berlaku timbal balik.

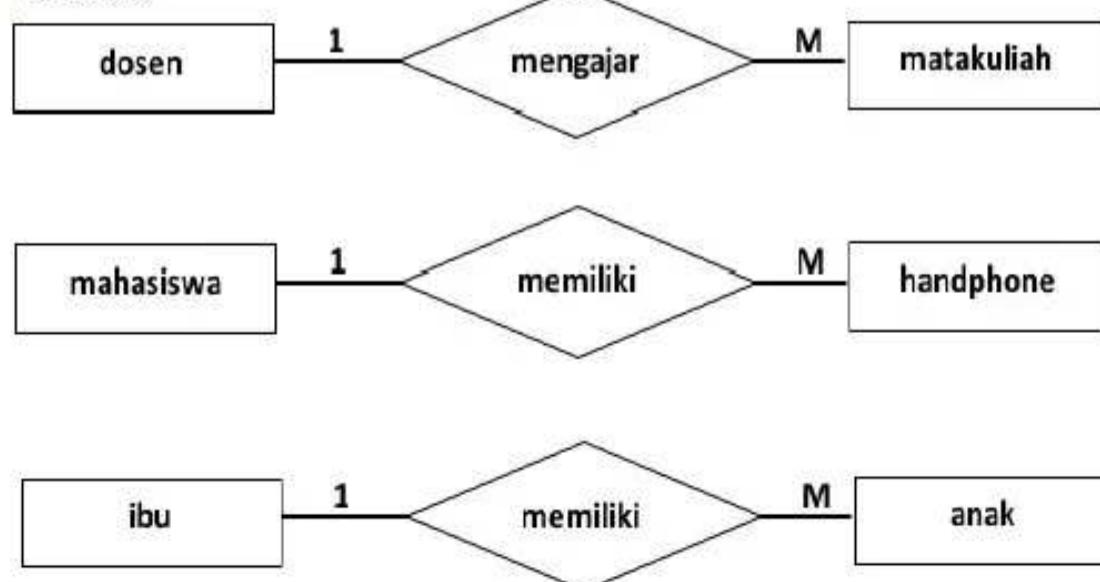
Contoh:

Contoh:



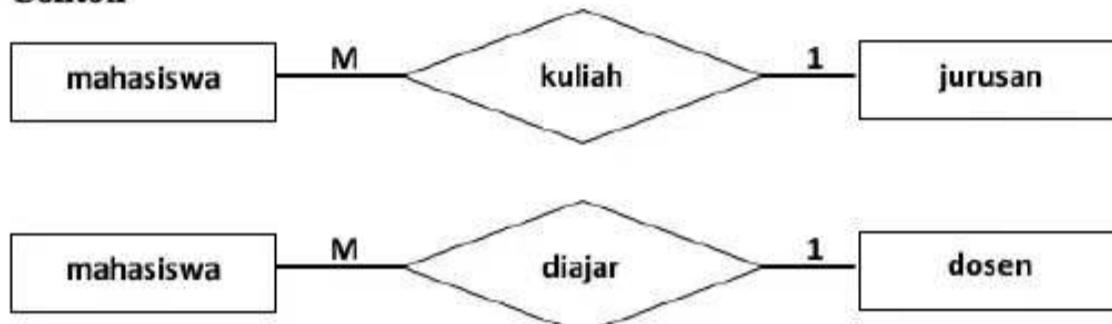
- One-to-Many (Satu ke Banyak)** adalah dimana setiap sebuah entitas tabel dapat berhubungan atau berrelasi dengan banyak entitas tabel lainnya pada basis data, namun tidak sebaliknya.

Contoh:



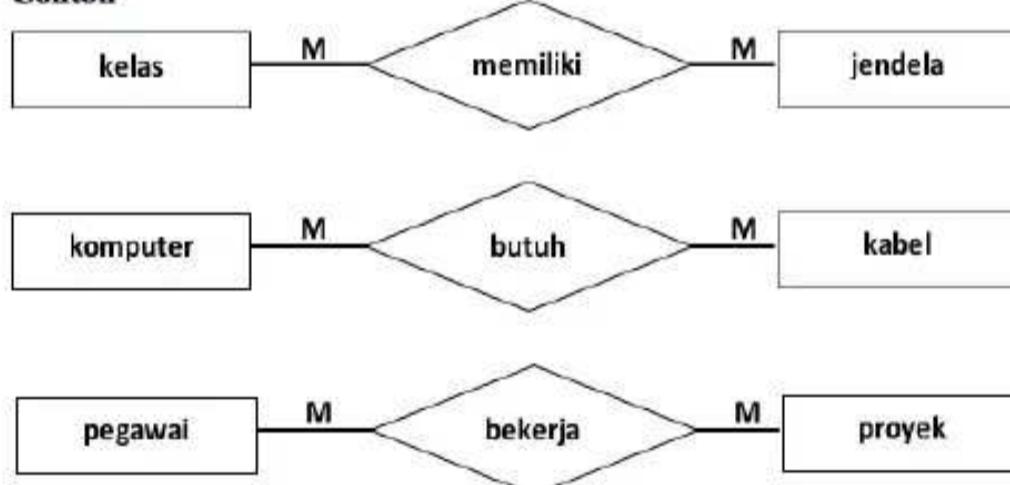
3. **Many-to-One (banyak ke satu)** adalah dimana setiap entitas tabel dapat berhubungan dengan satu entitas tabel lainnya dalam database, namun tidak sebaliknya.

Contoh



4. **Many-to-Many (Banyak ke Banyak)** adalah dimana setiap entitas tabel dapat berelasi dengan banyak entitas tabel lainnya pada basis data dan berlaku sebaliknya.

Contoh



Fungsi RDBMS

Pada dasarnya RDBMS (*Rational Database Management System*) mengelola basis data tipe rasional. Basis data rasional sendiri sebenarnya mengacu pada penyimpanan data yang terstruktur dalam bentuk tabel menggunakan baris dan kolom, dan RDBMS sendiri mengacu pada sistem bagaimana data dalam database tetap konsisten.

RDBMS memiliki fungsi dasar membaca data yang terdapat dalam database, memperbarui struktur basis data, membuat beberapa tabel dalam satu basis data, dan menghapus struktur yang tidak lagi digunakan dalam data. Atau dengan kata lain RDBMS memiliki fungsi dasar dengan istilah CRUD (**Create, Read, Update, dan Delete**). Secara umum, RDBMS akan menyediakan kamus

data dan metadata yang dapat digunakan untuk menangani data.

Ini adalah alasan mengapa RDBMS memiliki fungsi untuk membangun hubungan atau hubungan data antar tabel sehingga dapat didefinisikan dengan baik. Hubungan dapat terjadi karena pembentukan kunci. Kunci tersebut disebut kunci utama (*primary Key*) di tabel pertama dan dikaitkan dengan tabel kedua yang memiliki kunci tamu atau kunci asing. Selain itu RDBMS juga dapat mencegah duplikasi data atau ada pengulangan data. RDBMS juga dapat digunakan untuk membangun database yang kompleks.

Perbedaan RDBMS dengan DBMS

Perbedaan utama antara RDBMS dan DBMS adalah RDBMS menyimpan data dalam bentuk tabel yang terdiri dari kolom dan baris. Secara umum, data konsep RDBMS akan disajikan dalam bentuk model hierarkis atau jaringan. Ini adalah salah satu keterbatasan model hierarkis karena tidak dapat mengakomodasi banyak masalah dalam database. Berbeda dengan RDBMS yang memiliki kunci identifikasi yang disebut kunci primer.

Nilai kunci utama disimpan dalam sebuah tabel. Oleh karena itu data akan mudah diakses dan diperbarui oleh sistem. Tidak hanya itu, RDBMS juga dapat menangani banyak masalah logis dalam database, dengan kunci utama dalam tabel dapat mencegah redundansi data. Selain itu RDBMS juga menyediakan proses normalisasi.

Berbeda dengan DBMS yang tidak menyediakan proses normalisasi karena tidak ada kunci yang membedakannya. RDBMS dibuat khusus agar menangani data yang memiliki ukuran besar dan mempunyai banyak user. Sedangkan untuk DBMS hanya dapat menangani data dengan ukuran kecil dan jumlah penggunaan terbatas, untuk meningkatkan integritas basis data, RDBMS mengimplementasikan sistem keamanan yang sering disebut sebagai ACID (*Atomicity, Consistency, Isolasi, Durability*).

Kelebihan dari RDBMS

Berikut ini adalah kelebihan dari *Relational Database Management System* (RDBMS):

1. Keandalan dijamin dan diuji
2. Tidak ada kesulitan dalam menemukan SDM yang ahli dalam pengembangan basis data
3. Membutuhkan investasi yang lebih kecil jika dibandingkan dengan DBMS non-relational
4. Sesuai untuk struktur database yang komplek dan terstruktur.

Kekurangan dari RDBMS

Berikut ini adalah kekurangan dari *Relational Database Management System* (RDBMS):

1. Tidak sesuai/tidak cocok untuk data besar yang tidak tersusun seperti *Big data / Cloud database*
2. Skema data diperbaiki sesuai dengan struktur DBMS yang bersangkutan (relevan)
3. Membutuhkan skema tertentu jika diterapkan dalam basisdata distribusi

Daftar Pustaka

<https://www.nesabamedia.com/pengertian-rdbms>

https://books.google.co.id/books?hl=id&lr=&id=hKdADwAAQBAJ&oi=fnd&pg=PR5&dq=Jurnal+tentang+basis+data&ots=cpDfBTIG7&sig=zfJ0--_esc=y#v=onepage&q&f=false

Rahmadya Trias Handayanto, ST.,M.Kom.,Ph.D dan Herlawati, S.Si.,M.M.,M.Kom; Pemrograman Basis Data di Matlab dengan MySQL dan Microsoft Access, Penerbit Informatika

<https://id.wikipedia.org/wiki/Sistem>

<https://ruangguru.co/author/yolanda-agustina/>

<https://www.bakhel.com/2020/11/pengertian-sejarah-contoh-jenis-relasi-rdbms.html>

<https://www.learncomputerscienceonline.com/database-management-system/>

Setyawati, E., S., WIJOYO, H., & Soeharmoko, N. (2020, November 17). RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS). <https://doi.org/10.17605/OSF.IO/JWSHN>

Profil Penulis



Djamaludin

Penulis dilahirkan di kota Brebes, tertarik dengan computer Ketika menyelesaikan studi di SMA 1990 an dengan mengikuti kursus-kursus microsof office, dan dilanjutkan Ketika menyelesaikan studi pada perguruan tinggi profesi dengan mengambil jurusan Informatika Komputer, setelah menyelesaikan Pendidikan Diplomanya penulis lebih sering bekerja dibidang desain dan web, disela-sela bekerja penulis juga menjadi salah satu tim pengajar pada perguruan tinggi profesi, saat ini penulis masih aktif sebagai salah satu dosen tetap di perguruan tinggi swasta diwilayah Tangerang Banten, pada program studi Teknik informatika, setelah lepas sebagai Pekerja profesi dibidang IT sepuluh tahun yang lalu, Penulis menyelesaikan Studi S1 di Program Studi Sistem Informasi STMIK PGRI Tangerang dan pada tahun 2014 penulis menyelesaikan Studi Strata 2 (Magister) di prodi Program Pascasarjana Ilmu Komputer Universitas Budiluhur Jakarta.

Email Penulis : djamaludin@unis.ac.id

Email Pribadi : dsn.bjems@gmail.com

PENGENALAN DATABASE 2

Nurul Aini, S.Kom., M.T.

Universitas Dipa Makassar

Pengetian MySQL

MySQL (*My Structured Query Language*) sebuah perangkat lunak atau aplikasi dalam manajemen basis data SQL yang bersifat open source. MySQL termasuk dalam relasi manajemen database sistem atau biasa disebut RDBMS (*Relational Database Management System*), sehingga penggunaan tabel, kolom, baris didalam struktur databasenya, proses pengambilan data menggunakan metode relational database dan juga sebagai perantara aplikasi dengan database server dari aplikasinya. Ditahun 1994 Michael Widenius bersama David Axmark dan Allan Larsson mendirikan MySQL , kata awal "My" diadopsi dari nama anak perempuan Michael, untuk logo lumba-lumba diambil dari sebuah acara "Name the Dolphin" yang diadakan oleh MySQL AB itu sendiri (Solichin, 2010). Pada 1995 MySQL dirilis versi pertama yang diperuntukkan pengguna pribadi dibawah lisensi GNU GPL (*General Public License*) secara gratis walau ada ketentuan, selama aplikasi yang dikembangkan menggunakan database MySQL juga nantinya disediakan secara gratis dibawah lisensi GNU GPL, namun jika aplikasi yang dikembangkan untuk tujuan komersil (bisnis) maka pengembang harus membayar royalti. MySQL dibagi menjadi dua lisensi, pertama adalah Free Software dimana perangkat lunak dapat diakses oleh siapa saja dan yang kedua adalah Shareware dimana perangkat lunak berpemilik memiliki batasan dalam

penggunaannya. MySQL dapat didownload melalui situs resminya <http://www.mysql.com>.



Gambar 1. Logo MySQL

MySQL tersedia untuk beberapa platform seperti diantaranya Linux dan Windows. Untuk proses akses secara mudah dapat menggunakan software tertentu seperti phpMyAdmin, MySQL Yog, Xampp dan lainnya.

Fungsi MySQL

Fungsi secara umum dari MySQL adalah mengelola dan membuat database sistem pada bagian server yang berisi berbagai informasi dengan menggunakan perintah dasar dari SQL. Adapun fungsi lain yang dimiliki adalah kemudahan user dalam mengakses data berisi informasi dalam bentuk String (teks), yang dapat diakses secara personal maupun publik dalam sistem yang berbasis web. Disaat ini banyak penyedia server web atau host menyediakan fasilitas untuk MySQL dalam pengembangan aplikasi berbasis website untuk dikelola oleh web developer. Kemudian, antarmuka dari MySQL adalah PHPMyAdmin. Yang berfungsi untuk menghubungkan antara bahasa pemrograman PHP dengan MySQL untuk proses pengelolaan basis data pada web.

Kelebihan dan Kekurangan MySQL

MySQL termasuk aplikasi populer karena banyak digunakan oleh berbagai kalangan,

Beberapa kelebihan dari MySQL adalah sebagai berikut:

1. Merupakan salah satu software yang portable

keunggulan yang pertama dimiliki MySQL adalah jenis software yang portable, Software portable ini berarti MySQL bisa dijalankan untuk mengolah database multi platform. Seperti Windows, Linux, Mac, dan sebagainya bisa menggunakan MySQL dalam konsep DBMS, sehingga hal ini membuat MySQL menjadi lebih baik dari segi efisiensi dan juga fungsionalitas yang lebih baik.

2. MySQL merupakan salah satu DBMS yang opensource

MySQL dengan versi paling basic atau sederhana dijual dengan harga yang gratis, karena merupakan software Open source. Namun demikian, meskipun merupakan software opensource, MySQL sudah memiliki lisensi GPL. MySQL menyediakan versi enterprise untuk membutuhkan fungsi yang lebih dengan harga yang lebih terjangkau.

3. Multi-User

Sama seperti program DBMS lainnya, meskipun merupakan software yang open source, MySQL memiliki kemampuan yang sangat baik untuk mendukung kepentingan multiuser, dimana bisa dijalankan oleh banyak user dalam satu waktu tanpa perlu mengalami kendala seperti crash, dan semacamnya.

4. Tipe Data Bervariasi

MySQL menawarkan tipe data yang bervariasi diantaranya integer, float, double, char, text, date, timestamp dan masih banyak lagi. Hal ini menjadi salah satu keunggulan dalam kebutuhan DBMS.

5. Segi Keamanan yang Baik

Kelebihan lainnya dari MySQL adalah fitur keamanannya yang cukup baik, apalagi dengan statusnya yang open source, fitur keamanan yang ditawarkan oleh software ini sudah sangat diandalkan.

6. Administrative tools yang lengkap

Administrative tools yang terdapat di dalam software ini pun sudah terbilang lengkap. User dan juga programmer dapat menggunakan MySQL dengan mudah, tanpa perlu harus repot – repot mempelajari MySQL secara detil.

7. Struktur tabel yang lebih fleksibel

Struktur data yang dimiliki oleh MySQL juga dinilai lebih fleksibel dan juga mudah untuk digunakan. Hal ini terutama untuk menangani table berupa ALter Table.

8. Dapat diintegrasikan dengan berbagai bahasa pemrograman

MySQL juga dapat diintegrasikan dengan berbagai macam bahasa pemrograman yang ada. Dengan begitu, MySQL bisa membantu pembangunan dari sebuah sistem dengan mudah dan juga efektif, karena dapat terintegrasi dengan berbagai macam bahasa pemrograman standar yang bisa digunakan dalam pembangunan suatu sistem.

9. Tidak membutuhkan spesifikasi hardware yang tinggi

Salah satu hal penting yang menarik yang ada pada MySQL adalah spesifikasi. Untuk dapat menjalankan program MySQL ini, maka tidak dibutuhkan

spesifikasi minimal komputer yang tinggi sehingga PC ataupun laptop sekali pun masih bisa menggunakan software MySQL ini dengan baik tanpa menemui kendala dan masalah mengenai spesifikasinya.

10. RAM Kecil dapat menggunakannya

DBMS yang satu ini memiliki kelebihan yaitu dapat di install di ram yang relatif kecil bila di bandingkan dengan database lain.

Meskipun memiliki banyak kelebihan, terutama karena merupakan salah satu program atau software yang

opensource, ternyata MySQL juga memiliki beberapa kekurangan. Berikut ini adalah beberapa kekurangan MySQL:

1. Sulit untuk diaplikasikan pada instansi atau perusahaan dengan database yang besar

Karena merupakan salah satu jenis DBMS yang ramah terhadap spesifikasi komputer, maka MySQL pun memiliki fitur yang tidak lengkap Oracle. Hal ini berhubungan dengan implementasi dari DBMS yang dilakukan, dimana MySQL tidak mampu atau diragukan kemampuannya untuk melakukan manajemen database dengan jumlah data yang sangat besar. Sehingga tidak cocok untuk diterapkan pada instansi atau perusahaan besar.

2. Support yang kurang

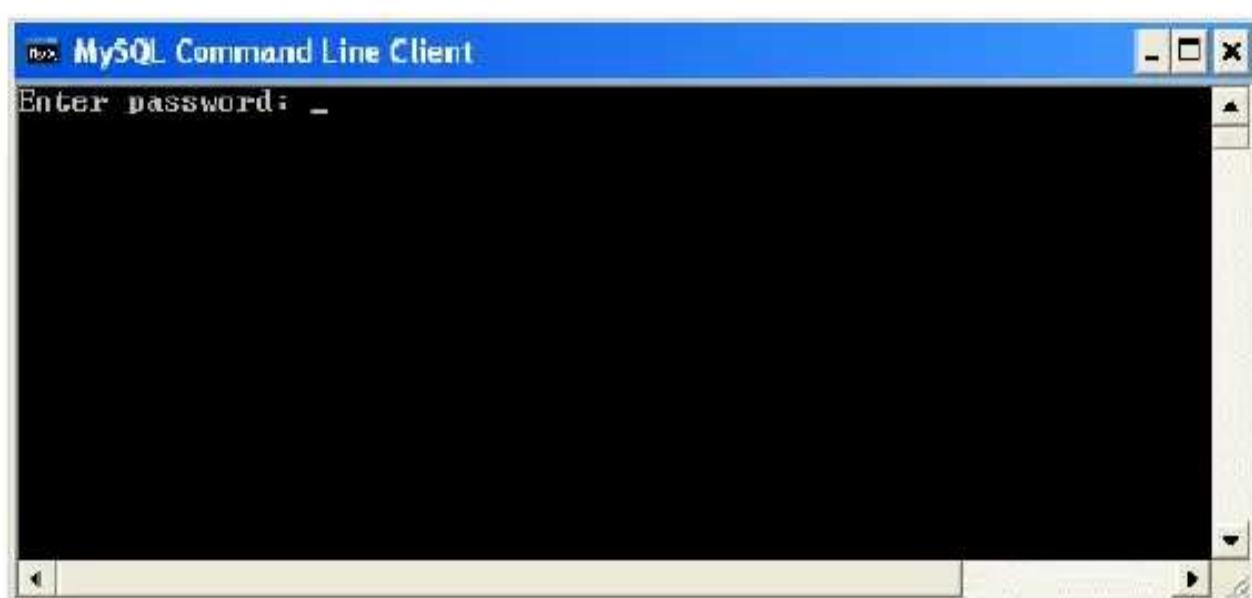
Technical support dari MySQL juga dianggap kurang baik. Hal ini mungkin berhubungan dengan status open source yang dimiliki oleh MySQL. Hal ini membuat user akan mengalami kesulitan dalam menghubungi technical support dari MySQL ketika dihadapkan pada suatu kendala atau permasalahan saat menggunakan software ini.

3. Tidak populer dikalangan aplikasi game

DBMS mysql ini sangat kurang digunakan untuk aplikasi Game. Jadi jika anda ingin mengembangkan dua jenis aplikasi ini, MySQL bukan teman yang tepat.

Akses Ke MySQL

MySQL menyediakan tools untuk dapat mengakses ke server MySQL, yaitu melalui command line. Dapat diakses melalui menu **Start > All Program > MySQL Server 5 > MySQL Command Line Client.**



Gambar 2. Akses MySQL melalui Command Line

Password koneksi diset pada saat proses pemasangan, jika password benar, maka akan ditampilkan seperti berikut:



Gambar 3. Koneksi berhasil ke MySQL

Setelah koneksi berhasil ke server, maka akan ditampilkan prompt mysql>, dimana baris perintah SQL dapat diisikan pada prompt tersebut dan setiap akhir baris perintah ditutup dengan titik-koma (;).

