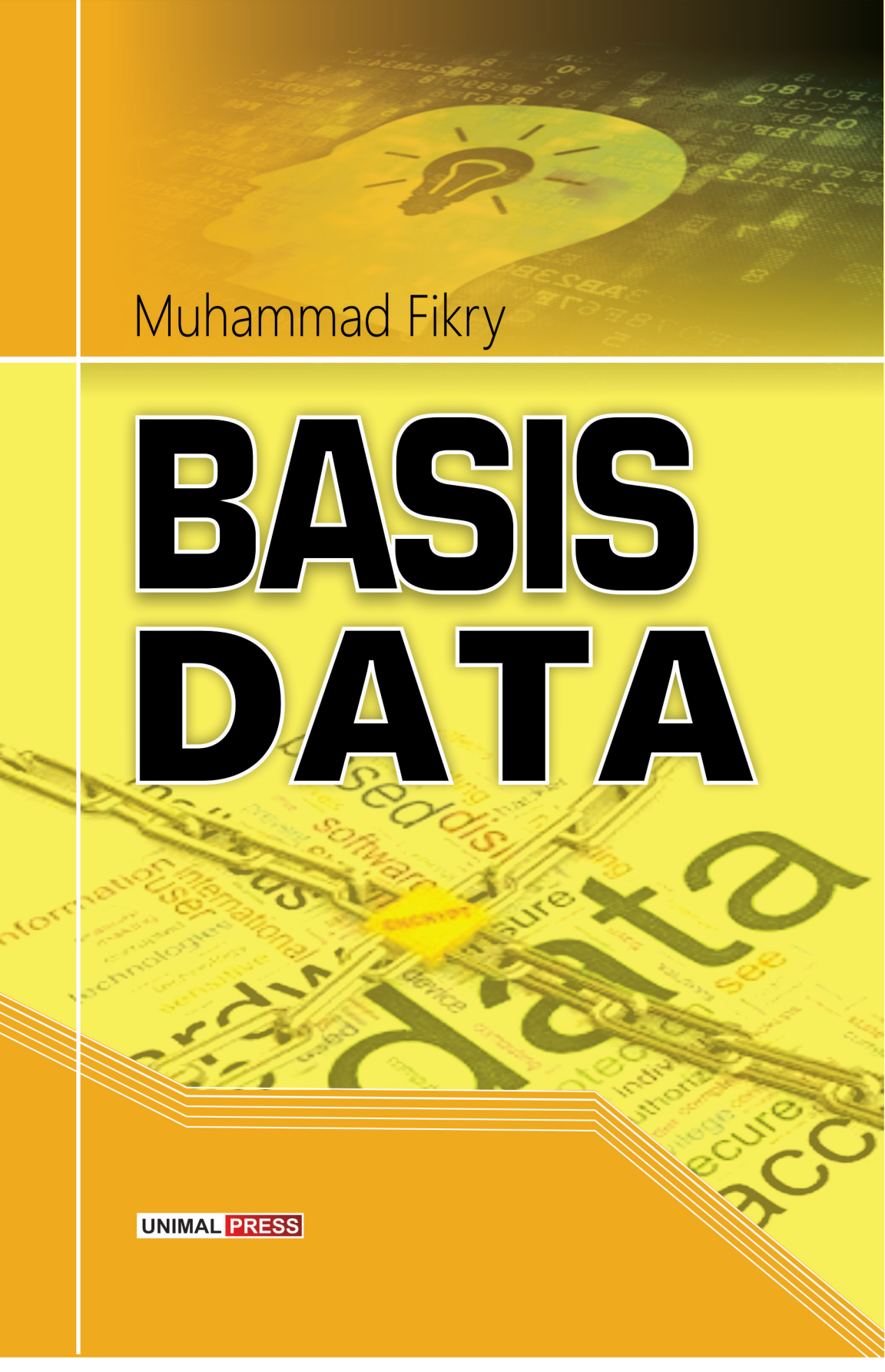


Muhammad Fikry

BASIS DATA

UNIMAL PRESS

BASIS DATA


universitas
MALIKUSSALEH

Muhammad Fikry

BASIS DATA

UNIMAL PRESS

Judul:**BASIS DATA**
xii + 88 hal., 15 cm x 23 cm

Cetakan Pertama: Maret, 2019

Penulis:
Muhammad Fikry

Perancang Sampul &
Penata Letak: **Eriyanto**
Pracetak dan Produksi: **Unimal Press**

Penerbit:

UNIMAL PRESS

Unimal Press
Jl. Sulawesi No.1-2
Kampus Bukit Indah Lhokseumawe 24351
PO.Box. 141. Telp. 0645-41373. Fax. 0645-44450
Laman: www.unimal.ac.id/unimalpress.
Email: unimalpress@gmail.com

ISBN:
978 – 602 –464- 078-1

ISBN 978-602-464-078-1
9 786024 640781

Kata Pengantar

Syukur Alhamdulillah kami panjatkan ke hadirat Allah SWT, atas berkat rahmat dan kemudahan yang diberikan sehingga buku yang berjudul "Basis Data" ini terselesaikan.

Buku ini ditulis untuk menjelaskan tentang materi data basis data yang mudah dimengerti dan dipelajari bagi pemula yang ingin mempelajari tentang cara membangun *database*.

Isi dari buku ini tentu jauh dari sempurna dan belum lengkap, karena isinya masih berupa penjelasan-penjelasan dasar membangun basis data. Diharapkan pembaca terus mengembangkan desain dan pembangunan basis data untuk lebih menarik dan sempurna.

Akhir kata semoga buku ini bisa membantu pembaca dalam mempelajari Basis Data dan bermanfaat bagi kita semua. Amin.

Lhokseumawe, Maret 2019

Penulis

This page is intentionally left blank

Daftar Isi

Daftar Tabel

Daftar Gambar

This page is intentionally left blank

BAB 1
KONSEP BASIS DATA

1.1 Perkembangan Basis Data

Pada awal tahun 1960, Charles Bachman di perusahaan General Electric mendesain generasi pertama DBMS yang disebut Penyimpanan Data Terintegrasi (Integrated Data Store). Dasar untuk model data jaringan terbentuk lalu distandarisasi oleh Conference on Data System Languages (CODASYL). Kemudian, Bachman menerima CM Turing Award (penghargaan semacam nobel pada ilmu komputer) tahun 1973.

Pada akhir tahun 1960-an, IBM mengembangkan sistem manajemen informasi (*Information Management System*). Hasil kerjasama antara IBM dengan perusahaan penerbangan Amerika mengembangkan sistem SABRE. Sistem SABRE memungkinkan user mengakses data yang sama pada jaringan komputer. Pada tahun 1970, Edgar Codd di laboratorium penelitian di San Jose mengusulkan suatu representasi data baru yang disebut model data relasional.