Selain melalui command line MySQL dapat diakses melalui software pendukung pemrograman bahasa PHP yang telah terintegrasi dengan MySQL, seperti menggunakan aplikasi Xampp, pada control panel Xampp dengan cara mudah anda dapat mengakses MySQL server melalui shell atau phpMyAdmin dengan mengaktifkan start service pada MySQL. Akses melalui shell pada control panel dengan perintah mysql -u (username) -p

(password). Apabila user MySQL masih default maka dapat menggunakan perintah berikut mysql -u root.

```

Setting environment for using XAMPP for Windows.
HP@DESKTOP-ELQ052I c:\xampp
# mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

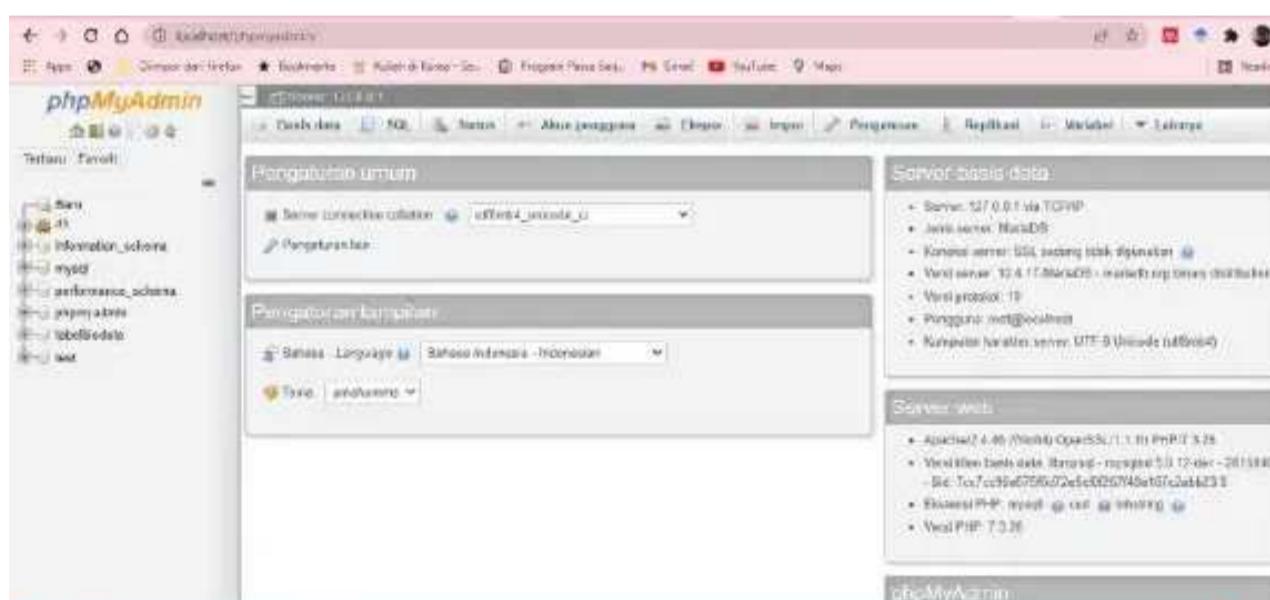
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> .

```

Gambar 4. Login MySQL melalui shell Xampp

Selain command prompt (shell), MySQL juga dapat diakses melalui phpMyAdmin yang menawarkan proses manajemen database berbasis GUI (*Graphical User Interface*), melalui ini banyak sekali pemula belajar dan mengenal MySQL tanpa perlu mengetikkan baris perintah SQL.



Gambar 5. Tampilan phpMyAdmin

Membuat Database di MySQL

Membangun sebuah database merupakan tahap awal yang harus dilakukan dalam membuat sebuah sistem atau aplikasi berbasis database, desain database yang baik menunjukkan seberapa baik sebuah sistem yang dibuat (Solichin, 2016; Tim, 2014). Berikut adalah

beberapa contoh pengoperasiaoan MySQL yang dapat digunakan.

1. Membuat database:

Dibuku ini contoh penggunaan database MySQL dituliskan dalam perintah SQL di command line. Pembuatan suatu database dapat dilakukan dengan perintah berikut:

`CREATE DATABASE nama_database;`

Contoh: `mysql>create database pegawai;`

Untuk melihat apakah perintah tersebut sudah diberhasil dilakukan, dapat dilakukan dengan perintah berikut:

`mysql>show databases;`

Database
db_contact
db_member
information_schema
mysql
pegawai
performance_schema
phpmyadmin
tabelbiodata
test

2. Membuat tabel

Untuk membuat tabel dalam database pegawai, digunakan perintah sebagai berikut:

`CREATE TABLE nama_tabel (defenisi tabel);`

Contoh

`mysql>create table pegawai (noPegawai int not null auto_increment, nama varchar (50), pekerjaan varchar (30), kodeDepart int not null, primary key (noPegawai));`

```
MariaDB [pegawai]> create table pegawai (noPegawai int not null auto_increment, nama varchar(50), pekerjaan varchar(30),
kodeDepart int not null, primary key(noPegawai));
Query OK, 0 rows affected (0.036 sec)

MariaDB [pegawai]> show tables
->;
+-----+
| Tables_in_pegawai |
+-----+
| pegawai |
+-----+
1 row in set (0.003 sec)
```

Untuk melihat struktur tabel yang telah dibuat dapat menggunakan perintah berikut.

DESCRIBE nama_tabel;

contoh: mysql>describe pegawai;

```
MariaDB [pegawai]> describe pegawai;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| noPegawai | int(11) | NO  | PRI | NULL    | auto_increment |
| nama       | varchar(50) | YES |     | NULL    |             |
| pekerjaan  | varchar(30) | YES |     | NULL    |             |
| kodeDepart | int(11)  | NO  |     | NULL    |             |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.018 sec)
```

3. Mengisi tabel data (*Insert*)

Untuk dapat mengisi data ke dalam tabel, dapat diberikan perintah sebagai berikut:

INSERT INTO nama_tabel VALUES (isi data sesuai urutan field pada tabel);

contoh:

mysql>insert into pegawai values (0, "Andi", "Keuangan", 205);

```
MariaDB [pegawai]> insert into pegawai values(0,"Andi","Keuangan",205);
Query OK, 1 row affected (0.012 sec)
```

```
MariaDB [pegawai]> Select * From pegawai;
+-----+-----+-----+-----+
| noPegawai | nama | pekerjaan | kodeDepart |
+-----+-----+-----+-----+
|          2 | Andi | Keuangan  |        205 |
+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

4. Mengubah Data (*Update*)

Data yang telah berhasil tersimpan didalam tabel dapat diubah, jika terjadi kesalahan pengisian data. Untuk dapat mengubah data ke dalam tabel, dapat diberikan perintah sebagai berikut:

```
UPDATE nama_tabel SET nama_field=data baru
WHERE nama_fields=data;
```

Setelah penulisan SET sebutkan kolom data yang akan disi data disertai data isian, clause WHERE membantu untuk memposisikan pada data yang mana akan dilakukan perubahan. Contoh berikut adalah mengubah data tabel dengan nomor pegawai “2” dengan mengubah nama Andi dengan Dahlan pada tabel pegawai.

```
MariaDB [pegawai]> Select * From pegawai;
+-----+-----+-----+-----+
| noPegawai | nama | pekerjaan | kodeDepart |
+-----+-----+-----+-----+
| 2 | Andi | Keuangan | 205 |
| 3 | Risma | Keuangan | 205 |
| 4 | Abdul | Promosi | 208 |
| 5 | Rahmat | Promosi | 208 |
| 6 | Ayudi | Sekretaris | 201 |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)
```

contoh:

```
mysql>update pegawai set nama= "Dahlan" where
noPegawai=2;
```

```
MariaDB [pegawai]> update pegawai set nama="Dahlan" where noPegawai=2;
Query OK, 1 row affected (0.010 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [pegawai]> Select * From pegawai;
+-----+-----+-----+-----+
| noPegawai | nama | pekerjaan | kodeDepart |
+-----+-----+-----+-----+
| 2 | Dahlan | Keuangan | 205 |
| 3 | Risma | Keuangan | 205 |
| 4 | Abdul | Promosi | 208 |
| 5 | Rahmat | Promosi | 208 |
| 6 | Ayudi | Sekretaris | 201 |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)
```

5. Menghapus Data (*Delete*)

Apabila data yang telah tersimpan namun tidak dibutuhkan lagi atau terdapat kesalahan dalam 1 baris data, maka data tersebut dapat dihapus. Untuk dapat menghapus data didalam tabel, dapat diberikan perintah sebagai berikut:

`DELETE FROM nama_tabel WHERE nama_field=;`

contoh:

`mysql> delete from pegawai where noPegawai=2;`

```
MariaDB [pegawai]> delete from pegawai where noPegawai=2;
Query OK, 1 row affected (0.008 sec)

MariaDB [pegawai]> Select * From pegawai;
+-----+-----+-----+-----+
| noPegawai | nama    | pekerjaan | kodeDepart |
+-----+-----+-----+-----+
|      3   | Risma   | Keuangan  |      205  |
|      4   | Abdul   | Promosi   |      208  |
|      5   | Rahmat  | Promosi   |      208  |
|      6   | Ayudi   | Sekretaris |      201  |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

Daftar Pustaka

Penulisan daftar pustaka menggunakan format APA Edisi-7 atau 6. Contoh:

Solichin, A. (2010). *MySQL5: Dari Pemula Hingga Mahir*. Achmad Solichin.

Solichin, A. (2016). *Pemrograman web dengan PHP dan MySQL*. Penerbit Budi Luhur.

Tim, E. M. S. (2014). *Teori dan praktik PHP-MySQL untuk Pemula*. Elex Media Komputindo.

Profil Penulis

Nurul Aini



Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2005 silam. Hal tersebut membuat penulis lanjut ke perguruan tinggi di UNIVERSITAS DIPA MAKASSAR dengan memilih Jurusan SISTEM INFORMASI dan berhasil lulus pada tahun 2009. Penulis kemudian melanjutkan pendidikan ke studi S2 di prodi TEKNIK INFORMATIKA PROGRAM PASCA SARJANA UNIVERSITAS HASANUDDIN dan menyelesaikan studi di tahun 2013.

Penulis memiliki kepakaran dibidang Decision Support System (DSS) dan Data Science. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI. Selain peneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini. Atas dedikasi dan kerja keras dalam menulis buku, Perpustakaan Nasional RI memberikan penghargaan sebagai salah satu Pemenang Buku Terbaik Tahun 2022.

Email Penulis: nurulaini.m11@undipa.ac.id

10

PENGENALAN ORACLE

Wiyanto, S.Kom., M.Kom.

Program Studi Teknik Informatika

Universitas Pelita Bangsa

Pengenalan Oracle

Basis Data *Oracle* adalah sebuah *Database Management System (DBMS)* untuk mengelola informasi secara terbuka, komprehensif dan terintegrasi yang dapat memanipulasi database. *Oracle server* menyediakan solusi yang efisien dan efektif karena kemampuannya dalam berbagai hal, dapat bekerja di lingkungan *client/server* (pemrosesan tersebar), menangani manajemen *space* dan basis data yang besar, mendukung akses data secara simultan, performansi pemrosesan transaksi yang tinggi, menjamin ketersediaan yang terkontrol dan lingkungan yang terreplikasi. (Indrajani, 2011)

Dalam bab ini memberikan pengenalan, landasan, konsep dan teknologi yang membentuk dasar dari *Oracle Database Server*, saat ini dikenal sebagai *Oracle Basis Data*. Bab ini memberikan pengenalan *Database Oracle* dari pemula hingga yang berpengalaman. Harapan kami bahwa setelah anda memiliki pemahaman dasar tentang produk ini, banyak buku-buku dan publikasi yang menggambarkan database *Oracle* ini. *Oracle* juga menawarkan Server Aplikasi dan *Fusion Middleware*, intelijen alat bisnis, dan aplikasi bisnis (*E-Business Suite*, *PeopleSoft*, *JD Edwards*, *Siebel*, *Hyperion*, dan *Fusion*, antara lain) (Greenwald, Stackowiak, & Stern, 2013).

Selama lebih dari 30 tahun terakhir, Oracle tumbuh dengan pesat dari banyak vendor yang berkembang dan menjual produk database untuk diakui secara luas sebagai pemimpin pasar database. Meskipun produk awal adalah tipikal perusahaan startup, *Oracle Database* tumbuh sedemikian rupa sehingga kemampuan teknisnya sekarang sering dipandang sebagai yang paling canggih di industri. Dengan setiap rilis database, Oracle telah meningkatkan skalabilitas, fungsionalitas, dan pengelolaan database. (Greenwald, Stackowiak, & Stern, 2013)

Sejarah Oracle

Oracle didirikan pada tahun 1977 oleh tiga orang programmer yang bernama Bob Miner, Ed Oates dan Larry Ellison yang kemudian menjabat sebagai CEO. Oracle pertama kali menyediakan sistem database yang dapat digunakan untuk kebijakan konvensional (Greenwald, Stackowiak, & Stern, 2013)

Selama berdiri sampai sekarang, Oracle telah terbukti dapat membangun masa depan di atas dasar inovasi dan pengetahuan mendalam untuk memenuhi tantangan pelanggan dan keberhasilan yang dianalisis oleh para pemikir bisnis kelas atas. Sekarang *Oracle* dapat ditemukan dihampir setiap industri dan pusat data. *Oracle* adalah perusahaan pertama untuk mengembangkan dan menyebarkan seratus persen *internet-enabled software enterprise* diseluruh lini produk database, aplikasi bisnis, pengembangan aplikasi dan alat pendukung keputusan.

Inovasi adalah mesin kesuksesan *Oracle*. *Oracle* merupakan salah satu perusahaan yang pertama untuk membuat aplikasi bisnis yang tersedia melalui ide *internet*. Oracle telah memperkenalkan produk baru *Oracle Fusion Middleware* dan fungsi onalitas yang mencerminkan tujuan-tujuan perusahaan untuk menghubungkan semua tingkat teknologi perusahaan, memastikan pelanggan mendapatkan akses pengetahuan yang dibutuhkan untuk merespon kondisi pasar dengan cepat dan leluasa.

Empat puluh tahun lebih *Oracle* berjaya, *Oracle* menjadi patokan untuk database teknologi dan aplikasi-aplikasi di dalam perusahaan diseluruh dunia, dari mulai perusahaan-perusahaan yang kecil sampai perusahaan-perusahaan yang besar multinasional dan internasional.

Perkembangan Oracle

Perkembangan Oracle luar biasa pesat, dari masa ke masa terus berkembang (Infoclutch, 2017) (Binus, 2018), berikut sejarah perkembangan Oracle:

1977 Oracle didirikan pertama kali

Untuk perkembangan Oracle dari tahun 1978 – 2005 dapat anda baca dilaman berikut:

<https://www.infoclutch.com/infographic/oracle-corporation-company-history-timeline>

2006 Inovasi berlanjut dengan *Oracle Database Express Edition 10g Release 2* untuk hal ini pengembangan dan *development system*. (Susanto, 2012)

2007 *Oracle Database 11g Release 1* untuk hal ini pengembangan dan *development system*. (Susanto, 2012)

2009 *Oracle Database 11g Release 2* untuk menurunkan biaya TI dan memberikan kualitas layanan yang lebih tinggi.

2013 Oracle Database 12c Release 1, Arsitektur multitenant, *In-Memory Column Store*, *Native JSON*, *SQL Pattern Matching*, *Database Cloud Service*. (Deveci, 2019)

2016 Oracle Database 12c Release 2, *Native Sharding*, Alat Pemulihan Nol Data Loss, Layanan *Cloud Exadata*, *Cloud di Pelanggan*. (Deveci, 2019)

2018 Oracle Database 18c, Database Otonom, Pengulangan Data Guard Multi-Instance, Fungsi Tabel Polimorfik, Integrasi Direktori Aktif. (Deveci, 2019)

2019 Oracle Database 19c, Pengindeksan Otomatis, Pengalihan DML Penjaga Data, Tabel Hibrida Terpartisi, Statistik Waktu Nyata + Kueri Khusus Statistik. (Deveci, 2019)

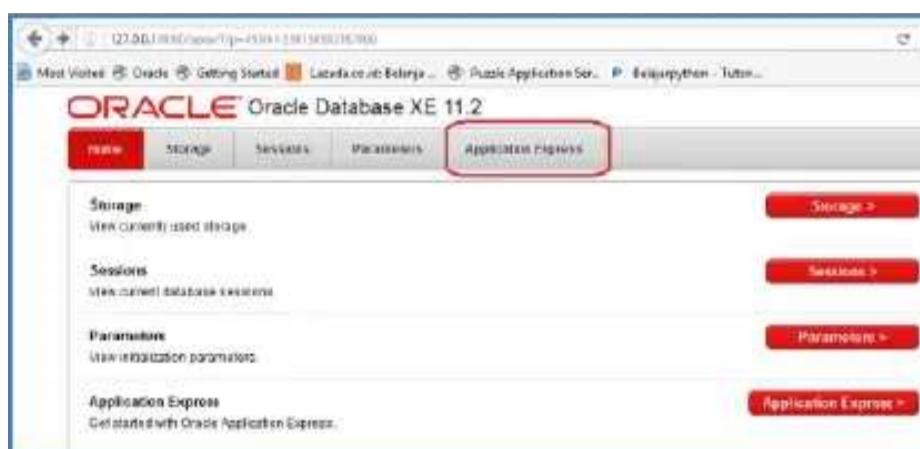
2021 Oracle Database 21c, Oracle. “Ini memberikan kinerja pemrosesan dokumen JSON terkemuka. Ini memberikan kinerja database operasional terobosan dengan dukungan Intel® Optane™ Persistent Memory. Ini memberikan kemampuan database analitik terdepan di industri dengan Penyimpanan Kolom Dalam Memori Pengelolaan Mandiri baru, pemrosesan grafik kinerja tertinggi, dan AutoML untuk pengembangan model pembelajaran mesin paling sederhana. Ini menyediakan Tabel Blockchain yang Tidak Dapat Diubah untuk tabel SQL yang tidak dapat diubah. Vendor yang bersaing memerlukan database dan layanan dokumen JSON, operasional, analitik, grafik, ML, dan blockchain yang terpisah untuk mendukung kemampuan ini. Pendekatan database terkonvergensi Oracle membuat pengembang jauh lebih produktif saat membangun aplikasi baru, dan memudahkan pengembangan aplikasi untuk memenuhi kebutuhan bisnis baru.” (Austin, 2021)

Membuat Basis Data dengan Oracle

Pada bab ini penulis akan mencontohkan dalam pembuatan Basis Data dengan *Oracle XE 11g*, dari contoh pembuatan basis data dengan *Oracle XE 11g* berikut diharapkan pembaca dapat menerapkannya baik dalam pembelajaran.

Berikut langkah dalam pembuatan database *Oracle XE 11g*:

1. Klik kanan/doble klik icon ini pada desktop seperti pada Gambar 10.10 Tampilan Get Started With Oracle Database Di Desktop Komputer diatas, atau ketikan pada browser berikut ini:
<https://127.0.0.1:8080/apex/f?p=4950>
2. Sehingga tampil pada browser anda sebagai berikut:



Gambar 10.1 Tampilan Oracle XE 11g Pada Browser

Klik Tab pada **Application Express** untuk langkah membuat Workspase/Schema

3. Setelah anda Klik pada **Aplication Express**, maka akan tampil **Form Login** sebagai berikut:



Gambar 10.2 Tampilan Login Masuk Oracle

- a. Isikan Username: SYS / SYSTEM
 - b. Password: <<isikan password saat instalasi>>
 - c. Lalu klik tombol **Login**
4. Sehingga tampil **Form Create Application Express Workspace** sebagai berikut:



Gambar 10.3 Tampilan Untuk Membuat Workspace/Schema

Keterangan:

- a. Pada Database User Pilih: Create New
- b. Database Username: <<Isi dengan nama database yang anda inginkan>>
- c. Application Express Username: <<Isi dengan User yang anda inginkan>>
- d. Password: <<isi dengan Password yang anda inginkan>> saran: gunakan kombinasi antar huruf besar, huruf kecil dan angka 8 digit minimal. (yang pasti mudah diingat)
- e. Ulangi Password anda
- f. Klik Tombol Create Workspace

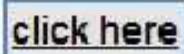


Gambar 10.4 Tampilan Isi Membuat Workspase/Schema

5. Setelah Create Workspace/Schema selesai, sehingga tampil form sebagai berikut:



Gambar 10.5 Tampilan Sukses Dalam Membuat Workspase/Schema

Lalu **click here**  untuk masuk ke menu Login Oracle Application Express database Oracle XE 11g.

6. Masuk ke menu **Login Oracle Application Express**



Gambar 10.6 Tampilan Login Database Oracle

Keterangan:

- Inputkan Workspace/Schema yang baru saja dibuat: DB_WIEY
- Inputkan Username: WIYANTO
- Inputkan Password: *****
- Lalu klik **Login** 

7. Masuk ke menu **Home Oracle Application Express**



Gambar 10.7 Tampilan Menu Utama Database Oracle

8. Bekerja Dengan *Oracle XE 11g*

Dalam bekerja menggunakan *Oracle XE 11g* pada pembuatan basis data terdapat beberapa cara, diantaranya dapat menggunakan Menu **SQL**,

PENGENALAN ORACLE

SQL Scripts. Pada bab ini akan dicontohkan dalam membuat table menggunakan Menu SQL Command. Berikut tampilan menu tersebut:



Gambar 10.8 Tampilan Menu SQL Workshop Pada Basis Data Oracle

Membuat Table Menggunakan Menu SQL Workshop – SQL Commands

Langkah-langkahnya sebagai berikut:

1. Klik menu SQL Workshop pilih SQL Commands, sehingga seperti tampilan berikut:



Gambar 10.9 Tampilan Lembar Kerja SQL Commands

2. Setalah tampil lembar kerja dari SQL Commands diatas, maka kita dapat bekerja dalam membuat table maupun meninputkan data. Berikut cara membuat table menggunakan lembar kerja SQL Commands:
 - a. Membuat Table, pada langkah ini penulis contohkan membuat table Departemen dengan cara mengetikkan SQL pada lembar kerja hal ini disebut dengan *Data Definition Language (DDL)*, seperti pada tampilan berikut:



Gambar 10.10 Tampilan Menuliskan SQL Dalam Membuat Table

- b. Setelah mengetikkan SQL pada lembar kerja diatas, untuk mengeksekusinya blok SQL yang akan di eksekusi, seperti tampilan berikut:

The screenshot shows the Oracle Application Express interface with the 'SQL Workshop' tab selected. The SQL code for creating the 'DEPARTEMEN' table is displayed in the workspace:

```
CREATE TABLE DEPARTEMEN
=====
create table departemen
(
    id_dept varchar (5),
    nama_dept varchar (25),
    lokasi varchar (50),
    constraint pk_departemen_01 primary key (id_dept)
);
```

The code is highlighted in blue. Above the workspace, there are buttons for 'Autocommit' (checked), 'Rows' (set to 10), and 'Save/Run' buttons.

Gambar 10.11 Tampilan Blok SQL Pembuatan Table Departemen

- c. Setelah SQL diblok, lalu klik button Run untuk mengeksekusi pembuatan Table, pada Result akan menampilkan pesan jika Table berhasil dibuat dengan “Table Create”.
- d. Untuk memastikan bahwa table tersebut telah terbuat di dalam database Oracle maka gunakan perintah SQL: Describe <<nama table>>, yaitu: describe departemen / desc departemen, lalu blok SQL tersebut dan klik button Run untuk mengeksekusi seperti pada tampilan berikut:

The screenshot shows the Oracle Application Express interface. In the top navigation bar, 'SQL Workshop' is selected. Below it, the toolbar has 'Autocommit' checked and a 'Run' button highlighted. The main area contains the following SQL code:

```
create table departemen
(
id_dept varchar(5),
nama_dept varchar(25),
lokasi varchar(50),
constraint pk_departemen_01 primary key (id_dept)
);

describe departemen
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Describe' tab is currently selected. A table titled 'Object Type TABLE Object DEPARTEMEN' displays the structure of the 'DEPARTEMEN' table:

Table	Column	Data-type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTEMEN	ID_DEPT	VARCHAR2	5	-	-	1	-	-	-
	NAMA_DEPT	VARCHAR2	25	-	-	-	✓	-	-
	LOKASI	VARCHAR2	50	-	-	-	✓	-	-

Gambar 10.12 Tampilan Structure Table Departemen

- e. Table Departemen sudah berhasil dibuat dengan menggunakan SQL Commands.
3. Saatnya penulis memberikan contoh untuk menginputkan data dengan SQL Command disebut dengan *Data Manipulation Language (DML)*.
- a. Langkah pertama, ketikkan SQL dari DML yang akan diinputkan, dalam hal ini, akan menginputkan data Departemen, terdapat beberapa cara yang dapat digunakan dalam menginputkan data ke dalam table ini, seperti pada tampilan berikut:

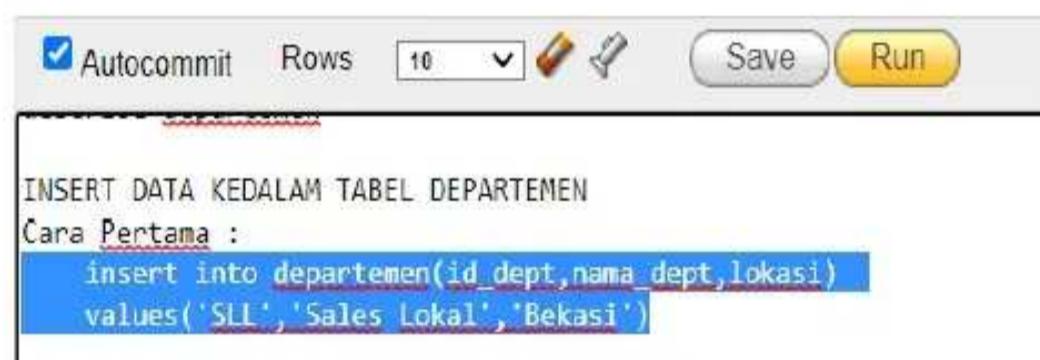
The screenshot shows the Oracle Application Express interface with the 'Run' button highlighted. The main area contains the following SQL code:

```
INSERT DATA KEDALAM TABEL DEPARTEMEN
Cara Pertama :
insert into departemen(id_dept,nama_dept,lokasi)
values('SLL','Sales Lokal','Bekasi')

Cara Kedua :
insert into departemen values('SLE','Sales Export', 'Bekasi')
insert into departemen values('ACT', 'Accounting', 'Bekasi')
insert into departemen values('MKT', 'Marketing', 'Bekasi')
insert into departemen values('PPC', 'PPIC', 'Bekasi')
```

Gambar 10.13 Tampilan DML Departemen

- b. Setelah SQL dari input data departemen diketikan di lembar kerja, terdapat 2 cara yang penulis sajikan pada bab ini seperti yang pada Gambar 10.13 diatas, maka langkah selanjutnya untuk mengeksekusi data tersebut, blok SQL yang akan dieksekusi lalu klik button Run.