Pada tahun 1980, model relasional menjadi paradigma DBMS paling dominan. Bahasa query SQL (*Structured Query Language*) dikembangkan untuk basis data relasional sebagai bagian proyek Sistem R dari IBM. SQL distandarisasi di akhir tahun 1980 dan SQL-92 diadopsi oleh American National Standards Institute (ANSI) dan International Standards Organization (ISO). Program yang digunakan untuk eksekusi bersamaan dalam basisdata disebut transaksi.

Pada akhir tahun 1980 dan permulaan tahun 1990, banyak bidang sistem basisdata dikembangkan. Penelitian di bidang basisdata meliputi bahasa query yang powerful, model data lengkap, dan penekanan pada dukungan analisis data yang kompleks, sistem diperluas dengan kemampuan menyimpan tipe data baru misalnya image dan text serta kemampuan query yang kompleks.

1.2 Definisi

Basis adalah Gudang / markas / tempat berkumpul / tempat bersarang. Data adalah Representasi fakta dunia nyata yang mewakili suatu obyek (manusia, benda, kejadian, dll) yang disimpan dalam bentuk teks, angka, gambar, bunyi, simbol, atau kombinasinya

Basis Data adalah Kumpulan dari item data yang saling berhubungan satu dengan lainnya yang diorganisasikan berdasar sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* digunakan untuk melakukan manipulasi data (diperbaharui, dicari, diolah dengan perhitungan-perhitungan tertentu, dan dihapus) dengan tujuan tertentu

1.3 Sistem Basis Data

Sistem basis data adalah sistem yang terdiri atas kumpulan tabel data yang saling berhubungan dan kumpulan program yang memungkinkan beberapa pemakai atau program lain untuk mengakses dan memanipulasi tabel tabel data tersebut (Fathansyah, 1999).

Dalam Sistem Basis data memiliki beberapa komponen yaitu:

1. Perangkat Keras (*Hardware*)

   Perangkat keras yang biasanya terdapat dalam sistem basis data adalah memori sekunder hardisk.

2. Sistem Operasi (*Operating System*)

3. Sistem Operasi (*Operating System*) merupakan program yang mengaktifkan atau mengfungsikan sistem komputer, mengendalikan seluruh sumber daya (resource) dan melakukan operasi-operasi dalam komputer. Sistem Operasi yang banyak digunakan. Basis data (*Database*)

   Sebuah basis data (*Database*) dapat memiliki sekumpulan data yang terorganisir dengan baik sehingga data tersebut mudah disimpan, diakses, dan juga dapat dimanipulasi. Setiap basis data dapat berisi atau memiliki sejumlah objek basis data seperi file atau tabel.

4. Management System (DBMS)

   Pengolahan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak yang disebut DBMS yang menentukan bagaimana data disimpan, diubah dan diambil kembali.

5. Pemakai (*User*)

   Bagi pemakai dapat berinteraksi dengan basis data dan memanipulasi data dalam program yang ditulis dalam bahasa pemograman.

6. Aplikasi atau Perangkat Lain

   Aplikasi ini tergantung kebutuhan, pemakai basis data bisa dibuatkan program khusus untuk melakukan pengisian, pengubahan atau pengambilan data yang mudah dalam pemakaianya. Program tersebut ada yang tersedia langsung dalam DBMS atau dibuat menggunakan Bahasa pemrograman.

Tujuan Basis Data

a. Mengatur data/mengorganisasikan data agar diperoleh kemudahan, ketepatan, dan kecepatan dalam pengambilan kembali
b. Tidak ada duplikasi data sehingga konsistensi data mudah dijaga
c. Data terintegrasi
d. Data tidak tergantung pada program aplikasi, sehingga pemeliharaan program aplikasi mudah dilakukan
e. Data dapat dipakai secara bersama oleh beberapa pemakai
f. Dapat diterapkan standarisasi
g. Informasi selalu mutakhir (*up to date*)
h. untuk memenuhi kebutuhan-kebutuhan konten informasi dari pengguna dan aplikasi-aplikasi tertentu
i. menyediakan struktur informasi yang alami dan mudah dipahami
j. mendukung kebutuhan-kebutuhan pemrosesan dan objektifitas kinerja (waktu respon, waktu pemrosesan, dan ruang penyimpanan)

Manfaat Basis Data

1. Kecepatan dan Kemudahan (*Speed*)

   Memungkinkan kita untuk dapat menyimpan dan melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data tersebut dengan lebih cepat dan mudah.

2. Kebersamaan Pemakaian (*Sharability*)

Pemakai basisdata tidak terbatas, pengisian data dapat dilakukan oleh beberapa orang dalam satu lokasi.

3. Pemusatan Kontrol Data (*Control*)

   Data yang ada menjadi terpusat pada satu tempat penyimpanan. Sehingga kita dapat mengaksesnya kapan saja.

4. Efisiensi Ruang Penyimpanan (*Space*)

   Tidak adanya redundansi data sehingga efisiensi/optimalisasi penggunaan ruang penyimpanan dapat dilakukan. Penekanan jumlah redundasi data, dilakukan dengan menerapkan sejumlah pengkodean atau membuat relasi antar kelompok data yang saling berhubungan.

5. Keakuratan (*Accuracy*)

   Pemanfaatan pengkodean dengan batasan tertentu, yang membuat satu data menjadi unik dan berbeda dengan yang lain, sehingga ketika menyimpan data tidak akan ada data yang sama dalam penyimpanan.

6. Ketersediaan (*Availability*)

   Karena kepentingan pemakaian data, sebuah basis data dapat memiliki data yang disebar di banyak lokasi. Dengan pemanfaatan teknologi jaringan computer, data nasabah yang berada di suatu cabang sebuah bank dapat diakses (menjadi tersedia/availibility) di cabang lainnya.

7. Kelengkapan (*Completeness*)

   Data yang di input ke dalam sebuah basis data memiliki ruang yag besar sehingga data dapat dimasukkan dalam jumlah yang banyak sesuai dengan kebutuhan pengguna.

8. Keamanan (*Security*)

   Adanya *password* setiap pemakai basis data. Kita juga dapat menentukan siapa saja yang boleh mengakses data penting atau data biasa

9. Kemudahan dalam Pembuatan Program Aplikasi Baru

   Data yang disimpan dalam di eksport ke program aplikasi lain dengan menjamin terjaga/terpeliharanya data.