```
Autocommit Rows 10 Save Run
INSERT DATA KEDALAM TABEL DEPARTEMEN
Cara Pertama :
insert into departemen(id_dept,nama_dept,lokasi)
values('SLL','Sales Lokal','Bekasi')
```

Gambar 10.14 Tampilan Eksekusi DML Departemen

- c. Maka setelah di eksekusi dengan mengklik button Run, jika SQL yang diketikan benar, akan tampil Result dibawah ini “1 Row(s) Inserted”:
- d. Untuk DML cara kedua, langkahnya seperti diatas hanya berbeda pada penulisan SQL saja.
- e. Sehingga untuk menampilkan data yang didapat dari menginputkan data ke dalam table Departemen secara keseluruhan dengan menggunakan perintah SQL “Select * From Departemen” sebagai berikut:



Select * from Departemen

ID_DEPT	NAMA_DEPT	LOKASI
SLL	Sales Lokal	Bekasi
SLE	Sales Export	Bekasi
ACT	Accounting	Bekasi
MKT	Marketing	Bekasi
PPC	PPIC	Bekasi

5 rows returned in 0.02 seconds Download

Gambar 10.15 Tampilan Data Pada Table Departemen

Demikian pembahasan pada bab ini semoga dapat bermanfaat untuk penulis dan pembaca yang budiman.

Daftar Pustaka

- Austin, A. (2021, January 13). *Oracle Extends Database Leadership with Oracle Database 21c*. Retrieved from Oracle.com:
<https://www.oracle.com/news/announcement/oracle-database-21c-011321/>
- Binus, I. (2018, Desember 12). *SEJARAH PERKEMBANGAN ORACLE*. Retrieved from Binus of Information System:
<https://sis.binus.ac.id/2018/12/12/sejarah-perkembangan-oracle-2/>
- Deveci, M. S. (2019, September 2). *Oracle Database Version History & Oracle Release Versions*. Retrieved from IT Tutorial: <https://ittutorial.org/oracle-version-history-oracle-database-release-versions/>
- Greenwald, R., Stackowiak, R., & Stern, J. (2013). *Oracle Essentials - Oracle Database 12c*. United States: O'Reilly Media.
- Indrajani, I. (2011). *Bedah Kilat 1 Jam Pengantar dan Sistem Basis Data*. Jakarta: PT Elex Media Komputindo.
- Infoclutch, I. (2017, December 20). *A Brief History of Oracle*. Retrieved from Infoclutch:
<https://www.infoclutch.com/infographic/oracle-corporation-company-history-timeline>
- Susanto, B. (2012). *Membangun Sistem Basis Data dengan Oracle XE*. Yogyakarta: CV. Andi Offset.

Profil Penulis



Wiyanto

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2002 silam. Setelah Penulis menyelesaikan Sekolah Menengah Kejuruan dengan Jurusan Mekanik Umum lulus pada tahun 2009, lalu bekerja pada salah satu perusahaan swasta di Cikarang-Bekasi. Setelah bekerja dan menikah penulis memutuskan memilih kuliah dibidang Komputer, dari mulai Diploma-1 Jurusan MANAJEMEN INFORMATIKA, Diploma-3 Jurusan MANAJEMEN INFORMATIKA dan kemudian Penulis melanjutkan pendidikan ke Jenjang Strata-1 SISTEM INFORMASI dan berhasil menyelesaikan pada tahun 2011. Penulis menyelesaikan studi Strata-2 pada Program Studi TEKNOLOGI SISTEM INFORMASI PROGRAM PASCA SARJANA UNIVERSITAS BUDI LUHUR pada Tahun 2014.

Penulis memiliki kepakaran dibidang Programming, Database dan Data Analyst, hal tersebut untuk mewujudkan karir sebagai praktisi dan dosen professional. Saat ini Penulis sebagai Praktisi dan dosen pada Program Studi TEKNIK INFORMATIKA UNIVERSITAS PELITA BANGSA, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI dan aktif juga pada Pengabdian Kepada Masyarakat dan banyak menulis buku-buku untuk Taman Kanak-Kanak.

Email Penulis: wiyanto@pelitabangsa.ac.id

11

KONSEP DASAR MODEL ER

Mohammad Ridwan, S.Kom., M.Kom.

Universitas Islam Syekh Yusuf

Tinjauan Model ER

Model Entity-relationship (model ER) menggambarkan struktur database dengan bantuan diagram yang dikenal sebagai Entity Relationship Diagram (ER Diagram). Model ER adalah desain atau cetak biru database yang nantinya dapat diimplementasikan sebagai database. Model ER telah ada selama lebih dari 35 tahun. Ini sangat cocok untuk pemodelan data untuk digunakan dengan database karena cukup abstrak dan mudah untuk didiskusikan dan dijelaskan. Model ER mudah diterjemahkan ke dalam relasi. Model ER, juga disebut skema ER, diwakili oleh diagram ER.

Pertanyaan yang sering disampaikan oleh pengguna *Database Management Sistem* (DBMS) adalah mengapa menggunakan Diagram ER. Disini, adalah alasan utama untuk menggunakan Diagram ER yaitu:

1. Membantu Anda mendefinisikan istilah yang terkait dengan pemodelan hubungan entitas.
2. Memberikan pratinjau tentang bagaimana semua tabel harus terhubung, bidang apa yang akan ada di setiap tabel.
3. Membantu menggambarkan entitas, atribut, hubungan/relasi.

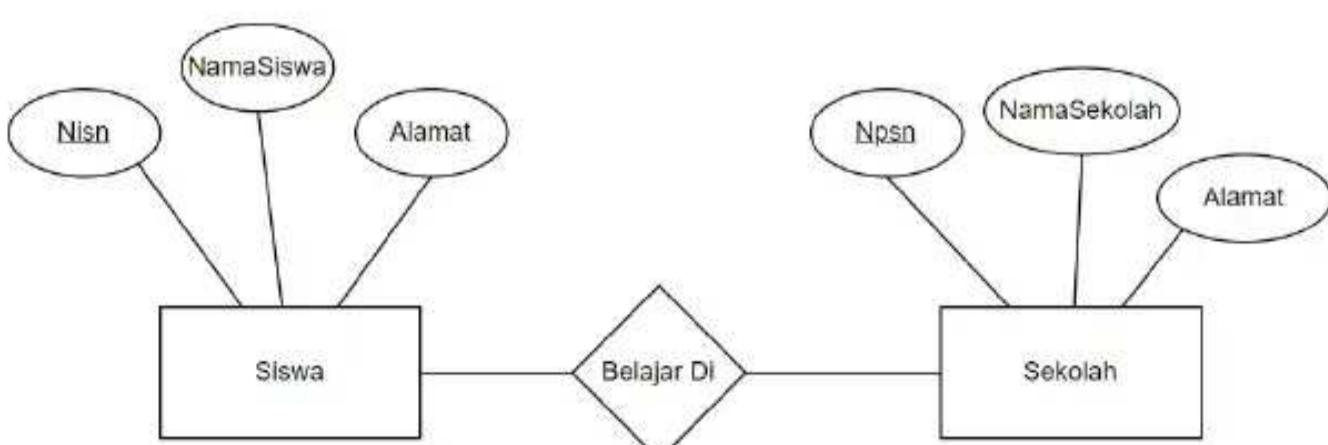
4. Diagram ER dapat diterjemahkan ke dalam tabel relasional yang memungkinkan anda membangun basis data dengan cepat.
5. Diagram ER dapat digunakan oleh perancang basis data sebagai cetak biru untuk mengimplementasikan data dalam aplikasi perangkat lunak tertentu.

Komponen utama dari model ER antara lain:

1. Entitas, didefinisikan sebagai obyek yang menyimpan informasi tertentu (data),
2. Relasi, didefinisikan sebagai asosiasi atau interaksi antara entitas.

Mari kita lihat diagram ER sederhana untuk memahami konsep ini.

Dalam diagram berikut kita memiliki dua entitas Siswa dan Sekolah dan hubungan mereka. Hubungan antara Siswa dan Sekolah adalah banyak banding satu karena sebuah Sekolah dapat memiliki banyak Siswa namun seorang siswa tidak dapat belajar di beberapa Sekolah pada saat yang bersamaan. Entitas Siswa memiliki atribut seperti Nisn, NamaSiswa & Alamat dan entitas Sekolah memiliki atribut seperti Npsn, NamaSekolah, Alamat. Kasus diatas bias digambarkan menjadi Model ER berikut:

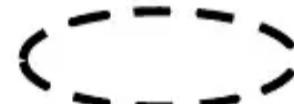


Gambar 1. Contoh Model ER

Berikut adalah bentuk-bentuk geometris dan artinya dalam Diagram E-R. Kami akan membahas istilah-istilah ini secara rinci di bagian selanjutnya (Komponen Diagram

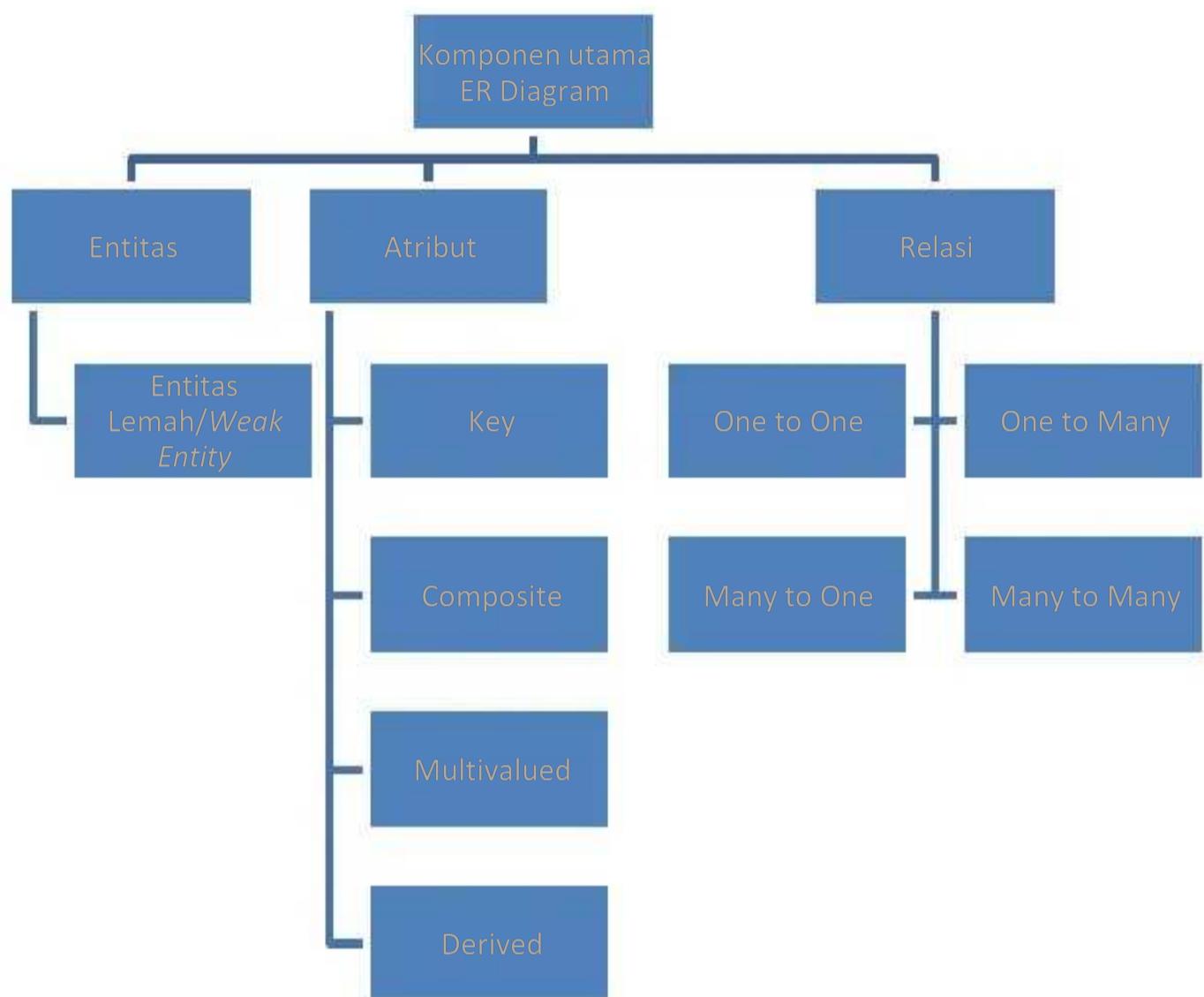
ER) dari panduan ini jadi jangan terlalu khawatir tentang istilah-istilah ini sekarang, cukup pelajari sekali saja.

Tabel 1. Simbol Komponen ER Diagram

No	Komponen	Simbol Geometris
1	Kumpulan Entitas	
2	Atribut	
3	Hubungan/Relasi	
4	Konektor Relasi, Entitas dan Atribut	
5	Atribut Multivilai/ <i>Multivalued</i>	
6	Atribut Turunan	
7	Entitas Lemah/Weak Entity	
8	Partisipasi total suatu entitas dalam himpunan hubungan	 

Komponen ER Diagram

Beberapa komponen ER Diagram yang direpresentasikan masing-masing simbol berbeda sehingga diagram tersebut mampu memberikan informasi jelas tentang skema perancangan sistem. Berikut gambar komponen utama ER Diagram:



Gambar 2. Komponen Utama ER Diagram

Seperti yang ditunjukkan pada gambar di atas, diagram ER memiliki tiga komponen utama:

1. Entitas

Entitas adalah objek atau komponen data. Entitas direpresentasikan sebagai persegi panjang dalam diagram ER. Sebagai contoh: Pada diagram ER berikut kita memiliki dua entitas Siswa dan Sekolah dan kedua entitas ini memiliki hubungan banyak ke satu karena banyak siswa yang belajar dalam satu sekolah. Kami akan membaca lebih lanjut tentang hubungan nanti, untuk saat ini fokus pada entitas.

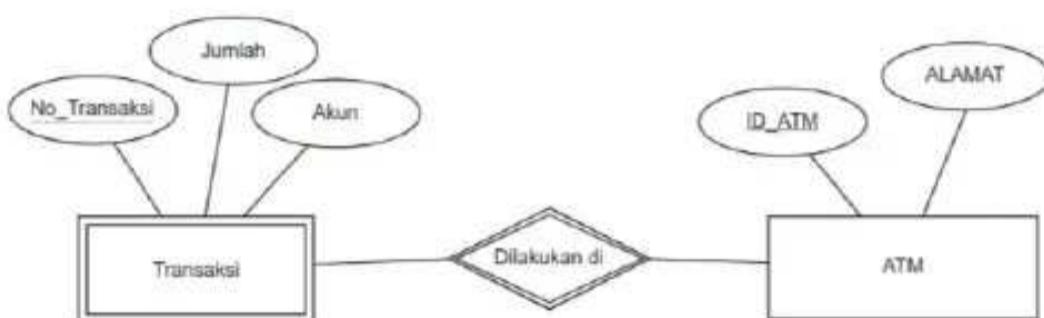


Sedangkan Entitas Lemah/*Weak Entity* merupakan Entitas yang tidak dapat diidentifikasi secara unik oleh atributnya sendiri dan bergantung pada hubungan dengan entitas lain disebut entitas lemah. Entitas yang lemah diwakili oleh persegi panjang ganda. Misalnya, akun bank tidak dapat diidentifikasi secara unik tanpa mengetahui bank yang memiliki akun tersebut, sehingga akun bank adalah entitas yang lemah.



Gambar 4. Contoh Entitas Lemah/*Weak Entity*

Entitas dapat juga lemah apabila entitas tidak memiliki atribut kuncinya. Itu dapat diidentifikasi secara unik dengan mempertimbangkan kunci utama dari entitas lain. Untuk itu, set entitas yang lemah perlu memiliki partisipasi.



Gambar 5. Contoh Entitas Lemah/*Weak Entity* yang mempunyai relasi/hubungan.

Pada contoh di atas, "No Transaksi" adalah pembeda dalam sekelompok transaksi di ATM. Mari kita pelajari lebih lanjut tentang entitas yang lemah dengan membandingkannya dengan Entitas yang kuat.

Tabel 2. Perbandingan Entitas Kuat Dan Lemah

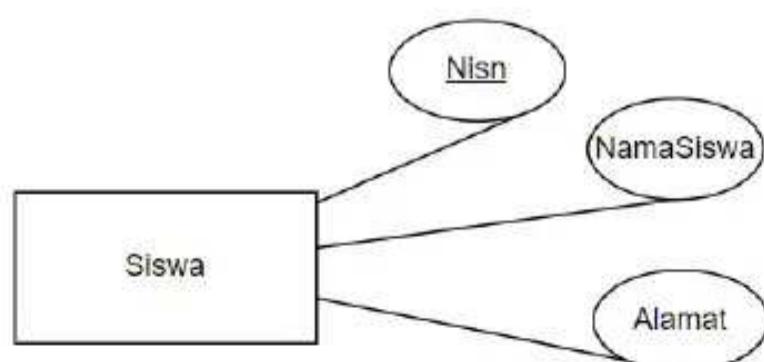
Entitas Kuat	Entitas Lemah
Kumpulan entitas yang kuat selalu memiliki kunci utama.	Itu tidak memiliki atribut yang cukup untuk membentuk kunci utama.
Ini diwakili oleh simbol persegi panjang.	Ini diwakili oleh simbol persegi panjang ganda.
Ini berisi kunci utama yang diwakili oleh simbol garis bawah.	Ini berisi Kunci Parsial yang diwakili oleh simbol garis bawah putus-putus.

2. Atribut

Atribut menggambarkan properti dari suatu entitas. Sebuah atribut direpresentasikan sebagai Oval dalam diagram ER. Ada empat jenis atribut:

a. Atribut kunci / *Key Attribute*

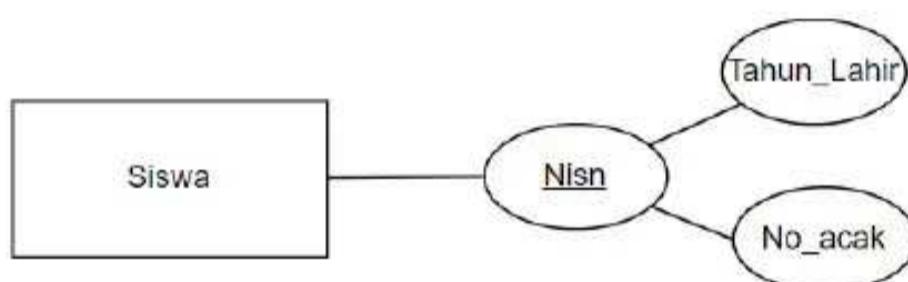
Atribut kunci dapat secara unik mengidentifikasi entitas dari kumpulan entitas. Misalnya, Nomor Induk Siswa Nasional (NISN) siswa dapat secara unik mengidentifikasi siswa dari sekumpulan siswa. Atribut kunci diwakili oleh oval sama seperti atribut lainnya namun teks atribut kunci digarisbawahi.



Gambar 6. Contoh Kunci Atribut.

b. Atribut komposit/*Composite Attribute*

Atribut yang merupakan kombinasi dari atribut lain dikenal sebagai atribut komposit. Misalnya, Dalam entitas siswa, NISN adalah atribut komposit karena terdiri dari atribut lain seperti tahun lahir dan no acak.



Gambar 7. Contoh Atribut Komposit.

c. Atribut multivilai/*Multivalued Attribute*

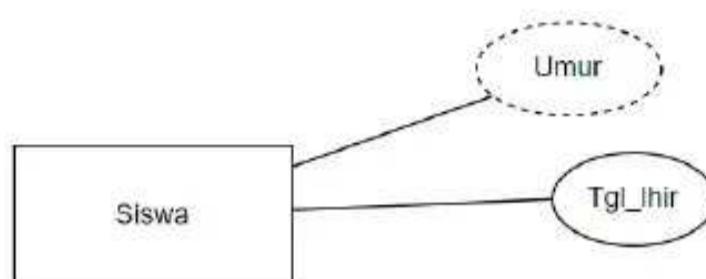
Atribut yang dapat menampung banyak nilai dikenal sebagai atribut multivilai. Hal ini diwakili dengan oval ganda dalam Diagram ER. Misalnya, Siswa dapat memiliki lebih dari satu alamat rumah sehingga atribut alamat multivilai.



Gambar 8. Contoh Atribut Multivilai/*Multivalued*.

d. Atribut turunan/*Derived Attribut*

Atribut turunan adalah atribut yang nilainya dinamis dan diturunkan dari atribut lain. Ini diwakili oleh oval putus-putus dalam Diagram ER. Misalnya – Usia orang adalah atribut turunan karena berubah seiring waktu dan dapat diturunkan dari atribut lain (Tanggal lahir).



Gambar 9. Contoh Atribut Turunan/ *Derived*.

3. Kardinalitas Relasi

Kardinalitas: Mendefinisikan atribut numerik dari hubungan antara dua entitas atau himpunan entitas.

Suatu hubungan diwakili oleh bentuk berlian dalam diagram ER, itu menunjukkan hubungan antar entitas. Ada empat jenis hubungan kardinal:

a. Satu Ke Satu (*One to One*)

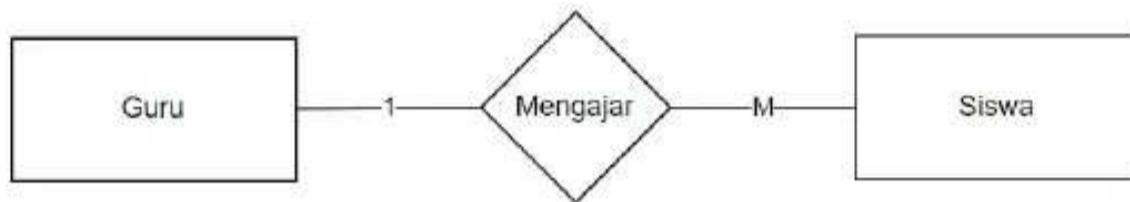
Ketika satu *record* dari suatu entitas dikaitkan dengan satu *record* dari entitas lain maka itu disebut hubungan satu ke satu. Misalnya, seorang Guru hanya boleh mengajar satu Mata pelajaran saja.



Gambar 11. Contoh Kardinalitas Relasi Satu ke Satu.

b. Satu ke Banyak (*One to Many*)

Ketika satu *record* dari suatu entitas dikaitkan dengan lebih dari satu *record* dari entitas lain maka itu disebut hubungan satu ke banyak. Misalnya, Guru dapat mengajar banyak siswa dalam satu waktu tetapi siswa tidak dapat diajar oleh banyak Guru dalam satu waktu.



Gambar 12. Contoh Kardinalitas Relasi Satu ke Banyak.

c. Banyak ke Satu (*Many to One*)

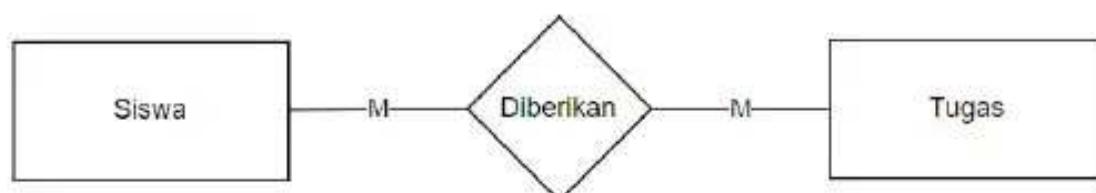
Ketika lebih dari satu *record* dari suatu entitas dikaitkan dengan satu *record* dari entitas lain maka itu disebut hubungan banyak ke satu. Misalnya, banyak siswa dapat belajar di satu Sekolah tetapi seorang siswa tidak dapat belajar di banyak Sekolah secara bersamaan.



Gambar 13. Contoh Kardinalitas Relasi Banyak ke Satu.

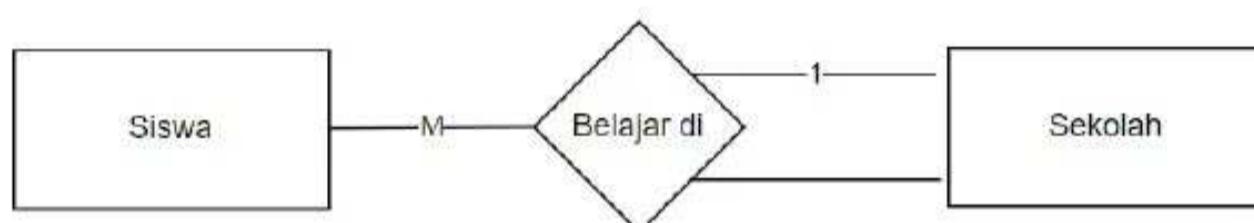
d. Banyak ke Banyak (*Many to Many*)

Ketika lebih dari satu *record* dari suatu entitas dikaitkan dengan lebih dari satu *record* dari entitas lain maka itu disebut hubungan banyak ke banyak. Misalnya, siswa dapat diberikan ke banyak tugas dan tugas dapat diberikan ke banyak siswa.



Gambar 14. Contoh Kardinalitas Relasi Banyak ke Banyak.