10. *User View*

    Pemakai dapat melihat langsung bentuk tampilan penginputan data, sehingga memudahkan pemakai dlm mengelola data.

Resiko Pemakaian Basis Data

a) Perlu personel khusus
b) Perlu perangkat lunak, bahkan perangkat keras khusus
c) Perlu *BackUp* eksplisit
d) Konflik pada data yang dipakai bersamaan
e) Perlu konsensus antara organisasi yang memakai basis data.

Operasi Dasar Basis Data

1. Pembuatan Basis data
2. Penghapusan Basis data
3. Pembuatan file/tabel
4. Penghapusan file/tabel
5. Pengubahan tabel
6. Penanbahan data
7. Pengambilan data
8. Penghapusan data

Tahap Pengumpulan dan Analisis Data

Untuk menentukan kebutuhan-kebutuhan suatu sistem basis data, pertama harus mengenal bagian-bagian lain dari sistem informasi yang akan berinteraksi dengan sistem basis data, termasuk para pemakai yang ada dan para pemakai yang baru, serta aplikasi-aplikasinya. Kebutuhan-kebutuhan dari para pemakai dan aplikasi inilah yang kemudian dikumpulkan dan dianalisa.

Tahap pertama, kita menentukan kelompok pemakai dan bidang-bidang aplikasinya yang artinya kita menentukan aplikasi utama dan kelompok pengguna yang akan menggunakan basis data. Individu utama pada tiap-tiap kelompok pemakai dan bidang aplikasi yang telah dipilih merupakan peserta utama pada langkah-langkah berikutnya dari pengumpulan dan spesifikasi data.

Tahap kedua, kita lakukan peninjauan dokumentasi yang ada, berarti dokumen yang ada yang berhubungan dengan aplikasi-aplikasi kita pelajari dan dianalisa. Dokumen-dokumen lainnya

(seperti: kebijakan-kebijakan, form, report, dan bagan organisasi) diuji dan ditinjau kembali untuk menguji apakah dokumen-dokumen tsb berpengaruh terhadap kumpulan data dan proses spesifikasi.

Tahap ketiga, analisa lingkungan operasi dan pemrosesan data mengartikan informasi yang sekarang dan yang akan datang diperinci dan dipelajari. Termasuk juga analisa jenis-jenis transaksi dan frekuensi-frekuensi transaksinya dan juga arus informasi dalam sistem. Informasi tersebut berupa input-output data.

Tahap keempat, kita membuat dan menyusun daftar pertanyaan dan wawancara yang dimana jawaban pertanyaan – pertanyaan yang telah dikumpulkan dari para pemakai basis data yang berpotensi akan diberikan kepada ketua kelompok (individu utama) untuk dapat diwawancarai sehingga input yang banyak dapat diterima dari mereka dengan memperhatikan informasi yang berharga dan mengadakan prioritas.

1.4 Database Management System

DBMS adalah Sistem pengorganisasian dan pengolahan *database* pada komputer. Sistem ini dirancang untuk mampu melakukan berbagai data dengan beberapa referensi data yang sama. DBMS ini mampu diakses oleh berbagai aplikasi.

Terobosan dari DBMS adalah *Relational Database Management System* (RDBMS) yang mengorganisasikan data dalam suatu struktur dan memaksimalkan berbagai cara serta menghubungkan antar kumpulan data yang disimpan dalam *database*.

Terobosan berikutnya adalah *Distributed Relational Database Management System* (DRDBMS). Dengan DRDBMS memungkinkan informasi berada pada baris data di lokasi yang berbeda (didistribusikan), dan direferensisilangkan, diperbaharui, dan akses dari semua lokasi, seolah-olah data tersebut berbasis data tunggal dan terpusat. Ada banyak perusahaan yang menawarkan DBMS itu sendiri, seperti Oracle, MySQL, Microsoft Access, MySQL-Front, SQLite dan semacamnya.

DBMS merupakan perangkat lunak yang bersifat general-purpose yang memiliki fasilitas proses *define*, *construct* dan *manipulate* basisdata untuk aplikasi yang bervariasi.

1. *Define*

fungsi untuk melakukan spesifikasi tipe data, struktur dan constraint data yang akan disimpan dalam basis data

2. *Construct*

   fungsi untuk melakukan proses penyimpanan data ke dalam beberapa media penyimpanan yang dikontrol DBMS

3. *Manipulate*

   fungsi untuk melakukan query atau memanggil data, update data dan menghasilkan laporan yang berasal dari basis data

Perbedaan Basis Data dengan DBMS

Basis Data dan DBMS sangat diperlukan dalam konsep basis data, namun secara artian keduanya sangat berbeda, jika Basis Data adalah Data / *Record* dari data - data, sedangkan DBMS adalah aplikasi untuk mengelola Data / *Record* tersebut, dan juga untuk menambahkan data, menghapus, mengubah dan menampilkan database itu sendiri.

Komponen DBMS

Hardware
Perangkat keras yang digunakan, tergantung dengan kebutuhan sistem dan kegunaan DBMS itu sendiri.
Software
Perangkat lunak disini adalah DBMS berikut sistem operasinya, termasuk perangkat lunak pendukung jaringan jika dababase tersebut terletak dalam suatu jaringan.

Procedure
Instruksi dan pengaturan yang mengatur desain dan penggunaan database:

a. penggunaan program aplikasi dan fasilitas yang ada
b. membuat *back up database*
c. dsb

Elemen Utama DBMS
Ada beberapa elemen utama dari *Data Base Management System*

1. Data pengguna
2. Metadata

3. Indeks
4. Metadata aplikasi

1. DATA PENGGUNA

Hampir semua database merepresentasikan data pengguna sebagai relasi dengan menganggapnya sebagai tabel data. Kolom dalam tabel berisi field-field atau atribut dan baris tabel berisi record/tuple (rekaman).

Relasi ini dapat digambarkan dengan bentuk hubungan antara pelajar dengan guru sebagai berikut:

NamaPelajar	TeleponPelajar	NamaGuru	TeleponGuru
Aminudin	7778889	Pardi	7789665
Usman	7896532	Pardi	7789665
Ari	7474856	Dadang	8965555
Rina	7895654	Marni	4562211
Tuti	7897744	Dadang	8965555
Joni	7845644	Dadang	8965555

Tabel 1.1 Relasi Hubungan Antara Pelajar dengan Guru

NamaPelajar	TeleponPelajar	NamaGuru
Aminudin	7778889	Pardi
Usman	7896532	Pardi
Ari	7474856	Dadang
Rina	7895654	Marni
Tuti	7897744	Dadang
Joni	7845644	Dadang

NamaGuru	TeleponGuru
Pardi	7789665
Dadang	8965555
Marni	4562211

Tabel 1.2 Relasi Hubungan Antara Pelajar dengan Guru

2. METADATA

Metadata berisi struktur dari suatu tabel. Metadata disebut dengan system tables. Tabel sistem ini berisi daftar tabel-tabel di dalam suatu database dan tabel yang berisi kolom-kolom pada suatu table.

Nama Tabel	Jumlah Kolom	Primary Key
Pelajar	4	NIS
Guru	3	NIP
Mata Pelajaran	4	Kode_MP
Relasi Belajar	3	{NIS,Kode_MP,NIP}

Tabel 1.3 SVS Table

Nama Kolom	Nama Tabel	Tipe Data	Panjang
NIS	Pelajar	String	5
Nama	Pelajar	String	20
Telepon	Pelajar	String	12
Alamat	Pelajar	String	50
NIP	Guru	String	6
Nama	Guru	String	20
Telepon	Guru	String	12
Divisi	Guru	String	20
Kode_MP	Mata Pelajaran	String	5
Nama MP	Mata Pelajaran	String	15
Jumlah Jam	Mata Pelajaran	Integer	4
NIS	Relasi Belajar	String	5
Kode_MP	Relasi Belajar	String	5
NIP	Relasi Belajar	String	6
Tingkat	Relasi Belajar	String	2

Tabel 1.4 SVS Columns

3. INDEX

Index digunakan untuk meningkatkan kinerja dan akses suatu database. Tipe data ini disebut dengan overhead data, terdiri dari prinsip-prinsip index serta beberapa penggunaan struktur data linked list. Apabila suatu tabel mengalami pengubahan data maka tabel indeks juga mengalami pengubahan

NO	Nama	Jurusan	Kelas
10	David Carradine	Akuntansi	2AB
20	Jaka Sembung	Manajemen	2CV
30	Kebo Ireng	Manajemen	2CV
40	Lasmini	Teknik Sipil	1SP
50	Joni Keboy	Akuntansi	1AB
60	Franc De Nero	Manajemen	2AB
70	Marco Van Basten	Teknik Sipil	1SP
80	Maradani	Teknik Sipil	1SP
90	Dona Doni	Akuntansi	1AB

Kelas	No
1AB	50,90
2AB	10,60
2CV	20,30
1SP	40,70,80

Table 1.5 Indeks Dari Kelas

Jurusan	No
Akuntansi	10,50,90
Manajemen	20,30,60
Teknik Sipil	40,70,80

Tabel 1.6 Indeks Dari Jurusan

4. Aplikasi METADATA

Application metadata digunakan untuk menyimpan struktur dan format dari user forms, report, queries dan komponen-komponen aplikasi lainnya.

Pemilihan *database* di tentukan oleh beberapa faktor, diantaranya:

1. Faktor teknis,
2. Faktor ekonomi,
3. Faktor politik organisasi.

Faktor Teknis:

a. Jenis-jenis DBMS (relational, network, hierarchical, dll),
b. Struktur penyimpanan, dan jalur akses yang mendukung DBMS, pemakai, dll.
c. Tipe antarmuka dan programmer
d. Tipe bahasa queri

Faktor Ekonomi:

a. biaya penyediaan perangkat lunak
b. biaya pemeliharaan
c. biaya penyediaan perangkat keras
d. biaya konversi dan pembuatan database
e. biaya untuk personalia
f. biaya pelatihan
g. biaya pengoperasian
h. biaya untuk personalia
i. biaya pelatihan

Pengoperasian Arsitektur Basis Data dibangun menggunakan format paket bahasa yaitu DDL, dan DML.

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

Merupakan satu paket bahasa DBMS yang berguna untuk melakukan spesifikasi terhadap skema basis data. Contoh perintah DDL misalnya, Create Table, Create Index, Alter table, drop view, Drop index.

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

   Merupakan satu paket DBMS yang memperbolehkan pemakai untuk mengakses atau memanipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat. Dengan DML kita akan dapat:

   a. Mengambil informasi yang tersimpan dalam basis data.

   b. Menyisipkan informasi baru dalam basis data.

   c. Menghapus informasi dari tabel.

Terdapat dua tipe DML yaitu prosedural dan non prosedural. Prosedural DML membutuhkan pemakai untuk mensfesifikasikan data apa yang dibutuhkan dan bagaimana cara mendapatkannya. Contoh bahasa prosedural adalah dBase III, FoxBase, FoxPro. Sedang non prosedural DML membutuhkan pemakai untuk menspesifikasikan data apa yang dibutuhkan tanpa harus tahu bagaimana cara mendapatkannya. Contoh bahasa non prosedural adalah SQL (*Structured Query Language*) atau QBE (*Query By Example*). Contoh perintah DML misalnya Insert, Select, Update, dan Delete

Dalam bahasa Query dikenal juga dengan **DCL** (*Data Control Language*), yang merupakan bahasa pengendali data, yang digunakan untuk melakukan otorisasi terhadap pengaksesan data dan mengalokasikan ruang. Contoh perintahnya misalnya Grant, Revoke, Commit, dan Rollback.

Ketiga perintah bahasa tersebut (DDL, DML, DCL) saat ini telah dibentuk menjadi paket bahasa yang disebut sebagai **SQL** (*Structured Query Language*), yang pada prakteknya implementasi SQL sangat bervariasi. Tidak semua fitur SQL didukung oleh vendor software. Beberapa contoh software basis data yang menggunakan SQL seperti DB2, Ingres, Informix, Oracle, MS-Access, MySQL, PostgreSQL, Rdb, dan Sybase.

Tipe File DBMS

Ada beberapa tipe file dari *DataBase Management System:*

Tipe-tipe file yang digunakan dalam DBMS:

1. File Induk *(master File)*
   i. file induk acuan *(reference master file)*: file induk yang recordnya relatif statis, jarang berubah nilainya. Misalnya file daftar gaji, file matakuliah.
   ii. file induk dinamik *(dynamic master file)*: file induk yang nilai dari record-recordnya sering berubah atau sering dimutakhirkan *(update)* sebagai hasil dari suatu transaksi. Misalnya file induk data barang, yang setiap saat harus di *up-date* bila terjadi transaksi.
2. File Transaksi *(transaction file)*

   File ini bisa disebut *file input*; digunakan untuk merekam data hasil dari transaksi yang terjadi. Misalnya file penjualan yang berisi data hasil transaksi penjualan.
3. File Laporan *(Report file)*

   File ini bisa disebut *output file*, yaitu file yang berisi informasi yang akan ditampilkan.
4. File Sejarah *(history file)*

   File ini bisa disebut file arsip (archival file), merupakan file yang berisi data masa lalu yang sudah tidak aktif lagi, tetapi masih disimpan sebagai arsip.
5. File Pelindung *(backup file)*

   File ini merupakan salinan dari file-file yang masih aktif di dalam database pada suatu saat tertentu. File ini digunakan sebagai pelindung atau cadangan bila file database yang aktif mengalami kerusakan atau hilang.