Dalam prakteknya ada beberapa kasus yang membutuhkan Partisipasi Total dari kumpulan Entitas. Partisipasi total dari himpunan entitas menyatakan bahwa setiap entitas dalam himpunan entitas harus memiliki setidaknya satu hubungan dalam himpunan hubungan. Sebagai contoh: Dalam diagram di bawah ini setiap Sekolah harus memiliki setidaknya satu Siswa terkait.



Gambar 15. Contoh Relasi Partisipasi Total Dari Kumpulan Entitas.

Langkah-Langkah Membuat ERD (E-R Diagram)

Berikut ini adalah langkah-langkah untuk membuat ERD.



Gambar 16. Langkah-langkah Membuat ERD.
Mari kita pelajari konsep diatas dengan sebuah contoh:

Di sekolah, seorang Siswa terdaftar di Mata Pelajaran. Seorang siswa harus ditugaskan ke setidaknya satu atau lebih Mata Pelajaran. Setiap Mata Pelajaran diajarkan oleh seorang Guru tunggal. Untuk menjaga kualitas pengajaran, seorang Guru hanya dapat menyampaikan satu Mata Pelajaran.

1. Langkah pertama: Identifikasi Entitas

Kami memiliki tiga entitas: Siswa, Mata Pelajaran dan Guru



Gambar 17. Identifikasi Entitas.

2. Langkah kedua: Identifikasi Hubungan/Relasi

Kami memiliki dua hubungan berikut: Siswa diajarkan Mata Pelajaran, Guru memberikan Mata Pelajaran.

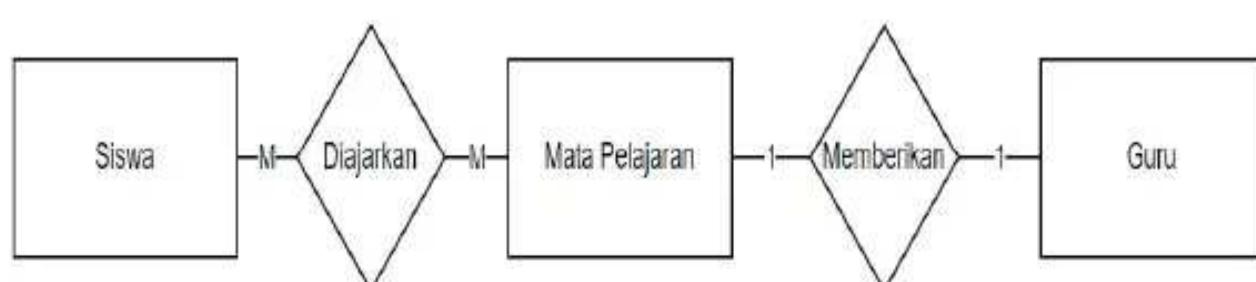


Gambar 18. Identifikasi Hubungan/Relasi.

3. Langkah ketiga: Identifikasi Kardinalitas

Dari pernyataan kasus yang kita bahas, kita tahu bahwa:

- Seorang Siswa dapat diberikan beberapa Mata Pelajaran dan juga Mata Pelajaran bisa diberikan ke banyak siswa
- Seorang Guru hanya dapat menyampaikan satu Mata Pelajaran dan sebaliknya



Gambar 19. Identifikasi Kardinalitas.

4. Langkah keempat: Identifikasi Atribut

Anda perlu mempelajari file, formulir, laporan, data yang saat ini dikelola oleh organisasi untuk

KONSEP DASAR MODEL ER

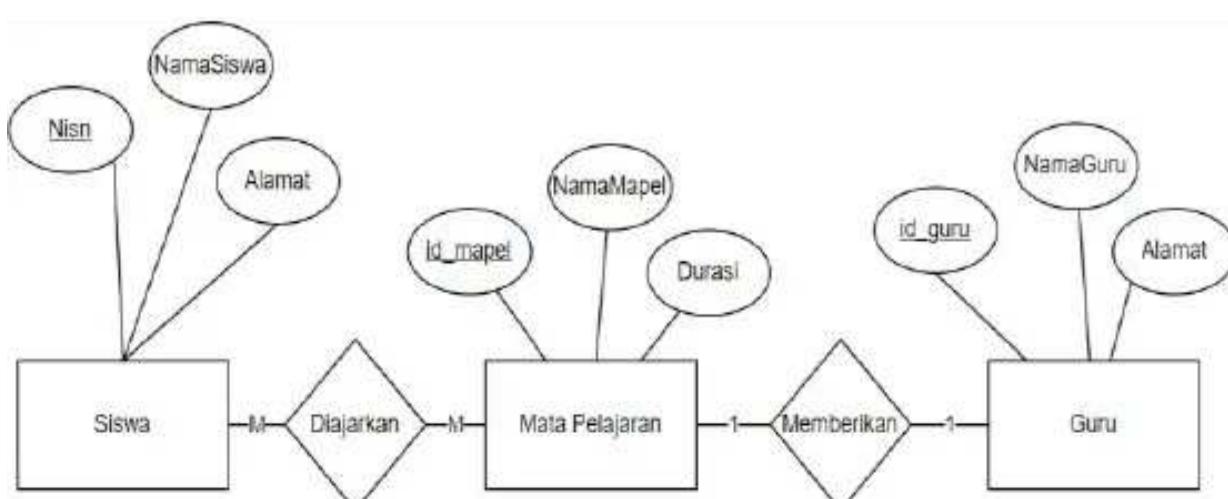
wawancara dengan berbagai pemangku kepentingan untuk mengidentifikasi entitas. Awalnya, penting untuk mengidentifikasi atribut tanpa memetakannya ke entitas tertentu.

Setelah Anda memiliki daftar Atribut, Anda perlu memetakannya ke entitas yang teridentifikasi. Pastikan sebuah atribut dipasangkan dengan tepat satu entitas. Jika menurut Anda atribut harus dimiliki oleh lebih dari satu entitas, gunakan pengubah untuk membuatnya unik. Contoh tabel pemetaan atribut masing-masing entitas sebagai berikut:

Tabel 3. Contoh pemetaan atribut masing-masing entitas.

No	Entitas	Kunci Atribut	Atribut
1	Siswa	Nisn	NamaSiswa, Alamat
2	Mata Pelajaran	Id_mapel	NamaMapel, Durasi
3	Guru	Id_guru	NamaGuru, Alamat

Setelah pemetaan selesai gambarkan kesebuah diagram ER secara detail sesuai dengan hasil pemetaan.

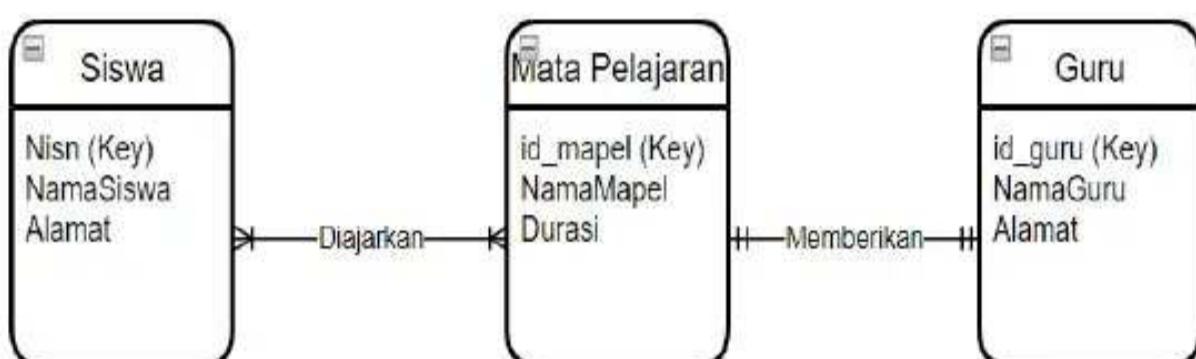


Gambar 20. Hasil ER Diagram.

Untuk Entitas Mata Pelajaran, atribut dapat berupa bahasan, Tugas, dll. Demi kemudahan, kami hanya mempertimbangkan beberapa atribut saja

5. Langkah kelima: Buat ER Diagram

Representasi yang lebih modern dari Diagram ERD sebagai berikut:



Gambar 21. ER Diagram Modern.

Praktik Terbaik untuk Mengembangkan Diagram ER yang Efektif

- a. Hilangkan entitas atau hubungan yang berlebihan
- b. Anda perlu memastikan bahwa semua entitas dan hubungan Anda diberi label dengan benar
- c. Mungkin ada berbagai pendekatan yang valid untuk diagram ER. Anda perlu memastikan bahwa diagram ER mendukung semua data yang perlu Anda simpan
- d. Anda harus memastikan bahwa setiap entitas hanya muncul satu kali dalam diagram ER
- e. Beri nama setiap relasi, entitas, dan atribut yang direpresentasikan pada diagram Anda
- f. Jangan pernah menghubungkan hubungan satu sama lain
- g. Anda harus menggunakan warna untuk menyorot bagian penting dari diagram ER

Dari pembahasan diatas bisa disimpulkan bahwa:

- a. Model ER adalah diagram model data tingkat tinggi
- b. Diagram ER adalah alat visual yang berguna untuk merepresentasikan model ER

- c. Diagram hubungan entitas menampilkan hubungan himpunan entitas yang disimpan dalam database
- d. Diagram ER membantu Anda mendefinisikan istilah yang terkait dengan pemodelan hubungan entitas
- e. Model ER didasarkan pada tiga konsep dasar: Entitas, Atribut & Hubungan/Relasi
- f. Entitas dapat berupa tempat, orang, objek, peristiwa atau konsep, yang menyimpan data dalam database
- g. Hubungan tidak lain adalah asosiasi antara dua atau lebih entitas
- h. Entitas yang lemah adalah tipe entitas yang tidak memiliki atribut kuncinya
- i. Properti bernilai tunggal baik tipe entitas atau tipe relasi
- j. Membantu Anda untuk mendefinisikan atribut numerik dari hubungan antara dua entitas atau himpunan entitas
- k. ER Diagram adalah representasi visual dari data yang menggambarkan bagaimana data terkait satu sama lain
- l. Saat Menggambar diagram ER, Anda perlu memastikan semua entitas dan relasi Anda diberi label dengan benar.

Daftar Pustaka

- Raza, N. H. (2015). Cost Estimation of Small Scale Web Based Applications using ER Model (Doctoral dissertation).
- Srivastava, S. (2019). ER Diagram.
- Javed, M., & Lin, Y. (2018, March). Iterative Process for Generating ER Diagram from Unrestricted Requirements. In ENASE (pp. 192-204).
- Anonim, (2022). What is an Entity Relationship Diagram (ERD)?
Online at
<https://www.lucidchart.com/pages/er-diagrams>,
accessed 20 Februari 2022

Profil Penulis



Mohammad Ridwan

Penulis mulai berminat terkait dengan teknologi komputer saat bekerja di reparasi komputer setelah lulus SMA pada tahun 2006 silam. Hal tersebut menjadi motivasi untuk penulis melanjutkan ke jenjang S1 Sistem Informasi Universitas Muria Kudus (UMK) Jawa Tengah. Penulis kemudian sambil bekerja di IT Garuda Indonesia Training Center (GITC) Cengkareng Jakarta Barat juga melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S2 di Magister Ilmu Komputer Universitas Budi Luhur (UBL) pada tahun 2016.

Penulis memiliki kepakaran dibidang Software Developer baik dari basis Web Application maupun Mobile Applicaton. Penulis juga menggeluti bidang keilmuan Data Mining serta keterkaitan dengan bidang Internet Of Thing Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI. Selain peneliti, penulis juga ingin aktif dalam menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini.

12

TRANSFORMASI ER KE MODEL DATA RELASIONAL

Muhammad Yani, S.Kom., M.T.I.

Universitas Mulia

Pendahuluan

Pada bab ini akan menjelaskan susunan logika data dan hubungan dari berbagai bagian di dalam perancangan basis data, proses transformasi model ER ke dalam bentuk model data relasional. Dalam transformasi ER dengan cara melakukan perubahan entitas menjadi nama tabel dan atribut menjadi kolom pada tabel. Pada proses transformasi, kardinalitas yang dimiliki setiap Model ER sangat menentukan untuk pembentukan tabel fisik pada sebuah basis data.

Di dalam bab ini akan dijelaskan bagaimana perancangan database menggunakan model data relasional. Berikut juga akan diberikan contoh proses transformasi model ER ke model data relasional.

Model Data Relasional

Model Data Relasional adalah model data ke dalam bentuk tabel relasi. Model ini diperkenalkan oleh E.F Codd pada tahun 1970. Pada model data relasional terdiri dari berbagai tabel dua dimensi yang memiliki relasi antar tabel lainnya di dalam basis data.

Keuntungan Model Data Relasional

Model data relasional sampai saat ini masih banyak digunakan dalam penerapan ke dalam basis data. Beberapa keuntungan model data relasional antara lain:

1. Model data relasional memiliki kemudahan-kemudahan dalam penerapan berbagai kebutuhan pengelahan basis data
2. Tabel merupakan yang digunakan bentuk fakta yang sama digunakan dalam penerapan basis data
3. Mudah diterapkan ke dalam perintah bahasa pemrograman basis data

Istilah-istilah dalam Model Data Relasional

Beberapa istilah yang penting dalam model data relasional yang akan sering ditemui yaitu:

1. Relasi

Relasi adalah tabel yang terdiri dari baris dan kolom. Di dalam penerapan basis data, relasi akan membentuk hubungan antar tabel di dalam basis data. Gambaran relasi data dilihat pada gambar 12.1.

2. Atribut

Atribut adalah suatu nama kolom dari tabel. Pada contoh gambar 12.1, kolom tabel nilai terdiri dari: kode_nilai, nim, kode_matakuliah dan nilai.

3. Tupel

Tupel adalah sebuah baris dari tabel. Pada contoh gambar 12.1, tupel terdapat 4 baris dari hasil relasi tabel nilai.

4. Domain

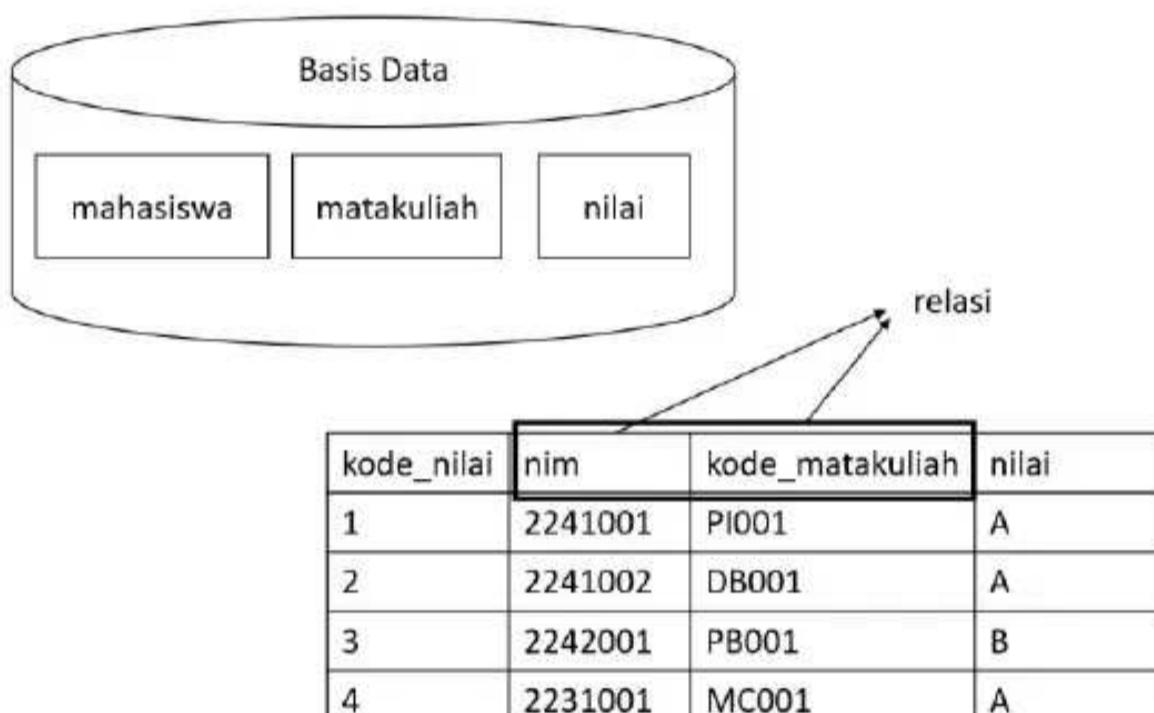
Domain adalah himpunan nilai yang berlaku pada satu atribut. Sebagai contoh, pada atribut nilai dengan domain berupa {A, A, B, A}.

5. Derajat

Derajat adalah jumlah atribut yang terdapat pada sebuah relasi. Pada gambar 12.1 relasi pada tabel nilai berjumlah 4 derajat.

6. Kardinalitas

Kardinalitas adalah jumlah tupel yang ada pada tabel. Sebagai contoh kardinalitas pada tabel nilai berjumlah 4 baris.



Gambar 12.1 Basis Data, Relasi

Aturan Umum dalam Transformasi ER Menjadi Model Data Relasional

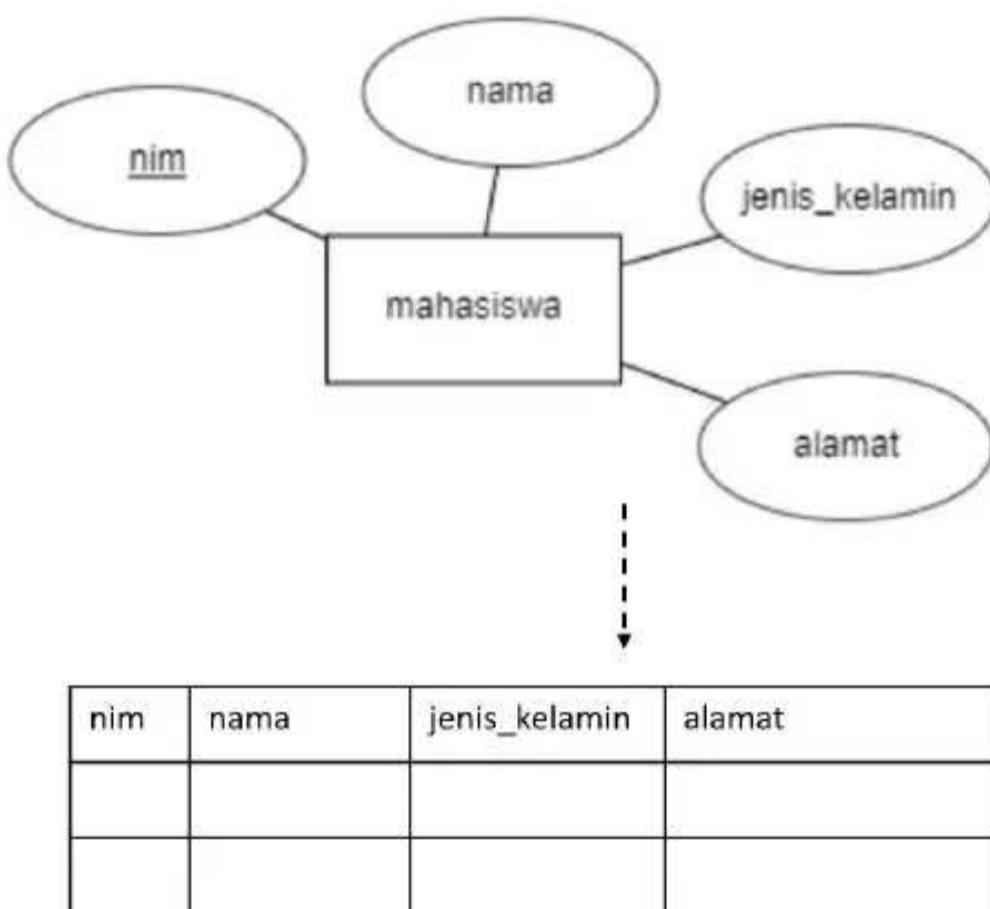
Data Relasional

Hasil pemodelan diagram ER perlu dilakukan transformasi ke bentuk relasi hingga sampai dengan bentuk fisik basis data. Beberapa tabel fisik yang merupakan hasil dari transformasi ER ke model data relasional.

Aturan umum yang sering digunakan dalam transformasi ER ke dalam model data relasional adalah mengubah entitas menjadi tabel, atribut menjadi kolom. Dalam pemberian nama atribut sebaiknya menggunakan huruf kecil dan tidak menggunakan spasi. Hal ini disebabkan nantinya pada proses penggunaan atribut pada relasi

TRANSFORMASI ER KE MODEL DATA RELASIONAL

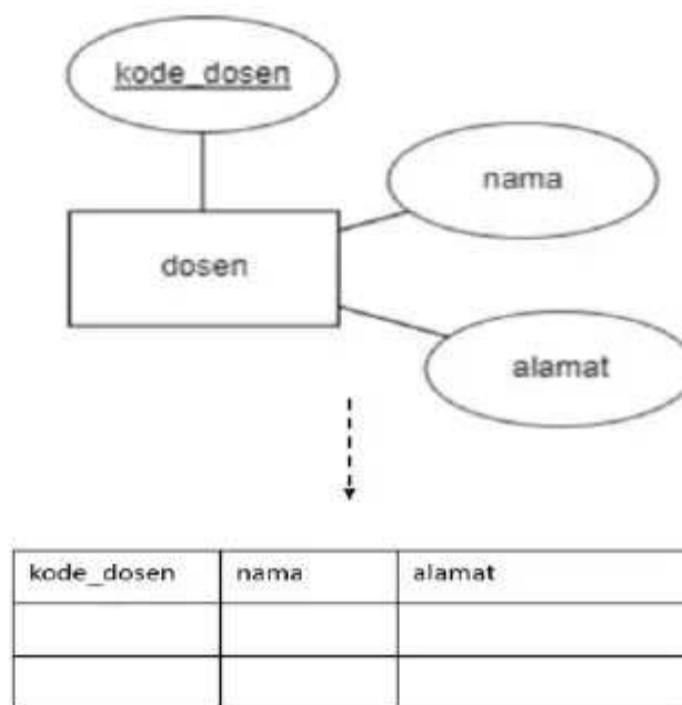
yang memiliki relasi. Berikut 3 ilustrasi sederhana transformasi ER ke dalam bentuk tabel fisik.



Gambar 12.2 Ilustrasi transformasi entitas mahasiswa

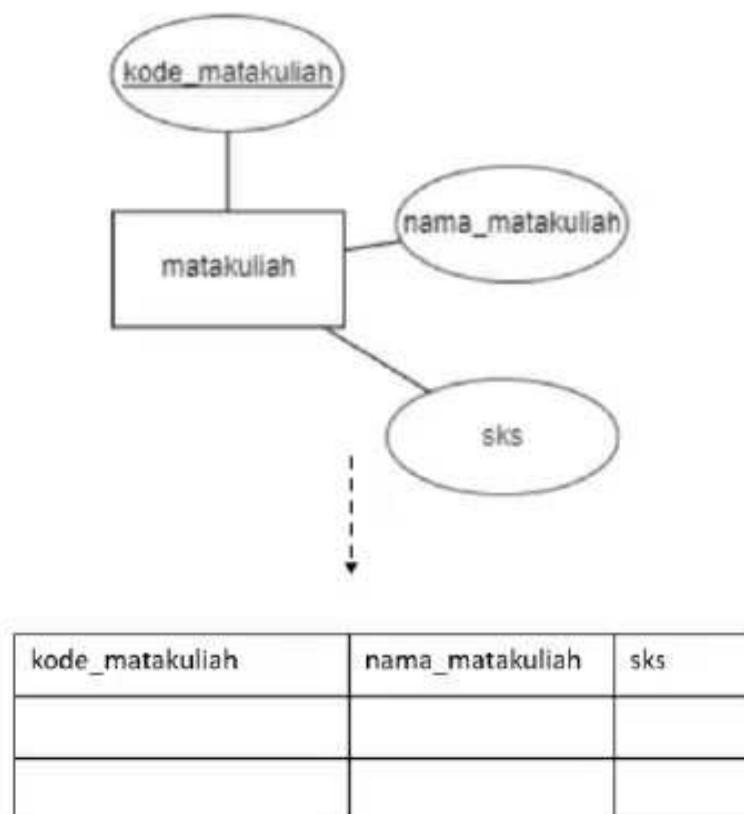
Berdasarkan pada gambar 12.2 entitas yang bernama mahasiswa pada Model ER berubah menjadi tabel mahasiswa, sedangkan atribut nim, nama, jenis_kelamin dan alamat berubah menjadi *field* dengan nim, nama, jenis_kelamin dan alamat. Pada umumnya dalam setiap atribut *key* pada entitas diletakkan pada kolom pertama pada tabel yang disebut sebagai kunci tabel (*primary key*), sehingga atribut *key* diletakkan di kolom pertama setiap tabel mahasiswa.

TRANSFORMASI ER KE MODEL DATA RELASIONAL



Gambar 12.3 Ilustrasi transformasi entitas dosen

Berdasarkan pada gambar 12.3 entitas yang bernama dosen pada Model ER berubah menjadi tabel dosen, sedangkan atribut kode_dosen, nama dan alamat berubah menjadi *field* dengan kode_dosen, nama dan alamat. Pada umumnya dalam setiap atribut *key* pada entitas diletakkan pada kolom pertama pada tabel yang disebut sebagai kunci tabel (*primary key*), sehingga atribut *key* diletakkan di kolom pertama setiap tabel dosen.



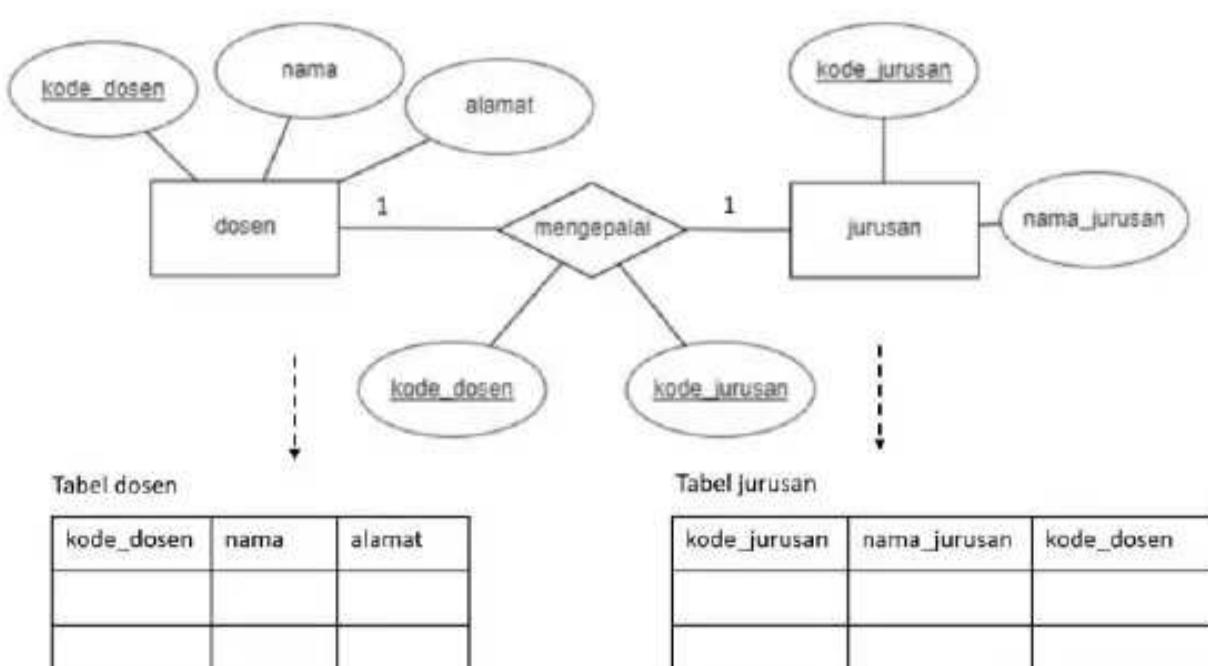
TRANSFORMASI ER KE MODEL DATA RELASIONAL

Berdasarkan pada gambar 12.4 entitas yang bernama matakuliah pada Model ER berubah menjadi tabel matakuliah, sedangkan atribut kode_matakuliah, nama_matakuliah dan sks berubah menjadi *field* dengan kode_matakuliah, nama_matakuliah dan sks. Pada umumnya dalam setiap atribut *key* pada entitas diletakkan pada kolom pertama pada tabel yang disebut sebagai kunci tabel (*primary key*), sehingga atribut *key* diletakkan di kolom pertama setiap tabel matakuliah.

Berdasarkan ketiga ilustrasi pada gambar 12.2, 12.3 dan 12.4 dapat disimpulkan bahwa transformasi model ER memiliki aturan transformasi. Suatu entitas ditransformasikan menjadi tabel, setiap tabel memiliki *field* atau kolom yang berasal dari atribut pada entitas. Jika dicermati, setiap tabel memiliki kolom kunci (*primary key*) yang berasal dari atribut *key*.

Transformasi Model ER untuk Kardinalitas Satu ke Satu