∞

BAB 2
BASIS DATA RELASIONAL

2.1 Pengantar Basis Data Relasional

Basis Data Relasional merupakan suatu cara untuk mengelola data secara fisik kedalam memori.Basis Data relasional menggunakan tabel dua dimensi yang terdiri atas baris dan kolom untuk memberi gambaran sebuah berkas data.Untuk menerapkan sebuah basis data (yg terdiri atas sejumlah tabel yang saling berhubungan), dibutuhkan perangkat lunak (*software*) khusus. Perangkat lunak ini disebut Sistem Pengelola Basis Data.

KEUNGGULAN BASIS DATA RELASIONAL

1. Bentuknya sederhana
2. Mudah untuk melakukan berbagai operasi data

ISTILAH DALAM BASIS DATA RELASIONAL

1. ***Relasi***

   Sebuah tabel yang terdiri dari beberapa kolom dan beberapa baris.

2. ***Atribut:***

   Kolom pada sebuah relasi

3. ***Tupel***

   Baris pada sebuah relasi

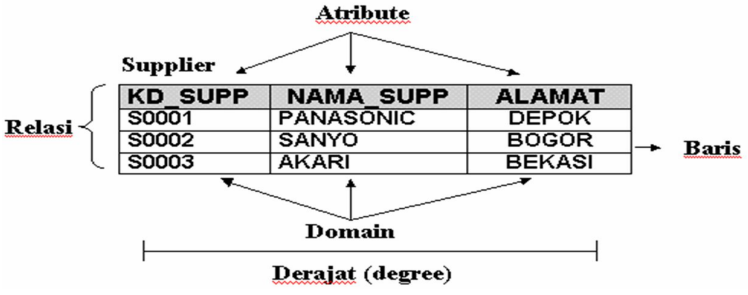
4. ***Domain***

   Kumpulan nilai yang valid untuk satu atau lebih atribut

5. ***Derajat (degree)***

   Jumlah atribut dalam sebuah relasi

6. ***Cardinality***



Gambar 2.1 Jumlah tupel dalam sebuah relasi

Relasional Keys

Ada beberapa macam relasional keys ini diantaranya:

1. **Super Key**

   adalah sebuah atau sekumpulan atribut yang secara unik mengidentifikasi sebuah baris dalam tabel relasi.

2. **Candidate Key**

   adalah relasi mungkin mempunyai lebih dari satu key. Masing-masing disebut CANDIDATE KEY.

3. **Primary Key**

   adalah candidate key yang dipilih sebagai pengidentifikasi unik untuk sebuah tabel relasi.

4. **Alternate Key**

   adalah suatu schema relasi dapat memiliki atribut yang menunjuk ke PRIMARY KEY dari relasi lain. Atribut ini disebut FOREIGN KEY.

Relational Integrity Rules

1. **Null**

   Nilai suatu atribut yang tidak diketahui dan tidak cocok untuk baris tersebut

2. **Entity Integrity**

   Tidak ada satu komponen primary key yang bernilai null

3. **Referential Integrity**

Suatu domain dapat dipakai sebagai kunci primer bila merupakan atribut tunggal pada domain yang bersangkutan

Bahasa Pada Basis Data Relational

Bahasa yang digunakan adalah bahasa query sebagai pernyataan yang diajukan untuk mengambil informasi Terbagi 2:

1. Bahasa Formal

   Bahasa query yang diterjemahkan dengan menggunakan simbol-simbol matematis

   Contoh:

   a. Aljabar relasional

      Bahasa query prosedural: pemakai menspesifikasikan data apa yang dibutuhkan dan bagaimana untuk mendapatkannya

   b. Kalkulus relasional

      Bahasa query non prosedural: pemakai menspesifikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana utntuk mendapatkannya
      Kalkulus relasional terbagi 2:

      i. Kalkulus relasional tupel

      ii. Kalkulus Relasional Domain

2. Bahasa Komersial

   Bahasa query yang dirancang sendiri oleh programmer menjadi suatu program aplikasi agar pemakai lebih mudah menggunakannya (*user friendly)*

   Contoh:

   a. QUEL

      Berbasis pada bahasa kalkulus relasional

   b. QBE

      Berbasis pada bahasa kalkulus relasional

   c. SQL

      Berbasis pada bahasa kalkulus relasional dan aljabar relasional

2.2 Normalisasi

Normalisasi adalah suatu proses memperbaiki / membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika.

Proses normalisasi adalah proses pengelompokan data elemen menjadi tabel-tabel yang menunjukkan entity dan relasinya. Pada proses normalisasi dilakukan pengujian pada beberapa kondisi apakah ada kesulitan pada saat menambah/menyisipkan, menghapus, mengubah dan mengakses pada suatu basis data. Bila terdapat kesulitan pada pengujian tersebut maka perlu dipecahkan relasi pada beberapa tabel lagi atau dengan kata lain perancangan basis data belum optimal.

Tujuan Normalisasi

Ada beberapa tujuan dari normalisasi ini:

1. Untuk menghilangkan kerangkapan data
2. Untuk mengurangi kompleksitas
3. Untuk mempermudah pemodifikasian data

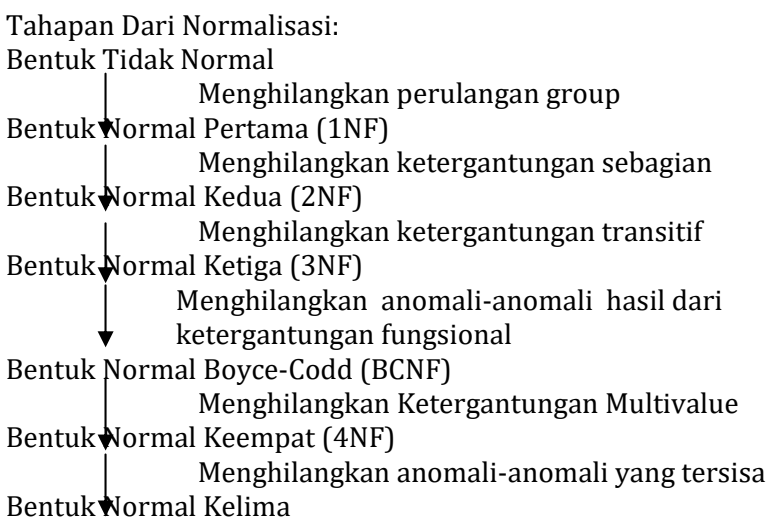
Proses Normalisasi

Dimana Proses Normalisasi ini ada dua langkah:

1. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat.
2. Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

Tahap Normalisasi

Tahapan Dari Normalisasi:



1. **Bentuk Tidak Normal**

   Kumpulan data yang direkam, tidak ada keharusan mengikuti suatu format tertentu sehingga bisa saja datanya tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan kedatanngannya.