Pada bagian ini, disajikan ilustrasi transformasi model ER yang memiliki kardinalitas satu ke satu (1:1). Penting untuk diingat, bahwa derajat kardinalitas menentukan variasi transformasi ke tabelnya. Sehingga setiap derajat kardinalitas memiliki variasi transformasi tabel yang berbeda-beda.



Gambar 12.5

TRANSFORMASI ER KE MODEL DATA RELASIONAL

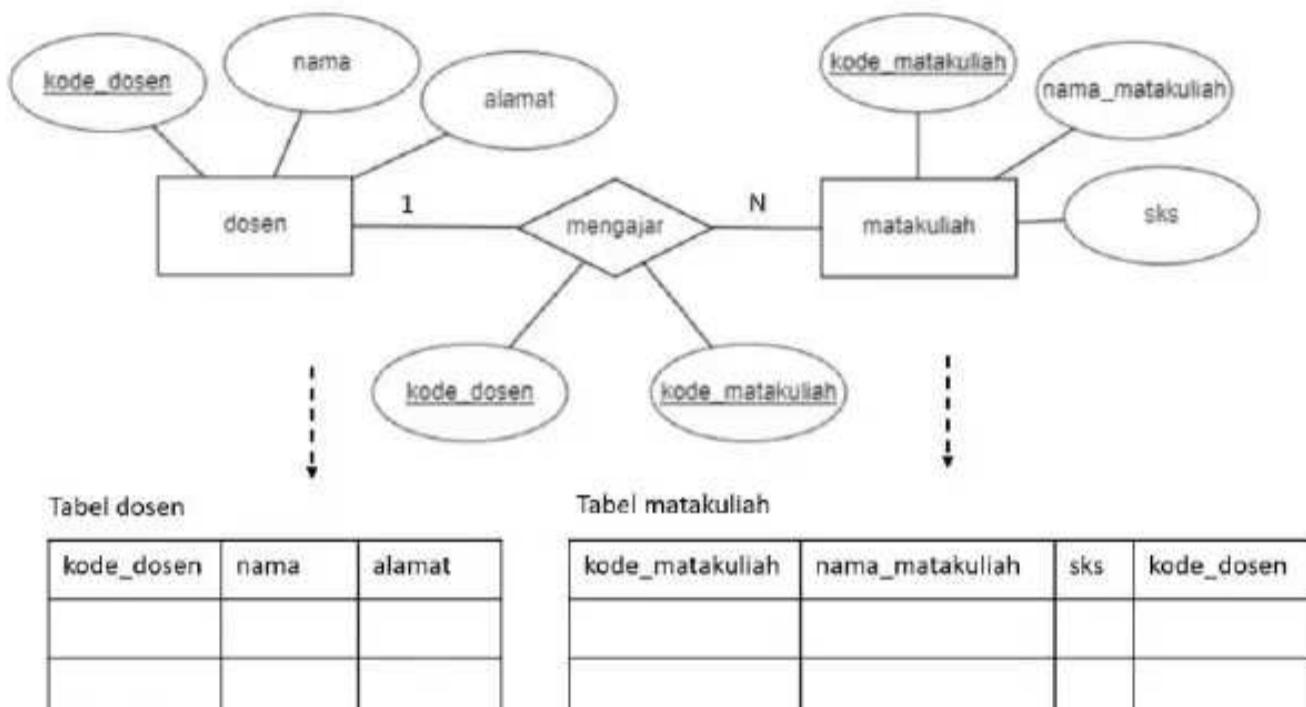
Berdasarkan ilustrasi pada gambar 12.5 menunjukkan bahwa entitas dosen ditransformasi menjadi tabel dosen, atribut entitas dosen menjadi kolom kode_dosen, nama dan alamat pada tabel dosen. Sedangkan pada entitas jurusan ditransformasi menjadi tabel jurusan, atribut entitas jurusan menjadi kolom kode_jurusan, nama_jurusan dan kode_dosen.

Pada kedua entitas tersebut memiliki relasi mengepalai dengan memiliki atribut kode_dosen dan kode_jurusan. Pada tabel jurusan bertambah 1 kolom baru yaitu kode_dosen yang merupakan hasil dari relasi antar dosen dan jurusan. Pada tabel jurusan, kode_jurusan sebagai *primary key* sedangkan kode_dosen sebagai *foreign key*. Jika dicermati, pada tabel jurusan terdapat *field* atau nama kolom yang sama pada tabel dosen yaitu *field* kode_dosen. Hal ini disebabkan dari hasil relasi yang memiliki derajat kardinalitas satu ke satu (1:1) antara tabel dosen dan jurusan, yang artinya setiap satu kode_dosen hanya boleh memiliki satu hubungan dengan tabel jurusan.

Transformasi Model ER untuk Kardinalitas Satu ke Banyak

Berikutnya disajikan ilustrasi transformasi model ER yang memiliki kardinalitas satu ke banyak (1:N). Perlu diingatkan kembali, bahwa derajat kardinalitas menentukan variasi transformasi ke tabelnya. Sehingga setiap derajat kardinalitas memiliki variasi transformasi tabel yang berbeda-beda.

TRANSFORMASI ER KE MODEL DATA RELASIONAL



Gambar 12.6

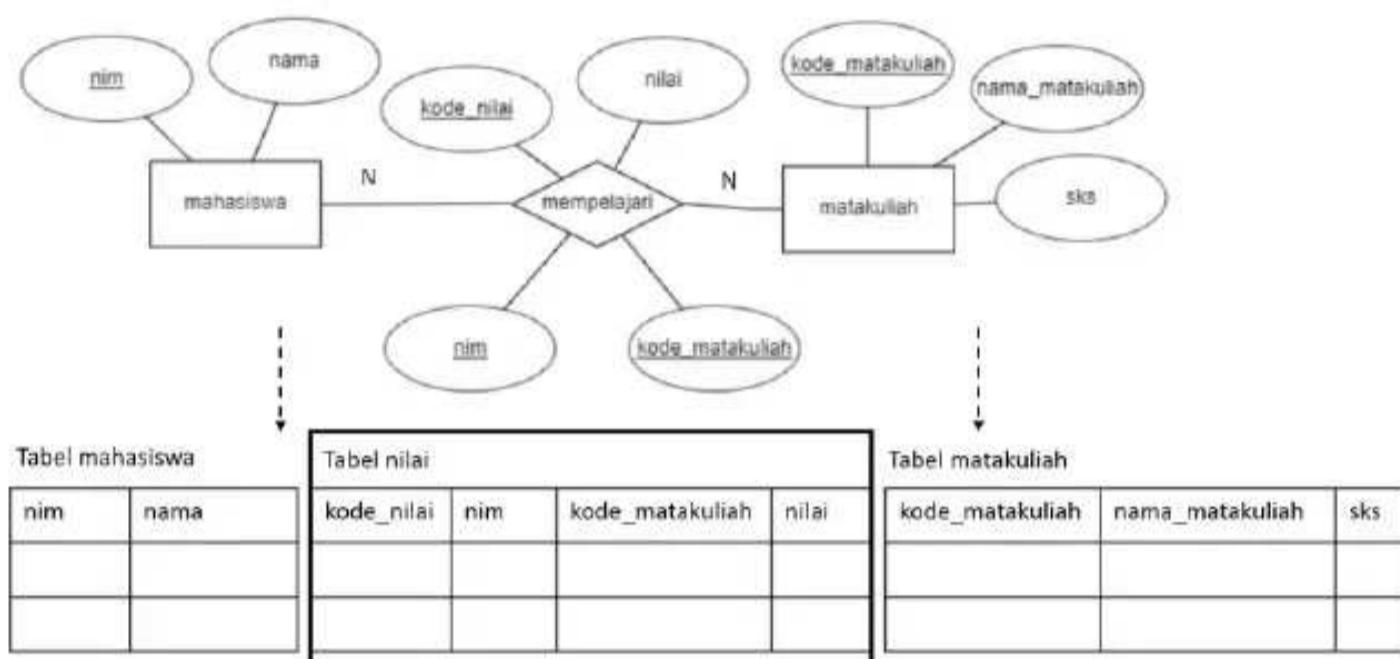
Transformasi Model ER dengan kardinalitas satu ke banyak

Berdasarkan ilustrasi pada gambar 12.6 menunjukkan bahwa entitas dosen ditransformasi menjadi tabel dosen, atribut entitas dosen menjadi kolom kode_dosen, nama dan alamat pada tabel dosen. Sedangkan pada entitas matakuliah ditransformasi menjadi tabel matakuliah, atribut entitas matakuliah menjadi kolom kode_matakuliah, nama_matakuliah, sks dan kode_dosen.

Pada gambar 12.6 menunjukkan relasi dengan derajat kardinalitas satu ke banyak (1:N). Relasi dengan keterangan mengajar dengan memiliki atribut kode_dosen dan kode_matakuliah. Pada tabel matakuliah bertambah 1 kolom baru yaitu kode_dosen yang merupakan hasil dari relasi antar dosen dan matakuliah. Sehingga pada tabel matakuliah kode_matakuliah sebagai *primary key* sedangkan kode_dosen sebagai *foreign key*. Jika dicermati, pada tabel matakuliah terdapat *field* atau nama kolom yang sama pada tabel dosen yaitu *field* kode_dosen. Hal ini disebabkan dari hasil relasi yang memiliki derajat kardinalitas satu ke satu (1:N) antara tabel dosen dan jurusan, yang artinya setiap satu kode_dosen boleh mengajar lebih dari satu matakuliah.

Transformasi Model ER untuk Kardinalitas Banyak ke Banyak

Berikutnya disajikan ilustrasi transformasi model ER yang memiliki kardinalitas banyak ke banyak (N:N). transformasi ER ke tabel pada membentuk tabel baru dari hasil proses relasi.



Gambar 12.7

Transformasi Model ER dengan kardinalitas banyak ke banyak

Seperti yang ada pada gambar 12.7 menunjukkan transformasi entitas mahasiswa dan matakuliah membentuk atribut baru. Dari hasil transformasi ini, atribut baru didefinisikan sebagai tabel fisik baru dengan nama tabel nilai sesuai dengan keterangan relasinya. Pada tabel nilai terdiri dari kolom yaitu kode_nilai, nim, kode_matakuliah dan nilai.

Pada kardinalitas banyak ke banyak (N:N) untuk desain pemodelan ER, simbol relasinya menggunakan atribut *key* yang berasal dari entitas yang dihubungkannya. Selain itu pada transformasi ER ke relasi ditambahkan atribut deskriptif, sehingga hasil konversi pemodelan tersebut terlihat jelas pada tabelnya.

Hasil transformasi ER pada kardinalitas banyak ke banyak terlihat membentuk tabel fisik, oleh sebab itu pada ilustrasi yang ditunjukkan pada gambar 12.7 entitas

tabel fisik dengan nama menjadi tabel nilai. Jika diartikan hasil relasi di atas adalah banyak mahasiswa dapat banyak mempelajari mata kuliah. Berdasarkan hasil ilustrasi transformasi di atas, dapat dipastikan bahwa peran kardinalitas dapat mempengaruhi bentuk fisik tabel dalam basis data. Jadi, saat melakukan transformasi Model ER ke bentuk model data relasional perlu menganalisis proses bisnis sistemnya, agar dapat menggambarkan model ER.

Implementasi Transformasi ER dengan Entitas Lemah (Weak Entity)

Pada prinsipnya, entitas lemah terbentuk dari hasil dekomposisi atribut. Beberapa atribut yang dimiliki entitas induk akan disesuaikan atau didistribusikan pada entitas yang diciptakan dari keberadaan atribut tersebut.

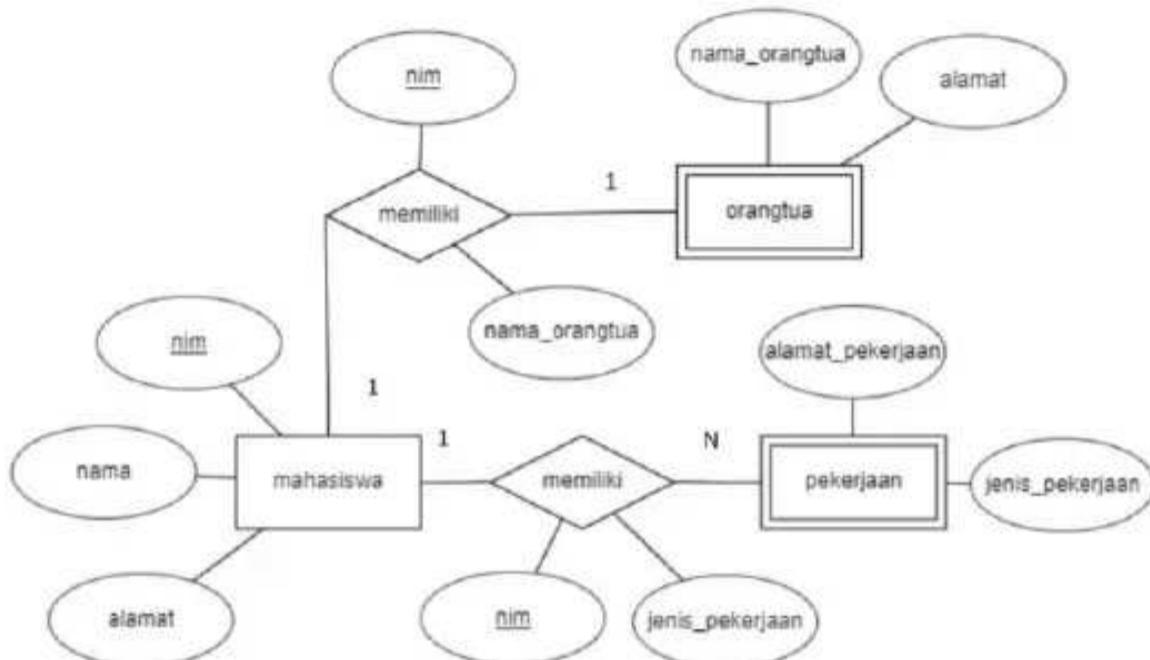
Himpunan entitas lemah tidak dapat berdiri sendiri tanpa entitas kuat.

Dalam mentransformasikan dari ER ke dalam bentuk basis data fisik keduanya sama akan terbentuk sebuah tabel. Namun yang membedakan adalah jika entitas kuat jika ditransformasikan ke dalam basis data fisik langsung menjadi tabel memiliki *atribut key* yang bertindak sebagai *primary key* dan atribut yang lain secara utuh, maka entitas lemah hanya dapat di transformasikan dalam bentuk tabel dengan menyertakan *primary key* dari entitas kuat dan atribut yang dimiliki oleh entitas lemah itu sendiri.

Pada gambar 12.8 diilustrasikan transformasi ER ke entitas lemah, setelah di analisis entitas mahasiswa memiliki atribut yang dapat dipisahkan dari entitas kuat, salah satu cara ini digunakan untuk membagi atribut dalam satu entitas. Jadi, terdapat entitas lemah dari entitas mahasiswa yang bernama sebagai berikut:

1. *Weak Entity* orangtua dengan atribut *nama_orangtua* dan alamat.
2. *Weak Entity* pekerjaan dengan atribut *jenis_pekerjaan* dan alamat pekerjaan.

TRANSFORMASI ER KE MODEL DATA RELASIONAL



Gambar 12.8 Ilustrasi entitas lemah dan relasinya

Berikut hasil dari transformasi ER entitas lemah dan relasinya menjadi tabel fisik, sehingga tabel dan relasinya seperti pada gambar 12.9.

Tabel mahasiswa

nim	nama	alamat

Tabel orangtua

nim	nama_orangtua

Key diambil dari himpunan entitas kuat

Tabel pekerjaan

nim	jenis_pekerjaan

Gambar 12.9

himpunan tabel dari hasil transformasi entitas lemah

Pada gambar di atas jelas terlihat bahwa entitas kuat dapat langsung ditransformasikan, membentuk tabel

dengan atribut yang dimiliki oleh dirinya sendiri. Sedangkan pada entitas orangtua yang merupakan entitas

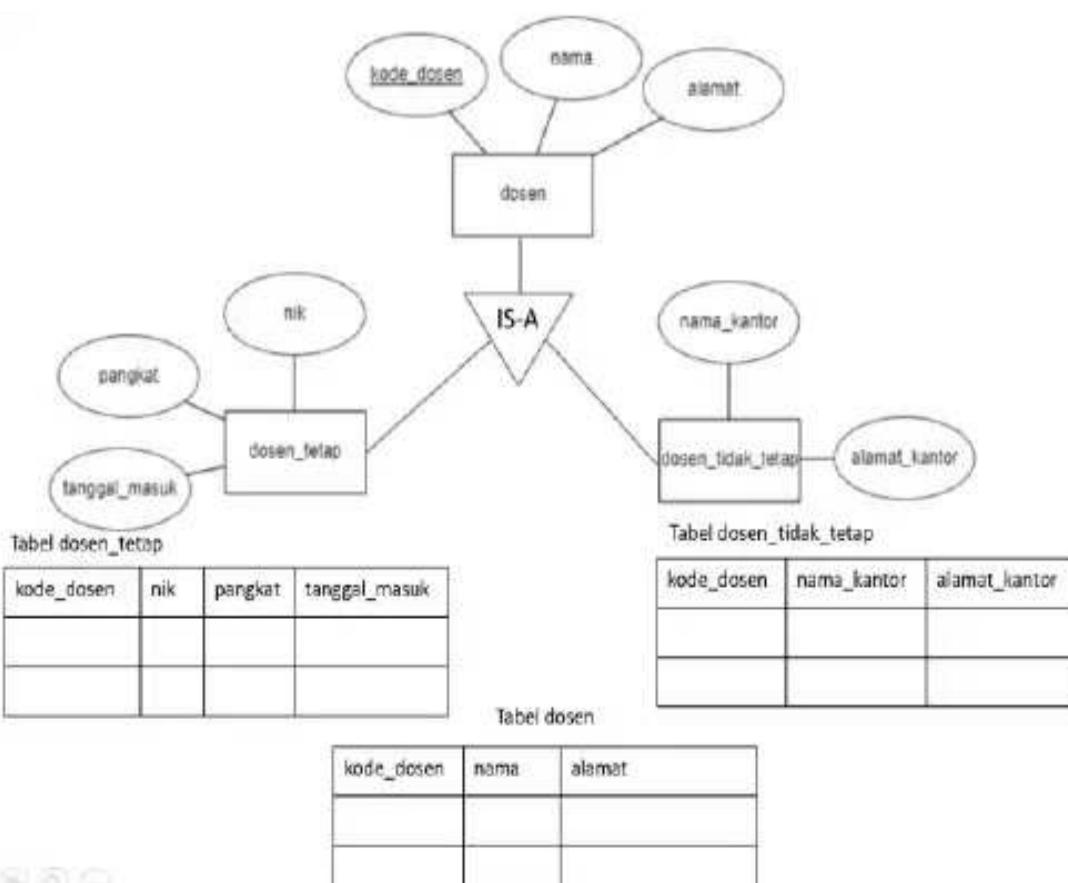
lemah setelah ditransformasikan ke bentuk tabel basis data atributnya harus mengambil *primary key* dari entitas mahasiswa yang merupakan entitas kuat karena memang entitas lemah tidak dapat berdiri sendiri. Begitu juga dengan entitas pekerjaan.

Transformasi Model ER untuk Relasi Khusus IS-A

Suatu entitas mengalami proses spesialisasi IS-A, entitas turunannya tidak memiliki atribut *key*. Entitas turunan menggunakan atribut *key* yang berasal dari entitas induknya. Dapat dilakukan *tracing entity* keterkaitan antara entitas induk dengan sub-sub entitasnya.

Dalam pemberian nama sub-sub entitas sangat disarankan menggunakan nama depan entitas induknya, hal ini dapat memudahkan bagi seorang pengembang aplikasi basis data. Pada gambar 12.10, sebagai contoh entitas dosen yang memiliki sub-sub entitas, yaitu:

1. Dosen tetap yang memiliki nik, pangkat, tanggal_masuk.
2. Dosen tidak tetap yang memiliki nama_kantor, alamat_kantor.



Gambar 12.10 Ilustrasi Transformasi pemodelan IS-A

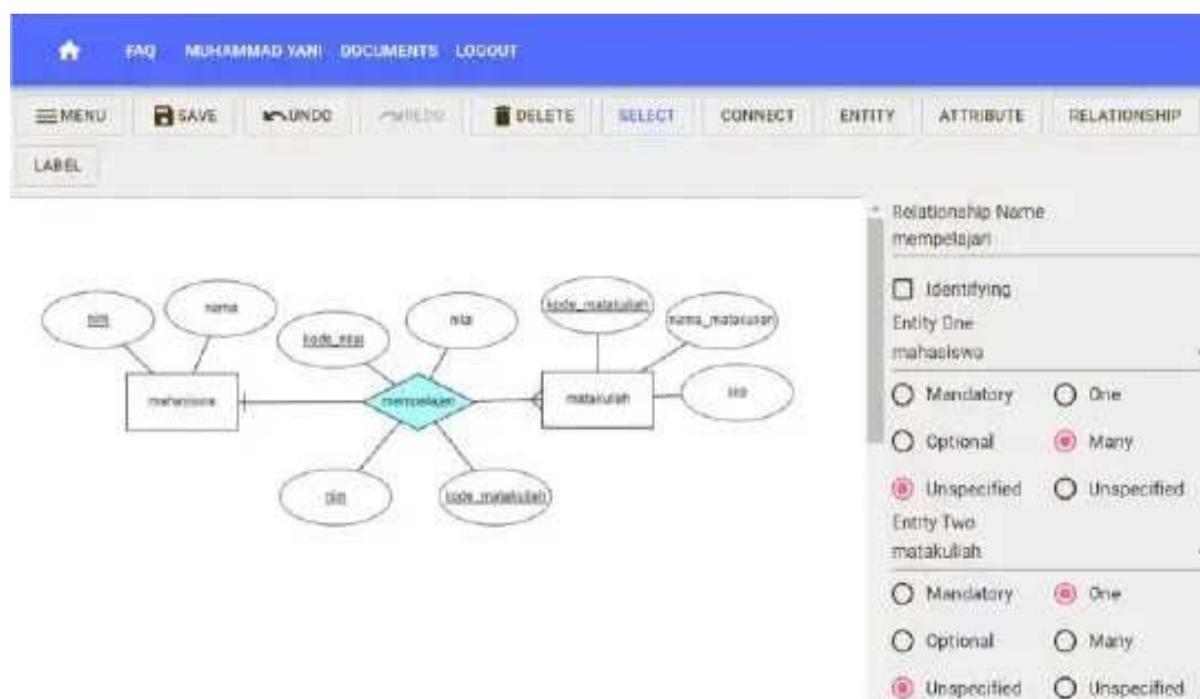
TRANSFORMASI ER KE MODEL DATA RELASIONAL

Setiap entitas, yaitu entitas dosen tetap dan dosen tidak tetap ditransformasikan menjadi tabel fisik. Atribut *key* diberi nama *kode_dosen* dari entitas dosen yang harus dipakai pada entitas turunannya, yaitu *dosen_tetap* dan *dosen_tidak_tetap*. Hal ini dilakukan karena melalui atribut *key* *kode_dosen* akan digunakan pada setiap tabel yang berelasi.

Penggunaan model IS-A tidak menggunakan derajat kardinalitas. Meskipun demikian, setiap entitas akan menjadi suatu tabel fisik. Seperti pada derajat kardinalitas banyak ke banyak (N:N).

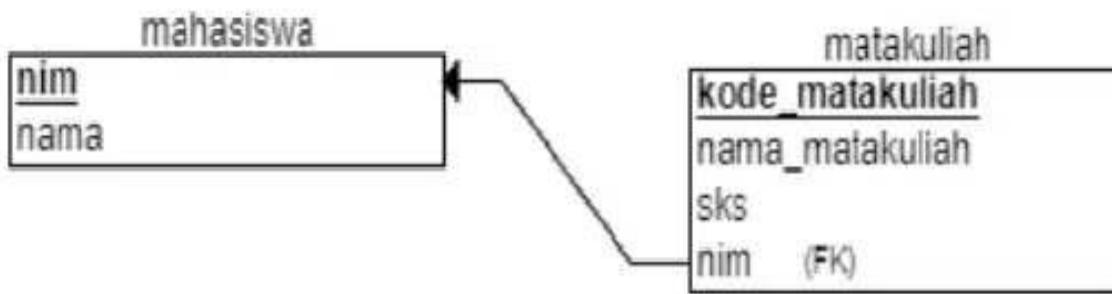
Studi Kasus Transformasi ER ke Tabel Fisik Menggunakan ERDPLUS

Pada bagian ini, memberikan studi kasus transformasi ER menjadi tabel fisik menggunakan tools online yaitu ERDPLUS dengan alamat erdplus.com. Entitas yang digunakan pada pembahasan sebelumnya yaitu entitas mahasiswa dan mata kuliah. Prosesnya entitas dibuat dengan menggunakan ERDPLUS selanjutnya dikonversi menjadi model data relasional atau tabel fisik. Berikut proses transformasi ER ke model data relasional ditunjukan pada gambar 12.11.



Gambar 12.11 proses pembuatan ER menggunakan ERDPLUS

Selanjutnya, diagram ER dikonversi menjadi model data



Gambar 12.12
hasil konversi diagram ER ke Model Data Relasional
(tabel fisik)

Pada gambar di atas, *Primary key* pada tabel di atas yaitu nim dan kode_matakuliah. Selanjutnya tabel mahasiswa dengan *field* (kolom) nim dan nama, sedangkan tabel matakuliah dengan kolom kode_matakuliah, nama_matakuliah, sks dan nim sebagai *foreign key*.

Dari studi kasus di atas, bahwa dalam melakukan transformasi ER perlu analisis dalam menentukan hubungan relasi setiap tabel, dengan menentukan derajat relasi setiap entitas akan mempermudah proses transformasi ER ke dalam bentuk model data relasional atau tabel fisik.

Daftar Pustaka

- Kadir, A. (2009). Dasar Perancangan dan Implementasi Database Relasional. Yogyakarta: Penerbit Andi.
- Priyadi, Y. (2014). Kolaborasi SQL dan ERD dalam implementasi Database. Bandung: Penerbit Andi.
- Umar, R., Hadi, A., Widiandana, P., Anwar, F., Jundullah, M., & Ikrom, A. (2019). Perancangan Database Point of Sales Apotek Dengan Menerapkan Model Data Relasional. QUERY: Jurnal Sistem Informasi, 3, 33-41.

Profil Penulis



Muhammad Yani

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2008 silam. Hal tersebut membuat penulis memilih untuk masuk ke Sekolah Tinggi Manajemen Informatika dan Komputer Widya Cipta Dharma Samarinda dengan memilih Jurusan Sistem Informasi dan berhasil lulus pada tahun 2012. Penulis bekerja sebagai Pendidik di SMK TI Airlangga sejak 2011 sampai dengan sekarang. Penulis kemudian melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S2 di prodi TEKNIK INFORMATIKA UNIVERSITAS BINA NUSANTARA di Jakarta pada tahun 2019. Selanjutnya 2019 akhir, Penulis bergabung sebagai Dosen Tetap di Universitas Mulia PSDKU Kota Samarinda sampai dengan sekarang. Saat ini, penulis sedang melanjutkan studi Ph.D di Universiti Malaysia Sarawak (UNIMAS).

Penulis memiliki kepakaran dibidang teknologi aplikasi desktop dan web content management system. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan hasil penelitiannya dilanjutkan dengan kegiatan pengabdian masyarakat penulis. Selain peneliti, penulis juga aktif sebagai instruktur pelatihan Teknologi Informasi bagi Tenaga Pendidik dan Tenaga Kependidikan di tingkat provinsi Kalimantan Timur. Penulis juga sebagai content creator di Youtube: https://www.youtube.com/c/MuhammadYani_id dan web

www.muhammadyani.com

Email Penulis: muhammadyani@universitasmulia.ac.id

13

NORMALISASI BASIS DATA

Herianto, S.Pd., MT.

Teknologi Informasi - Universitas Darma Persada

Pengertian dan Tujuan Normalisasi