2. **Bentuk Normal Pertama (1NF)**

   Bentuk normal pertama terpenuhi jika sebuah tabel tidak memiliki attribut yang bernilai banyak (multi value attribute) artinya setipa pertemuan baris dan kolom hanya berisikan satu nilai (single value attribute)

3. **Bentuk Normal Kedua (2NF)**

   Bentuk norma kedua akan terpenuhi jika bentuk data telah memenuhi kriteria bentuk normal pertama dan setiap attribut yang bukan kunci haruslah bergantung secara fungsional (functional dependency) terhadap attribut kunci / primary key. Sehingga untuk membentuk normal ke dua haruslah sudah ditentukan field kunci.

4. **Bentuk Normal Ketiga (3NF)**

   Bentuk normal ketiga telah memenuhi kriteria bentuk normal pertama dan kedua serta tidak terdapat transitive dependency yaitu sebuah attribute yang bukan kunci selain bergantung

kepada attribut kunci, juga bergantung kepada attribut bukan kunci lainnya. Sehingga setiap attribut bukan kunci haruslah bergantung hanya pada attribut kunci (primary key)

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

   BCNF memiliki ketentuan yaitu masing-masing atribut utama bergantung fungsional penuh pada masing-masing kunci dimana kunci tersebut bukan bagiannya. Relasi adalah BCNF (optimal) jika setiap determinan atribut-atribut relasi adalah kunci relasi. Relasi adalah BCNF (optimal) jika kapanpun fakta-fakta disimpan mengenai beberapa atribut, maka atribut-atribut ini merupakan satu kunci relasi. BCNF dapat memiliki lebih dari satu kunci. Properti penting BCNF adalah relasi tidak memiliki informasi yang redundan.

6. **Bentuk Normal Keempat (4NF)**

   Relasi dalam bentuk normal keempat (4NF) jika relasi dalam BCNF dan tidak berisi kebergantungan banyak nilai. Untuk menghilangkan kebergantungan banyak nilai dari satu relasi, kita membagi relasi menjadi dua relasi baru. Masing-masing relasi berisi dua atribut yang mempunyai hubungan banyak nilai.

7. **Bentuk Normal Kelima**

   Bentuk normal kelima (5NF) berurusan dengan properti yang disebut join tanpa adanya kehilangan informasi (lossless join). Bentuk normal kelima (5NF) juga disebut PJNF (projection-join normal form). Kasus-kasus ini sangat jarang muncul dan sulit untuk dideteksi secara praktis.

Ketergantungan Fungsional

Ketergantungan Fungsional menggambarkan hubungan *attributes* dalam sebuah relasi. Suatu attribute dikatakan *functionally dependency* pada yang lain jika kita menggunakan harga atribut tersebut untuk menentukan harga atribut yang lain.

Simbol yang digunakan adalah → untuk mewakili functional dependency.
→ dibaca secara fungsional menentukan.

1. X → Y

X dan Y adalah atribut dari sebuah tabel. Berarti secara fungsional X menentukan Y atau Y tergantung pada X, jika dan hanya jika ada 2 baris data dengan nilai X yang sama, maka nilai Y juga sama.

Contoh:

KodeBarang	NamaBarang
B001	Televisi
B002	Tape

Tabel 2.1 Ketergantungan Fungsional

2. KodeBarang → Namabarang

   Karena untuk setiap nilai KodeBarang yang sama, maka nilai NamaBarang juga sama

Ketergantungan Fungsional Penuh

Atribut Y tergantung fungsional penuh pada X, jika Y tidak tergantung pada subset X (bila X adalah key gabungan).

Contoh:

KodeBarang	NamaBarang	KodeTransaksi	Jumlah
B001	Televisi	TRX001	15
B002	Tape	TRX002	20

Tabel 2.2 Ketergantungan Fungsional Penuh

KodeBarang → Namabarang
KodeBarang, KodeTransaksi → Jumlah

Karena untuk setiap nilai KodeBarang yang sama, maka nilai NamaBarang juga sama. Karena untuk setiap nilai KodeBarang yang terdapat pada KodeTransaksi yang sama, maka nilai Jumlah juga sama.

Ketergantungan Fungsional Transitif

Atribut Z tergantung fungsional tansitif pada X, jika Y tidak tergantung pada X dan Z tergantung pada Y (X → Y, Y → Z, maka X→Z).

Contoh:

KodeBarang	NamaBarang	KodeJenis	JenisBarang
B001	Televisi	1	Baru
B002	Tape	2	Bekas

Tabel 2.3 Ketergantungan Fungsional Transitif

KodeBarang → KodeJenis
KodeJenis → JenisBarang
Maka
KodeBarang → JenisBarang

Beberapa Contoh dari normalisasi:

PT SANTA PURI
Jl. Senopati II
Padang

Faktur Pembelian Barang

Kode Supplier : S01
Nama Supplier : Gobel Nusantara
Tanggal : 07/02/2012
No. Faktur 998

Kode	Nama Barang	Qty	Harga	Jumlah
A01	AC split 1/2 PK	10	1.350.000	13.500.000
A02	AC split 1PK	10	2.000.000	20.000.000
TOTAL				33.500.000

Tangal Jatuh Tempo : 09/02/2012

Tabel 2.4 Contoh Normalisasi

No. Faktur	Kode Supplier	Nama Supplier	Kode Barang	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S05	Hitachi	R02	Rice Cooker	02/02/2012	05/02/2012	10	150.000	1.500.000	1.500.000
998	S01	Gobel Nusantara	A01	AC split 1/2 PK	07/02/2012	09/02/2012	10	1.350.000	13.500.000	33.500.000
			A02	AC split 1PK	07/02/2012	09/02/2012	10	2.000.000	20.000.000	

Table 2.5 Bentuk Unnomalized

No. Faktur	Kode Supplier	Nama Supplier	Kode Barang	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S05	Hitachi	R02	Rice Cooker	02/02/2012	05/02/2012	10	150.000	1.500.000	1.500.000
998	S01	Gobel Nusantara	A01	AC split 1/2 PK	07/02/2012	09/02/2012	10	1.350.000	13.500.000	13.500.000
999	S02	Gobel Nusantara	A02	AC split 1PK	07/02/2012	09/02/2012	10	2.000.000	20.000.000	20.000.000

Table 2.6 Bentuk Normal Kesatu

Tabel Barang

Kode Barang	Nama Barang	Harga
R02	Rice Cooker	150.000
A01	AC split 1/2 PK	1.350.000
A02	AC split 1PK	2.000.000

Tabel Supplier

Kode Supplier	Nama Supplier
S05	Hitachi
S01	Gobel Nusantara
S02	Gobel Nusantara

Tabel Faktur

No. Faktur	Tanggal	Jatuh Tempo	Qty	Jumlah	Total
779	02/02/2012	05/02/2012	10	150.000	1.500.000
998	07/02/2012	09/02/2012	10	13.500.000	33.500.000
999	07/02/2012	09/02/2012	10	20.000.000	33.500.000

Table 2.7 Bentuk Normal Kedua

Tabel Barang

Kode Barang	Nama Barang	Harga
R02	Rice Cooker	150.000
A01	AC split 1/2 PK	1.350.000
A02	AC split 1PK	2.000.000

Tabel Supplier

Kode Supplier	Nama Supplier
S05	Hitachi
S01	Gobel Nusantara
S02	Gobel Nusantara

Tabel Faktur

No. Faktur	Kode Barang	Qty
779	R02	10
998	A01	10
999	A02	10

Tabel Rekap Faktur

No. Faktur	Kode Supplier	Tanggal	Jatuh Tempo	Total
779	S05	02/02/2012	05/02/2012	1.500.000
998	S01	07/02/2012	09/02/2012	33.500.000
999	S02	07/02/2012	09/02/2012	33.500.000

Table 2.8 Bentuk Normal Ketiga

Tabel Barang

Kode Barang	Nama Barang	Harga
R02	Rice Cooker	150.000
A01	AC split 1/2 PK	1.350.000
A02	AC split 1PK	2.000.000

Tabel Supplier

Kode Supplier	Nama Supplier
S05	Hitachi
S01	Gobel Nusantara
S02	Gobel Nusantara

Tabel Faktur

No. Faktur	Kode Barang	Qty
779	R02	10
998	A01	10
999	A02	10

Tabel Rekap Faktur

No. Faktur	Kode Supplier	Tanggal	Jatuh Tempo
779	S05	02/02/2012	05/02/2012
998	S01	07/02/2012	09/02/2012
999	S02	07/02/2012	09/02/2012

Table 2.9 Bentuk BCNF

Mahasiswa	Nilai	Bid.Penilaian
Tomi	90	Tugas
Tomi	98	UTS
Tomi	94	UAS
Tomi	*null*	Absen

Mahasiswa	Nilai
Tomi	90
Tomi	98
Tomi	94

Mahasiswa	Bid.Penilaian
Tomi	Tugas
Tomi	UTS
Tomi	UAS
Tomi	Absen

Konsultan	Nilai_Proyek	Bidang
Koko	100.000.000	Pertanian
Reza	200.000.000	Pembangunan
Reza	*null*	Pendidikan

Konsultan	Nilai_Proyek
Koko	100.000.000
Reza	200.000.000
Reza	*null*

Konsultan	Bidang
Koko	Pertanian
Reza	Pembangunan
Reza	Pendidikan

Table 2.10 Contoh Normalisasi Bentuk Keempat

∞

BAB 3
SIKLUS HIDUP PERANCANGAN BASIS DATA

3.1 Aktifitas Utama

Siklus Hidup Perancangan Basis Data

gambar 3.1 Siklus Hidup Perancangan Basis Data

Ringkasan Aktifitas Utama

Stage	Main Activities
Database planning	Perencanaan bagaimana tahapan-tahapan dalam siklus hidup dapan direalisasikan dengan efektif dan efisien
System definition	Mespesifikasikan lingkup dan batasan aplikasi database, para penggunanya dan area aplikasi
Requirements collection and analysis	Mengumpulkan dan menganalisa seluruh kebutuhan pengguna dan area aplikasi
Database design	Perancangan conseptual, logical, dan physical dari database
DBMS selection (optional)	Memilih DBMS yang lebih cocok untuk aplikasi database yang ada
Application design	Perancangan interface pengguna, program aplikasi yang digunakan dan proses databasenya

Stage	Main Activities
Prototyping (optional)	Membuat model kerja dari aplikasi database, yang memungkinkan perancang atau pengguna melihat dan mengevaluasi bagaimana sistem akhir akan berfungsi dan bekerja
Implementation	Pembuatan definisi database eksternal, konseptual dan internal dan program aplikasi
Data conversion and loading	Mengambil dan memindahkan data dari sistem lama ke sistem baru
Testing	Aplikasi database diuji untuk kesalahan dan validasi, dibandingkan dengan spesifikasi kebutuhan yang diberikan oleh pengguna
Operational maintenance	Setelah aplikasi database terimplementasi seluruhnya. Secara berkesinambungan diawasi dan diatur. Kebutuhan baru dapat ditambahkan kedalam aplikasi database melalui tahapan-tahapan yang ada, jika diperlukan

Table 3.1 Ringkasan Aktiva Utama

Perencanaan Database (Database planning)

Merupakan aktivitas manajemen yang memungkinkan tahapan dari *database application lifecycle* direalisasikan se-efektif dan se-efisien mungkin.

Perencanaan database harus terintegrasi dengan keseluruhan strategi sistem informasi dari organisasi.

Terdapat 3 hal pokok yang berkaitan dengan strategi sistem informasi, yaitu :

1. Identifikasi rencana dan sasaran (*goals*) dari enterprise termasuk mengenai sistem informasi yang dibutuhkan.
2. Evaluasi sistem informasi yang ada untuk menetapkan kelebihan dan kekurangan yang dimiliki.
3. Penaksiran kesempatan IT yang mungkin memberikan keuntungan kompetitif.

Metodologi untuk mengatasi hal tersebut diatas yaitu :

Database Planning – **Mission Statement:**

Mission statement untuk database project mendefinisikan tujuan utama dari aplikasi database.

Mengarahkan database project, biasanya mendefinisikan perintah tugas (mission statement)

Mission statement membantu menjelaskan kegunaan dari database project dan menyediakan alur yang lebih jelas untuk mencapai efektifitas dan efisiensi penciptaan dari suatu aplikasi database yang diinginkan.