Normalisasi adalah teknik merancang basis data (database) yang tujuannya untuk meminimalkan redundansi dan menghilangkan anomali. Redundansi sendiri merupakan data duplikasi atau pengulangan yang tidak boleh terjadi pada struktur database kecuali pada *foreign key* (kunci asing). Anomali adalah efek samping yang tidak diharapkan yang dapat menyebabkan ketidak konsistenan data atau membuat sebuah data menjadi hilang ketika data lain dihapus. Proses normalisasi pada setiap tahapnya melakukan dekomposisi (memecah) sebuah table yang besar menjadi beberapa table yang lebih kecil dan mengaitkan antar table tersebut dengan prinsip relasi (*relationship*). Hasil dari proses normalisasi ini menghasilkan struktur database yang secara logik disebut database yang baik yaitu database yang redundansinya minimal, terintegrasi dan siap digunakan oleh aplikasi melalui query atau SQL.

Tabel Universal

Proses normalisasi dimulai dari sebuah tabel universal, yaitu sebuah tabel yang mengandung semua item yang akan disimpan pada database. Tabel Universal ini umumnya bersifat tidak normal (*Unnormalize form*) yaitu tabel yang masih memiliki data duplikat (*redundancy*) dan data tidak konsisten (*inconsistency*). Karena tabel

data tidak konsisten (*inconsistency*). Kenapa tabel universal ini bersifat tidak normal? Karena tabel ini

diperoleh dari mengumpulkan semua item data yang sumbernya dari dokumen sistem yang akan dirancang misalnya dari laporan-laporan. Mengumpulkan semua item data dari sebuah sistem ke dalam sebuah tabel umumnya akan menghasilkan data duplikat (berulang) karena hubungan antara item data seringkali tidak bersifat satu ke satu, tetapi satu ke banyak bahkan banyak ke banyak yang jika dicatat dalam 1 tabel akan menyebabkan duplikasi.

Contoh dari dokumen berikut:

NIM	2020240063			
Nama Mahasiswa	RAHMA HANNI IZZATI			
SKS Lulus	64			
IPK SKS Lulus	3.84			
Periode	2021/2022 Gasal			
No	Kode	Nama Mata Kuliah	SKS	Nilai
1	00110010	PENDIDIKAN AGAMA ISLAM	2	A
2	20710010	ETIKA REKAYASA	2	A
3	24310021	ANALISA PROSES BISNIS	2	A-
4	24320150	PEMROGRAMAN WEB BERBASIS FRAMEWORK	2	A-
5	24320161	PRAK. PEMROGRAMAN WEB BERBASIS FRAMEWORK	1	A-
6	24320171	SISTEM OPERASI	3	A
7	24320181	ANALISIS & DESAIN SISTEM INFORMASI BERBASIS OBJEK	3	A
8	24320191	DASAR AKUNTANSI & APLIKASI	2	A
9	24320201	PRAK. APLIKASI AKUTANSI	1	A
10	24320211	REKAYASA PERANGKAT LUNAK	3	B
Total SKS			21	
Indeks Prestasi Semester			3.79	

Gambar 1. Dokumen Kartu Hasil Belajar (KHS) seorang mahasiswa pada sebuah Perguruan Tinggi

Jika disusun kembali maka item data dari dokumen di atas terdiri dari:

-
1. Periode (periode)
 2. NIM (nim)
-

3. Nama Mahasiswa (nm_mhs)
4. Kode Mata Kuliah (kd_mk)
5. Nama Mata Kuliah (nm_mk)
6. SKS (sks)
7. Nilai (nilai)
8. Total SKS Semester (sk_sem)
9. Indeks Prestasi Semester (ips)
10. SKS Lulus (t_sks)
11. IPK SKS Lulus (ipk)

Secara sederhana jika item/atribut di atas dinyatakan dalam bentuk relasi yang disebut relasi: universal, maka dapat juga dituliskan dengan:

Universal (periode, nim, nm_mhs, kd_mk, nm_mk, sks, nilai, sk_sem, ips, t_sks, ipk)

Jika dibuat dalam bentuk tabel yang disebut tabel universal. Hasilnya adalah:

Tabel 1. Tabel Universal / Unnormal:

periode	nim	nm_mhs	kd_mk	nm_mk	sks	nilai	ips	sk_sem	t_sks	ipk
2021/2022 Gasal	202024 0063	Rahma Hanni Izzati	00110010	PENDIDIKAN AGAMA	2	A	3,79	21	64	3,84
2021/2022 Gasal	202024 0063	Rahma Hanni Izzati	20710010	ETIKA REKAYASA	2	A	3,79	21	64	3,84
2021/2022 Gasal	202024 0063	Rahma Hanni Izzati	24310021	ANALISA PROSES BISNIS	2	A-	3,79	21	64	3,84
dstnya...

Jika diperhatikan tabel di atas memiliki beberapa kolom yang bersifat duplikat (yaitu kolom: periode, nim, nm_mhs, ips, sks_sem, t_sks, ipk). Hal ini akan mengakibatkan anomali jika diberi operasi: *insert*, *update* dan *delete*. Penjelasan tentang bagaimana terjadinya anomali dan contohnya pada tabel di atas adalah sebagai berikut:

1. Anomali Update:

Jika nama mata kuliah pada tabel di atas diubah (*update*) dari: Pendidikan Agama menjadi Pendidikan Agama Islam, maka yang berubah hanya pada dokumen ~~di atas~~ ~~saja~~ ~~atau~~ ~~yang~~ mengambil mata kuliah tersebut. Sementara pada data nilai mahasiswa lain yang mengambil mata kuliahnya belum tentu turut berubah. Ini yang disebut inkonsistensi pada data, yaitu data yang semestinya sama di tempat yang berbeda memiliki nilai yang berbeda.

2. Anomali Insert:

Misal pada tabel di atas ingin ditambahkan (*insert*) mata kuliah Sistem Basis Data yang belum diambil oleh mahasiswa pada periode tersebut, maka akibatnya data nim, nama_mhs dan periode harus diberi nilai kosong (null) pada tabel tersebut. Ini dapat mengakibatkan anomali karena beberapa aturan seringkali melarang memberikan nilai null ke item tertentu.

3. Anomali Delete:

Contoh pada tabel di atas terjadi kebutuhan menghapus (*delete*) mata kuliah Etika Rekayasa karena ada kesalahan input atau penyebab lain, maka akibatnya data tentang mata kuliah Etika Rekayasa baik item kode, nama dan besar sks dan sebagainya juga ikut hilang dari penyimpanan yang mungkin saja item-item tersebut dibutuhkan oleh laporan lain.

Proses Normalisasi

Seperti yang telah dijelaskan sebelumnya, tujuan dari normalisasi adalah untuk menghasilkan tabel yang normal yang terhindar dari duplikasi dan anomali. Proses normalisasi ini dapat dilakukan beberapa tahap:

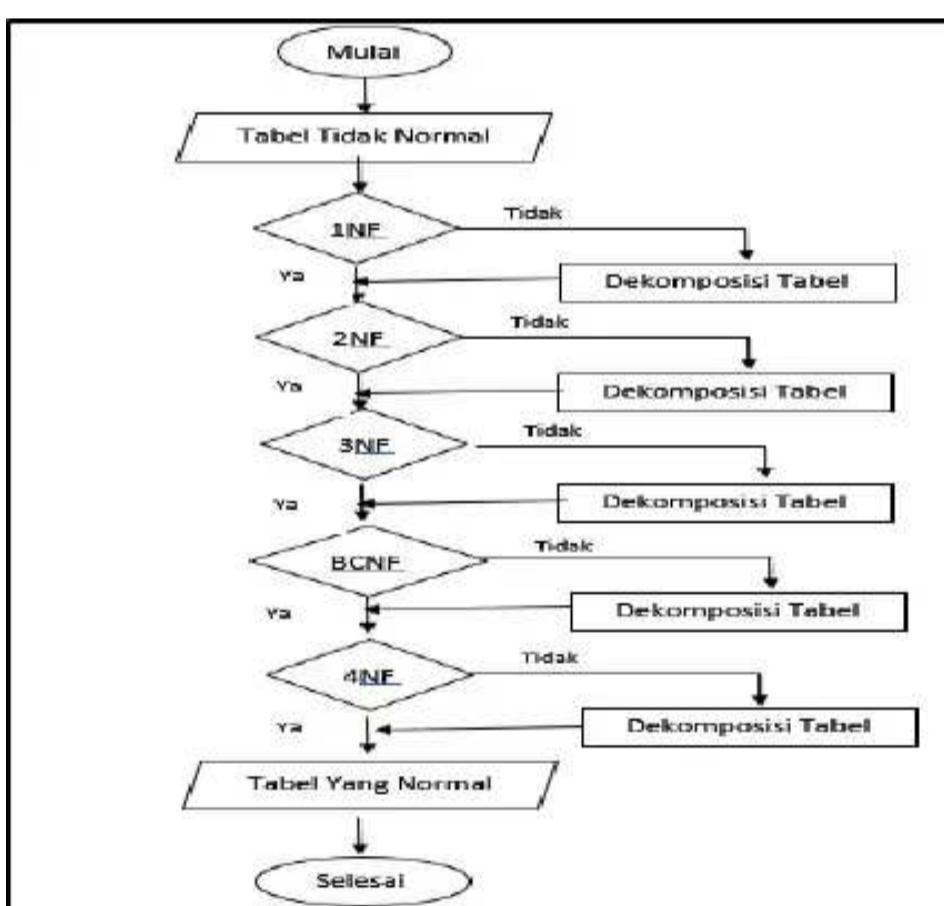
1. 1stNF (First Normal Form),
2. 2ndNF (Second Formal Form),
3. 3rdNF (Third Normal Form)

3. 3rdNF (Third Normal Form),
 4. BCNF (Boyce Codd Normal Form) dan
-

5. 4thNF (Fourth Normal Form).

Untuk beberapa kasus ada juga yang sampai tahap 5thNF bahkan 6thNF.

Setiap tahap normalisasi tersebut memiliki kriteria yang menjadi ukuran apakah sebuah tabel memenuhi kriteria tahap normalisasi tersebut atau tidak. Jika tidak memenuhi kriteria maka dilakukan dekomposisi (pemecahan) tabel menjadi beberapa tabel yang lebih kecil yang memenuhi kriteria tahapan yang diuji. Proses tahapan normalisasi tersebut diperlihatkan pada gambar berikut:



Gambar 2. Proses Normalisasi

Jadi proses normalisasi dimulai dari sebuah tabel universal yang umumnya bersifat tidak normal seperti telah dijelaskan sebelumnya. Pertama dilakukan pengujian apakah memenuhi kriteria 1NF, jika tidak memenuhi dilakukan dekomposisi, sedangkan jika memenuhi maka lanjut ke tahapan berikutnya yaitu pengujian kriteria 2NF. Demikian selanjutnya dilakukan tahap demi tahap sebagai proses normalisasi. Untuk beberapa kasus proses normalisasi cukup sampai pada

Bahkan untuk kasus yang lebih sederhana cukup sampai tahap 2NF saja.

Ketergantungan Fungsional (KF)

Ketergantungan Fungsional (*Functional Dependency*) adalah gambaran hubungan ketergantungan antara item data atau atribut. Misal atribut A memiliki hubungan ketergantungan dengan B, maka dinotasikan dengan:

$A \rightarrow B$

Dibaca: nilai A menentukan nilai B, atau B memiliki KF terhadap A.

Artinya sebuah nilai pada atribut A akan menentukan nilai apa yang terdapat pada atribut B.

Misal ditentukan:

nim → nama_mahasiswa

Artinya sebuah nilai nim akan menentukan nilai nama_mahasiswa. Pada contoh tabel sebelumnya, jika diketahui nilai nim: 2020240063, maka otomatis akan diketahui nama mahasiswa: Rahma Hanni Izzati.

Contoh lain jika ditulis:

nama_mahasiswa --/→ nim

Dibaca: atribut nama_mahasiswa **tidak** menentukan atribut nim, jadi nama_mahasiswa dan nim tidak membentuk KF. Dalam hal ini nama_mahasiswa tidak menentukan nilai nim. Jadi jika diketahui nama Rahma Hanni Izzati maka belum tentu nim nya adalah 2020240063, karena bisa jadi ada mahasiswa lain yang memiliki nama yang sama yang nilai nimnya tentu saja berbeda.

Ada beberapa jenis ketergantungan, sebagian yang akan dibahas:

1. Ketergantungan penuh

2. Ketergantungan parsial

-
2. Ketergantungan parsial
 3. Ketergantungan transitif
-

First Normal Form (1 NF)

Sebuah tabel disebut memenuhi kriteria 1NF jika memenuhi hal berikut:

1. Atribut harus bernilai tunggal atau bersifat atomik (*single value*)
2. Tidak mengandung atribut yang domainnya sama

Contoh:

Tabel 2. Tabel kuliah1

id_mahasiswa	mahasiswa	matakuliah
1001	Hanni	java, Pyhton
1002	Dedi	java
1003	Budi	C, python

Pada tabel di atas terdapat 2 baris (*record*) yang nilai mata kuliahnya tidak bersifat single (atomic) yaitu baris 1 (java, python) dan baris 3 (C, pyhton). Ini tidak memenuhi kriteria 1NF yang pertama yaitu: Atribut harus bernilai tunggal atau bersifat atomik (*single value*).

Misal diperbaiki menjadi:

Tabel 3. Tabel kuliah 2

id_mahasiswa	mahasiswa	matakuliah1	matakuliah2
1001	Hanni	java	python
1002	Dedi	java	
1003	Budi	C	python

Tabel di atas tampak seakan memecahkan permasalahan sebelumnya, tetapi sebenarnya tidak karena menghasilkan permasalahan baru yaitu: atribut

matakuliah1 dan matakuliah2 memiliki nilai domain yang

matakuliah2 dipertukarkan akan memiliki makna yang sama (tidak mengubah arti).

Untuk mengatasi masalah domain yang sama ini maka tabel didekomposisi kembali menjadi 2 tabel seperti berikut:

Tabel 4. Tabel mahasiswa

id_mahasiswa	mahasiswa
1001	Hanni
1002	Dedi
1003	Budi

Tabel 5. Tabel kuliah

id_mahasiswa	matakuliah
1001	java
1001	python
1002	java
1003	C
1003	python

Hasil dekomposisi menjadi 2 tabel di atas (tabel mahasiswa dan tabel kuliah) telah menghasilkan tabel yang memenuhi kriteria 1NF.

Second Normal Form (2 NF)

Sebuah tabel disebut memenuhi kriteria 2NF jika memenuhi hal berikut:

-
1. Memenuhi 1NF
 2. Setiap atribut bukan *primary key* memiliki ketergantungan fungsional penuh pada *primary key*
-

Contoh:

Tabel 6. Tabel perkuliahan

nim	nama_mhs	kode_mk	nama_mk
1001	Hanni	KK05	database
1001	Hanni	KK07	python
1003	Budi	KK05	database

Primary Key pada tabel di atas bersifat komposit yaitu kombinasi nim dan kode_mk. Jika digambarkan hubungannya dengan KF:

nim+kode_mk → nama_mhs, nama_mk

dimana nama_mhs dan nama_mk merupakan atribut yang bukan primary key.

Selain KF di atas ternyata ada KF lain pada tabel di atas yaitu:

nim → nama_mhs

kode_mk → nama_mk

Artinya atribut bukan primary key (nama_mhs, nama_mk) selain memiliki ketergantungan (KF) terhadap primary key (nim+kode_mk) juga memiliki ketergantungan masing-masing nama_mhs terhadap nim dan nama_mk terhadap kode_mk. Hal ini yang disebut dengan tidak memenuhi kriteria 2NF.

Cara mengatasinya adalah dengan mendekomposisi tabel sesuai dengan jumlah KF nya sehingga tabelnya menjadi:

Tabel 7. Tabel mahasiswa

nim	nama_mhs
1001	Hanni
1001	Ungu

1001	Hanni
1003	Budi

Dari KF: nim → nama_mhs

Tabel 8. Tabel matakuliah

kode_mk	nama_mk
KK05	database
KK07	python
KK05	database

Dari KF: kode_ → nama_mk

Tabel 9. Tabel Perkuliahuan

nim	kode_mk
1001	KK05
1001	KK07
1003	KK05

Dari KF: nim+kode_mk→ nama_mhs, nama_mk

Third Normal Form (3 NF)

Sebuah tabel disebut memenuhi kriteria 3NF jika memenuhi hal berikut:

1. Memenuhi 2NF
2. Atribut yang bukan *primary key* tidak memiliki ketergantungan transitif pada *primary key*

Contoh:

Tabel 10. Tabel mahasiswa

nim	nama_mhs	id_pembimbing	nama_dosen
1001	Hanni	D01	Herianto
1002	Dedi	D02	Aji

1002	Dedi	D02	Adam
1003	Budi	D01	Herianto

Primary key pada tabel di atas adalah: nim, sehingga KF utamanya adalah:

nim → nama_mhs, id_pembimbing, nama_dosen

Tetapi di sisi lain juga adalah KF, yaitu:

id_pembimbing → nama_dosen

Terjadi ketergantungan transitif seperti berikut:

nim → id_pembimbing → nama_dosen

Ketergantungan transitif di sini adalah: nim menentukan id_pembimbing dan id_pembimbing menentukan nama_dosen.

Untuk mengatasinya maka tabel dapat didekomposisi dengan memisahkan ketergantungan transitif di atas menjadi:

Tabel: mahasiswa

nim	nama_mhs	id_pembimbing
1001	Hanni	D01
1002	Dedi	D02
1003	Budi	D01

KF: nim → nama_mhs, id_pembimbing

Tabel 11. Tabel: pembimbing

id_pembimbing	nama_dosen
D01	Herianto
D02	Adam
D01	Herianto

KF: id_pembimbing → nama_dosen

atas sudah tidak terjadi lagi ketergantungan transitif (memenuhi kriteria 3NF).

Boyce Codd Normal Form (BC NF)

Tahap ini BCNF ini kadang disebut juga dengan 3.5NF. Sebuah tabel disebut memenuhi kriteria BCNF jika memenuhi hal berikut:

1. Memenuhi 3NF
2. Setiap determinan dalam relasi tersebut adalah *candidate key*

Contoh:

Tabel 12. Tabel: Matakuliah

kode_mk	nm_matakuliah	nm_dosen	lama_mengajar
KK01	agama	Husen	10
KK05	database	adam	7
KK07	python	herianto	8

Primary key (determinan) pada tabel di atas adalah: **kode_mk**, dengan KF:

kode_mk → **nm_matakuliah, nm_dosen, lama_mengajar**

Sekilas tabel di atas sudah memenuhi 3NF.

Tetapi sebenarnya ada satu hubungan ketergantungan lain yaitu antara **nm_dosen** dengan **lama_mengajar**:

nm_dosen → **lama_mengajar**

Tetapi untuk KF di atas **nm_dosen** tidak termasuk candidate key karena tidak bersifat unik, sehingga fenomena seperti disebut tidak memenuhi BCNF.

Untuk mengatasinya maka didekomposisi dengan menambah sebuah tabel dan memberikan id baru

(id_dosen) ke tabel tersebut seperti berikut:

Tabel 13. Tabel: matakuliah

190

NORMALISASI BASIS DATA

kode_mk	nm_matakuliah	id_dosen
KK01	agama	DOS01
KK05	database	DOS02
KK07	python	DOS03

Tabel 14. Tabel: dosen

id_dosen	nm_dosen	lama_mengajar
DOS01	Husen	10
DOS02	adam	7
DOS03	herianto	8

Fourth Normal Form (4 NF)

Sebuah tabel disebut memenuhi kriteria 4NF jika memenuhi hal berikut:

1. Memenuhi BCNF
2. Tidak memiliki ketergantungan *multivalued*.

Contoh:

Tabel 15. Tabel: mahasiswa_kuliah

nim	nama_mhs	kode_mk	nama_mk	hobi
1001	Hanni	KK05	database	membaca
1001	Hanni	KK07	python	memasak
1003	Budi	KK05	database	Olah raga

Data pada tabel di atas hendak menyatakan bahwa mahasiswa bernama Hanni memiliki 2 hobi yaitu:

membaca dan memasak. Ini yang disebut dengan memiliki ketergantungan *multivalued*, atau tidak memenuhi kriteria 4NF.

Untuk mengatasinya data hobi yang bersifat multi value dipisahkan pada tabel sendiri seperti berikut:

Tabel: hobi

id_hobi	nm_hobi
H001	membaca
H002	memasak
H003	Olah raga

Tabel: hobi_mahasiswa

nim	id_hobi
1001	H001
1001	H003
1003	H004

Daftar Pustaka

Date, C. J. (2012). *Database Design and Relational Theory*. O'Reilly
<http://www.oreilly.de/catalog/9781449328016/index.html>

Date, C. J. (December 21, 2015). The New Relational Database Dictionary: Terms, Concepts, and Examples. "O'Reilly Media, Inc." ISBN 9781491951699.

Köhler, Henning; Link, Sebastian (2018-07-01). "SQL schema design: foundations, normal forms, and normalization". *Information Systems*.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011).

Profil Penulis



Herianto

Mulai mengajar sejak tahun 1999. Pendidikan master (S2) ditempuh di Universitas Gadjah Mada (UGM) bidang Sistem Komputer dan Informatika -Teknik Elektro. Mata Kuliah yang pernah diampu: Perancangan Basis Data, Jaringan Komputer, Data warehouse dan Data Mining, Artificial Intelligence, Pemrograman Internet dan sejumlah mata kuliah dasar Teknologi Informasi.

Secara Khusus penulis memiliki ketertarikan di bidang Artificial Intelligence dan Data Science. Untuk mewujudkan karir sebagai dosen profesional, keahlian penulis di bidang data science telah dinyatakan Kompeten melalui Sertifikasi BNSP pada tahun 2021. Penulis juga aktif sebagai peneliti di bidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI.

Email Penulis: heri.unsada@gmail.com

14

STUDI KASUS ERD DAN NORMALISASI

Donny Maulana, S.Kom., M.M.Si.

Universitas Pelita Bangsa

Studi Kasus ERD

Sebuah perusahaan mempunyai beberapa bagian. Masing-masing bagian mempunyai pengawas dan setidaknya satu pegawai. Pegawai ditugaskan paling tidak di satu bagian (dapat pula dibeberapa bagian). Paling tidak satu pegawai mendapat tugas di satu proyek. Tetapi seorang pegawai dapat libur dan tidak dapat tugas di proyek.

Menentukan Entitas

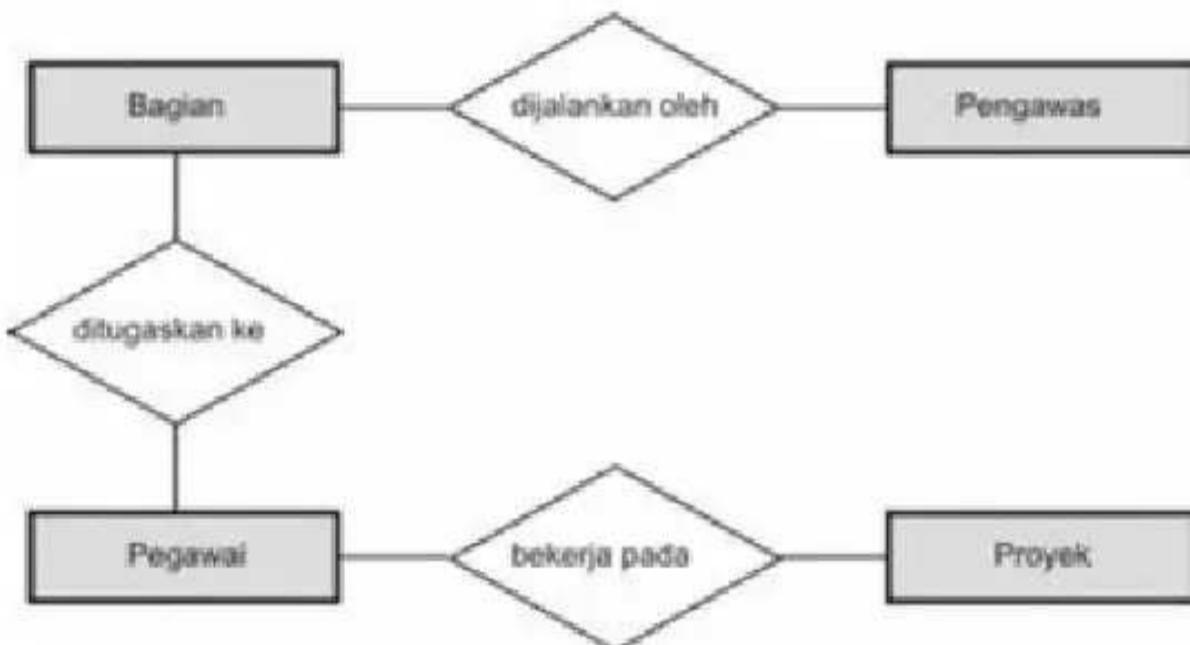
Entitasnya: pengawas, bagian, pegawai, proyek

Menentukan Relasi dengan Matrik Relasi

Tabel 12.1 Relasi dengan matrik relasi

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

Gambar ERD Sementara

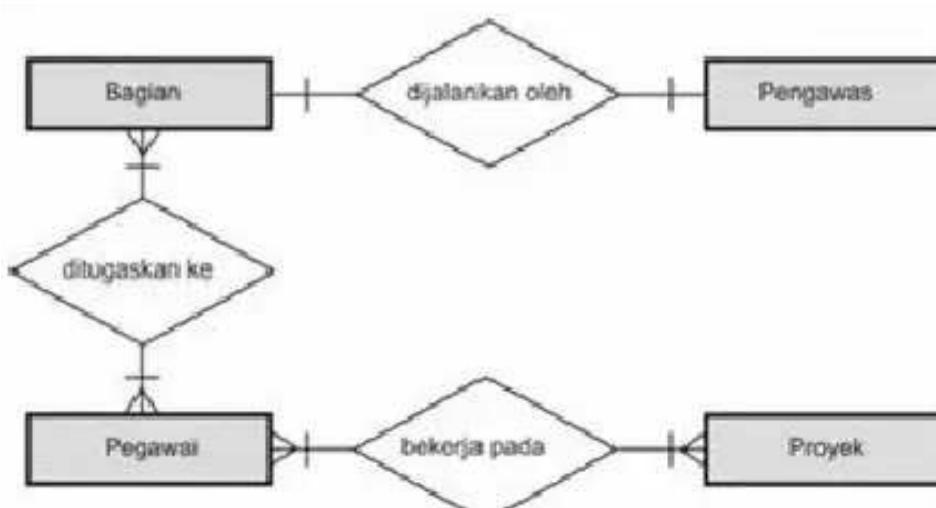


Gambar 12.1 ERD sementara

Dari gambaran permasalahan dapat diketahui bahwa:

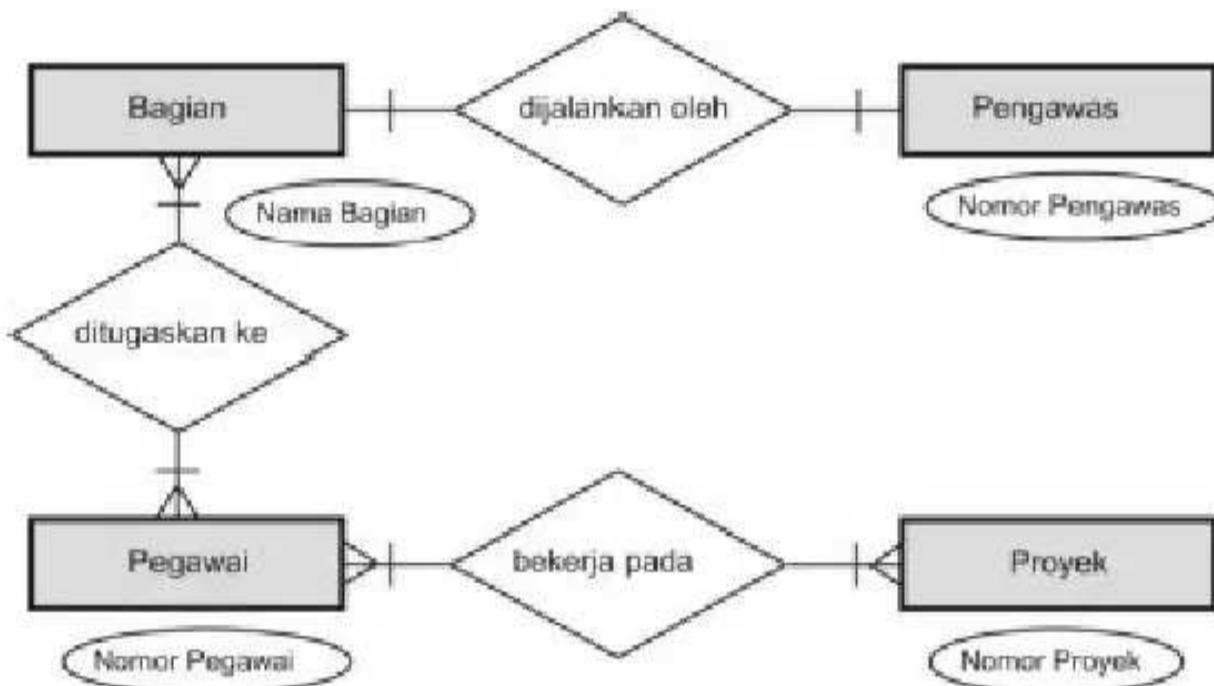
1. masing-masing bagian hanya punya satu pengawas
2. seorang pengawas bertugas di satu bagian
3. masing-masing bagian ada minimal satu pegawai
4. masing-masing pegawai bekerja paling tidak di satu bagian
5. masing-masing proyek dikerjakan paling tidak oleh satu pegawai

Gambaran Kardinalitas



Menentukan Kunci Utama

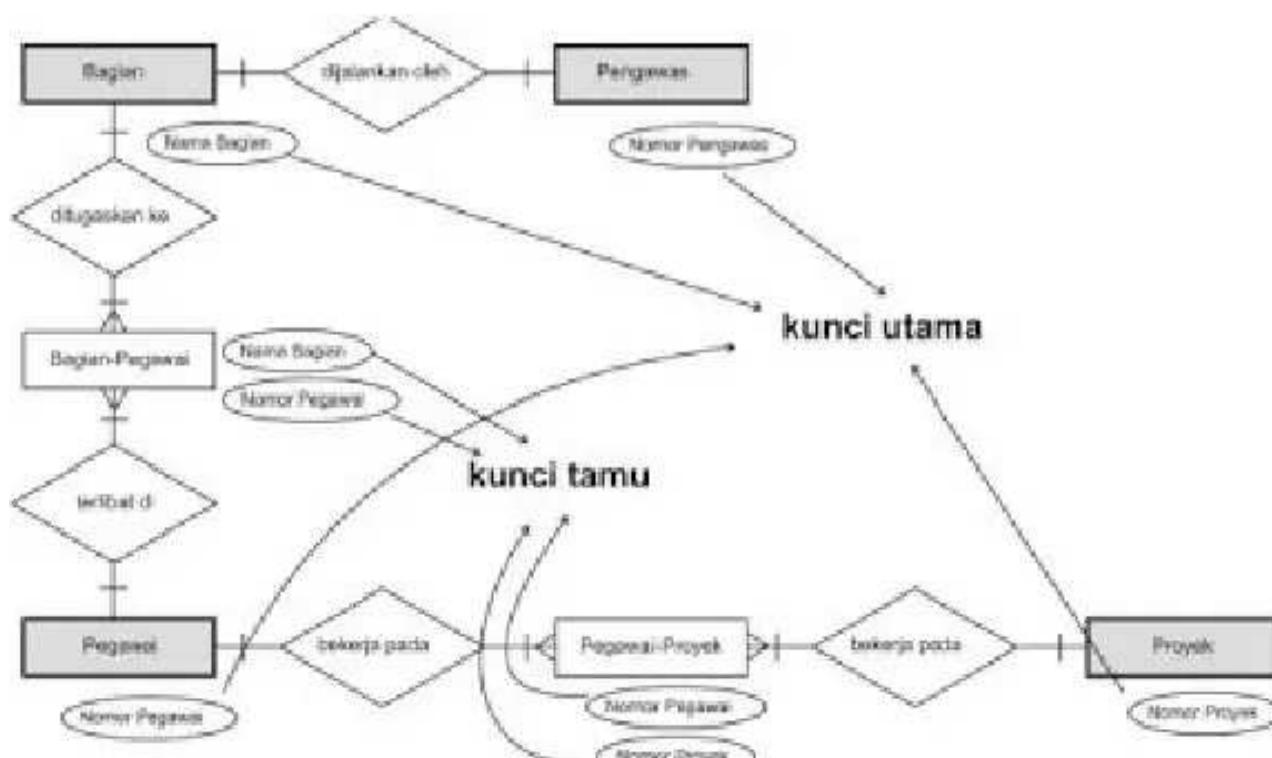
Kunci utamanya: Nomor Pengawas, Nama Bagian, Nomor Pegawai, Nomor Proyek.



Gambar 12.3 Menetukan Kunci Utama

Mengambarkan ERD Berdasarkan Kunci

Ada dua relasi many to many pada ERD sementara, yaitu antara bagian dengan pegawai, pegawai dengan proyek, oleh sebab itu kita buat entitas baru yaitu bagian -pegawai dan pegawai-proyek Kunci utama dari entitas baru adalah kunci utama dari entitas lain yang akan menjadi kunci tamu di entitas yang baru.



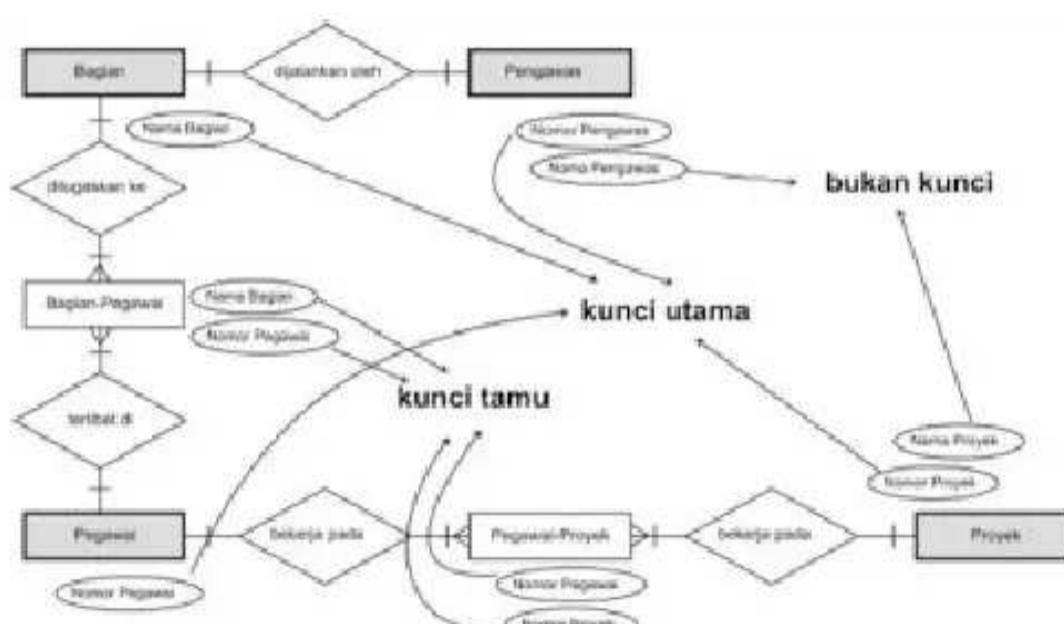
Menentukan Atribut

Atribut yang diperlukan adalah: nama bagian, nama proyek, nama pegawai, nama pengawas, nomor proyek, nomor pegawai, nomor pengawas.

Memetakan Atribut

1. Bagian: Nama bagian
2. Proyek: Nama proyek
3. Pegawai: Nama pegawai
4. Pengawas: Nama pengawas
5. Proyek-Pegawai: Nomor proyek, Nomor pegawai
6. Pengawas: Nomor pengawas

Menggambarkan ERD dengan Atribut



Gambar 12.5 ERD dengan atribut

Studi Kasus Normalisasi (1)

PT. Dennis Makmur
Jl. Sekeloa Utara No 62/152 C
Bandung

FAKTUR PEMBELIAN

Kode Pemasok : A101

Tanggal : 07/03/2004

Nama Pemasok : Akbar Comp.

Nomor : 111

Kode	Nama Barang	Jumlah	Harga	Total
A1	DD RAM 128	10	200.000	2.000.000
A2	GForce-FX 5200	10	500.000	5.000.000
A3	Athlon 2500+	10	700.000	7.000.000
Total faktur				14.000.000

STUDI KASUS ERD DAN NORMALISASI

Un-Normalized

NoFaktur	Tgl	KdPemasok	NmPemasok	KdBrg	NmBarang	Qty	Hrg	Total	TotFak	Tempo
111	07/03/04	A101	Akbar Comp	A1	DDRAM128	10	200	2000	14000	07/04/04
				A2	GForce5200	10	500	5000		
				A3	Athlon2500	10	700	7000		
222	10/03/04	B102	Bona Comp	A2	GForce5200	5	500	2500	3700	10/04/04
				A5	DDRAM512	3	400	1200		

1 NF

NoFaktur	Tgl	KdPemasok	NmPemasok	KdBrg	NmBarang	Qty	Hrg	Total	TotFak	Tempo
111	07/03/04	A101	Akbar Comp	A1	DDRAM128	10	200	2000	14000	07/04/04
111	07/03/04	A101	Akbar Comp	A2	GForce5200	10	500	5000	14000	07/04/04
111	07/03/04	A101	Akbar Comp	A3	Athlon2500	10	700	7000	14000	07/04/04
222	10/03/04	B102	Bona Comp	A2	GForce5200	5	500	2500	3700	10/04/04
222	10/03/04	B102	Bona Comp	A5	DDRAM512	3	400	1200	3700	10/04/04

2 NF

Tabel Pemasok (KdPemasok → NmPemasok)

KdPemasok	NmPemasok
A101	Akbar Comp
B102	Bona Comp

Tabel Barang (KdBrg → (NmBarang,Hrg))

KdBrg	NmBarang	Hrg
A1	DDRAM128	200
A2	GForce5200	500
A3	Athlon2500	700
A2	GForce5200	500
A5	DDRAM512	400

Tabel Faktur ((NoFaktur,KdPemasok,KdBrg)→(Tgl,Qty,Total,TotFak,Tempo))

NoFaktur	Tgl	KdPemasok	KdBrg	Qty	Total	TotFak	Tempo
111	07/03/04	A101	A1	10	2000	14000	07/04/04
111	07/03/04	A101	A2	10	5000	14000	07/04/04
111	07/03/04	A101	A3	10	7000	14000	07/04/04
222	10/03/04	B102	A2	5	2500	3700	10/04/04
222	10/03/04	B102	A5	3	1200	3700	10/04/04

STUDI KASUS ERD DAN NORMALISASI

3 NFTabel Pemasok ($\text{KdPemasok} \rightarrow \text{NmPemasok}$)

KdPemasok	NmPemasok
A101	Akbar Comp
B102	Bona Comp

Tabel Barang ($\text{KdBrg} \rightarrow (\text{NmBarang}, \text{Hrg})$)

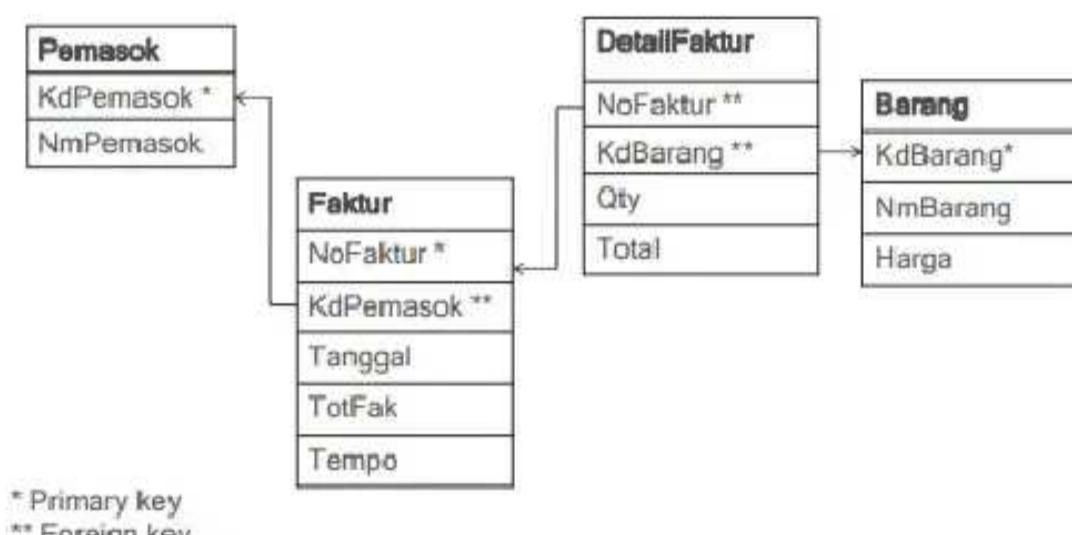
KdBrg	NmBarang	Hrg
A1	DDRAM128	200
A2	GForce5200	500
A3	Athlon2500	700
A2	GForce5200	500
A5	DDRAM512	400

Tabel Faktur ($(\text{NoFaktur}, \text{KdPemasok}) \rightarrow (\text{Tgl}, \text{TotFak}, \text{Tempo})$)

NoFaktur	Tgl	KdPemasok	TotFak	Tempo
111	07/03/04	A101	14000	07/04/04
222	10/03/04	B102	3700	10/04/04

Tabel Detail Faktur ($(\text{NoFaktur}, \text{KdBrg}) \rightarrow (\text{Qty}, \text{Total})$)

NoFaktur	KdBrg	Qty	Total
111	A1	10	2000
111	A2	10	5000
111	A3	10	7000
222	A2	5	2500
222	A5	3	1200

Skema Relasi**Studi Kasus Normalisasi (2)****Un-Normalized**

NIM	NamaMhs	Jurusan	KodeMK	NamaMK	KodeDosen	NamaDosen	Nilai
2683	Willi	MI	MI350	Manajemen Sistem Informasi	104	Dita	A
			MI240	Basis Data	317	Budi	B
5432	Bakri	AK	AK201	Akuntasi Dasar	219	Deni	A
			AK302	Pemasaran	280	Rini	A
			AK304	Manajemen Keuangan	211	Weni	C

1 NF

NIM	NamaMhs	Jurusan	KodeMK	NamaMK	KodeDosen	NamaDosen	Nilai
2683	Willi	MI	MI350	Manajemen Sistem Informasi	104	Dita	A
2683	Willi	MI	MI240	Basis Data	317	Budi	B
5432	Bakri	AK	AK201	Akuntasi Dasar	219	Deni	A
5432	Bakri	AK	AK302	Pemasaran	280	Rini	A
5432	Bakri	AK	AK304	Manajemen Keuangan	211	Weni	C

2 NF**Tabel Mahasiswa**

NIM → {NamaMhs, Jurusan}

NIM	NamaMhs	Jurusan
2683	Willi	MI
5432	Bakri	AK

Tabel Nilai

(NIM, KodeMK) → Nilai

NIM	KodeMK	Nilai
2683	MI350	A
2683	MI240	B
5432	AK201	A
5432	AK302	A
5432	AK304	C

Tabel Kuliah

KodeMK → {NamaMK, KodeDosen}

KodeDosen → NamaDosen

KodeMK	NamaMK	KodeDosen	NamaDosen
MI350	Manajemen Sistem Informasi	104	Dita
MI240	Basis Data	317	Budi
AK201	Akuntasi Dasar	219	Deni
AK302	Pemasaran	280	Rini
AK304	Manajemen Keuangan	211	Weni

3 NF**Tabel MataKuliah**

KodeMK → {NamaMK, KodeDosen}

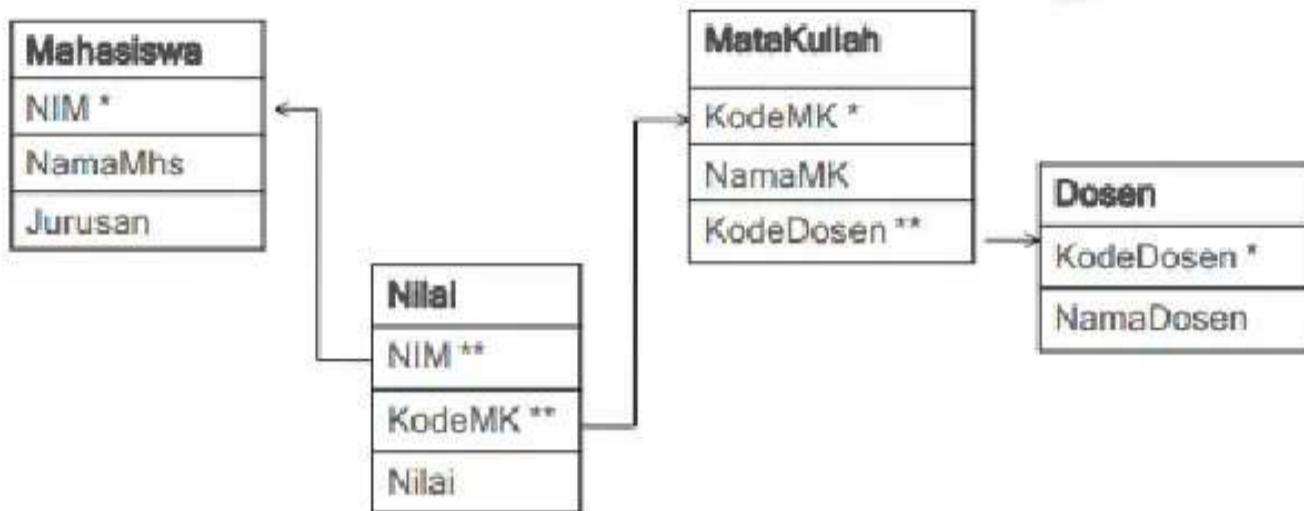
KodeMK	NamaMK	KodeDosen
MI350	Manajemen Sistem Informasi	104
MI240	Basis Data	317
AK201	Akuntasi Dasar	219
AK302	Pemasaran	280
AK304	Manajemen Keuangan	211

Tabel Dosen

KodeDosen → NamaDosen

KodeDosen	NamaDosen
104	Dita
317	Budi
219	Deni
280	Rini
211	Weni

Skema Relasi



* Primary key

** Foreign key

Daftar Pustaka

- Rerung, R. R., Fauzan, M., & Hermawan, H. (2020). Website Quality Measurement of Higher Education Services Institution Region IV Using Webqual 4.0 Method. *International Journal of Advances in Data and Information Systems*, 1(2), 89-102.
- Indrajani. (2015). *Database Design (Case Study All in One)*. Jakarta: PT Elex Media Komputindo.
- Saputra, Agus. (2014). Proyek Membuat Website Periklanan dengan PHP. Cirebon: CV. ASFA Solution.
- Sutanta, Edhy. (2011). Basis Data dalam Tinjauan Konseptual. Yogyakarta: ANDI Yogyakarta.

Profil Penulis

Donny Maulana



Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 1991 silam. Hal tersebut membuat penulis memilih untuk masuk ke Sekolah Menengah Atas di SMAN 1 Cibinong dengan memilih Jurusan A1 (Fisika) dan berhasil lulus pada tahun 1994. Penulis kemudian melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S1 di prodi ILMU KOMPUTER UNIVERSITAS GUNADARMA DEPOK pada tahun 1999. Dua tahun kemudian, penulis menyelesaikan studi S2 di prodi SISTEM INFORMASI MANAGEMEN PROGRAM PASCA SARJANA UNIVERSITAS GUNADARMA DEPOK.

Penulis memiliki pengalaman dalam bidang Technical Support, System Administrator, kepakaran dibidang Web Technology dan Mobile Programming. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI. Selain peneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi kemajuan bangsa dan negara yang sangat tercinta ini. Atas dedikasi dan kerja keras dalam menulis buku.

Email Penulis: danny.maulana@pelitabangsa.ac.id

Indonesia
menulis

- 1 PENGANTAR BASIS DATA
Ayu Manik Dirgayusari, S.Kom., M.MT.
- 2 LINGKUNGAN BASIS DATA
Nazaruddin Ahmad, M.T.
- 3 MODEL BASIS DATA RELASIONAL
Bagus Tri Mahardika, M.MSI.
- 4 REALASIONAL KEY (SUPER KEY, CANDIDAT KEY, PRIMARY KEY, ALTERNATIF)
Musyrifah, S.Pd., M.Pd.
- 5 BAHASA PADA MODEL RELASIONAL
Husna Gemasih, S.Inf., M.Cs.
- 6 PENGENALAN SQL BESERTA CONTOHNYA
Asmawati S, S.Kom., M.Pd.
- 7 ADVENCED SQL (EMBEDDED DAN DYNAMIC)
Asep Abdul Sofyan, S.Kom., M.Kom.
- 8 RDBMS (RELATIONAL DATABASE MANAGEMENT SYSTEM)
Djamaludin, S.Kom., M.Kom.
- 9 PENGENALAN DATABASE 2
Nurul Aini, S.Kom., M.T.
- 10 PENGENALAN ORACLE
Wiyanto, S.Kom., M.Kom.
- 11 KONSEP DASAR MODEL ER
Mohammad Ridwan, S.Kom., M.Kom.
- 12 TRANSFORMASI ER KE MODEL DATA RELASIONAL
Muhammad Yani, S.Kom., M.T.I.
- 13 NORMALISASI BASIS DATA
Herianto, S.Pd., MT.
- 14 STUDI KASUS ERD DAN NORMALISASI
Donny Maulana, S.Kom., M.M.Si.

Editor :

Dudih Gustian, S.T., M.Kom.