Database Planning – **Mission Objectives:**

Ketika mission statement telah didefinisikan, maka mission objectives didefinisikan.

Setiap objective (tujuan) harus mengidentifikasikan tugas khusus yang harus didukung oleh database.

Dapat juga disertai dengan beberapa informasi tambahan yang menspesifikasikan pekerjaan yang harus diselesaikan, sumberdaya yang digunakan dan biaya untuk membayar kesemuanya itu.

Database planning juga harus menyertakan pengembangan standar-standar yang menentukan :

1. Bagaimana data akan dikumpulkan,
2. Bagaimana menspesifikasikan format/bentuk data,
3. Dokumentasi penting apakah yang akan diperlukan,
4. Bagaimana desain dan implementasi harus dilakukan.

Definisi Sistem (System definition)

Menjelaskan batasan-batasan dan cakupan dari aplikasi database dan sudut pandang user (*user* view) yang utama.

User view mendefinisikan apa yang diwajibkan dari suatu aplikasi database dari perspektif, diantaranya :

Aturan kerja khusus (seperti Manajer atau Supervisor)

Area aplikasi enterprise (seperti Marketing, Personnel, atau Stock Control).

Aplikasi database dapat memiliki satu atau lebih user view

Identifikasi user view, membantu memastikan bahwa tidak ada user utama dari suatu database yang terlupakan ketika pembuatan aplikasi baru yang dibutuhkan.

User views juga membantu dalam pengembangan aplikasi database yang rumit/kompleks memungkinkan permintaan-permintaan dipecah kedalam bagian-bagian yang lebih mudah diatur.

Analisis dan Pengumpulan Kebutuhan (Requirements collection and analysis)

Suatu proses pengumpulan dan analisa informasi mengenai bagian organisasi yang didukung oleh aplikasi database, dan menggunakan informasi tersebut untuk identifikasi kebutuhan user pada sistem yang baru.

Informasi dikumpulkan untuk setiap user view utama meliputi:

1. Deskripsi data yang digunakan atau dihasilkan
2. Detail mengenai bagaimana data digunakan/dihasilkan
3. Beberapa kebutuhan tambahan untuk aplikasi database yang baru

Informasi dianalisa untuk identifikasi kebutuhan agar disertakan dalam aplikasi database yang baru.

Aktifitas penting lainnya, adalah menentukan bagaimana mengatur aplikasi database dengan multiple user view, ada 3 pendekatan yaitu:

1. Centralized approach;
2. View integration approach;
3. Combination of both approaches.

Pendekatan Terpusat (*Centralized approach*)

1. Kebutuhan untuk setiap user view digabungkan menjadi sekumpulan kebutuhan.
2. Sebuah *global data model* dibuat berdasarkan atas penggabungan kebutuhan (dimana merepresentasikan seluruh user view).

Pendekatan Integrasi View (View integration approach)

1. Kebutuhan untuk setiap user view digunakan untuk membangun model data terpisah untuk merepresentasikan user view tersebut. Hasil dari model data tersebut nantinya digabungkan dalam tahapan desain database.
2. Model data yang merepresentasikan user view tunggal disebut *local data model*, dan tersusun atas diagram-diagram dan dokumentasi yang secara formal menggambarkan kebutuhan dari user view khusus terhadap database.
3. Kemudian *Local data model* digabungkan untuk menghasilkan *global data model,* yang merepresentasikan seluruh user view untuk database.

Database Design

Merupakan suatu proses pembuatan sebuah desain database yang akan mendukung tujuan dan operasi suatu enterprise.

Tujuan utamanya adalah :

1. Merepresentasikan data dan relationship antar data yang dibutuhkan oleh seluruh area aplikasi utama dan user group.
2. Menyediakan model data yang mendukung segala transaksi yang diperlukan pada data.
3. Menspesifikasikan desain minimal yang yang secara tepat disusun untuk memenuhi kebutuhan performa yang ditetapkan pada sistem (misal : waktu respon).

Pendekatan dalam desain database

1. **Top-down**

   Diawali dengan pembentukan model data yang berisi beberapa entitas high-level dan relationship, yang kemudian menggunakan pendekatan *top-down* secara berturut-turut

untuk mengindentifikasikan entitas lower level, relationship dan atribut lainnya.

2. **Bottom-up**

   Dimulai dari atribut dasar (yaitu, sifat-sifat entitas dan relationship), dengan analisis dari penggabungan antar atribut, yang dikelompokan kedalam suatu relasi yang merepresentasikan tipe dari entitas dan relationship antar entitas.

3. **Inside-out**

   Berhubungan dengan pendekatan *bottom-up* tetapi sedikit berbeda dengan identifikasi awal entitas utama dan kemudian menyebar ke entitas, relationship, dan atribut terkait lainnya yang lebih dulu diidentifikasi.

4. **Mixed**

   Menggunakan pendekatan *bottom-up* dan *top-down* untuk bagian yang berbeda sebelum pada akhirnya digabungkan.

Kegunaan utama dari data modelling meliputi :

1. Untuk membantu dalam memahami arti (semantik) dari data.
2. Untuk memfasilitasi komunikasi mengenai informasi yang dibutuhkan.

Pembuatan model data mengharuskan menjawab pertanyaan mengenai entitas, relationship dan atribut.

Model data memastikan kita memahami:

1. Setiap perspektif user mengenai data
2. Sifat dari data itu sendiri, independen terhadap representasi fisiknya.
3. Kegunaan data melalui user view.

Tiga fase database design :

1. **Conceptual database design**

   Suatu proses pembentukan model dari informasi yang digunakan dalam enterprise, independen dari keseluruhan aspek fisik. Model data dibangun dengan menggunakan informasi dalam spesifikasi kebutuhan user. Model data

konseptual merupakan sumber informasi untuk fase desain logical.

2. **Logical database design**

   Suatu proses pembentukan model dari informasi yang digunakan dalam enterprise berdasarkan model data tertentu (misal : relasional), tetapi independen terhadap DBMS tertentu dan aspek fisik lainnya. Model data konseptual yang telah dibuat sebelumnya, diperbaiki dan dipetakan kedalam model data logical.

3. **Physical database design.**

   Suatu proses yang menghasilkan deskripsi implementasi database pada penyimpanan sekunder. Menggambarkan struktur penyimpanan dan metode akses yang digunakan untuk mencapai akses yang efisien terhadap data. Dapat dikatakan juga, desain fisikal merupakan cara pembuatan menuju sistem DBMS tertentu

DBMS selection (optional)

Pemilihan DBMS yang tepat untuk mendukung aplikasi database. Dapat dilakukan kapanpun sebelum menuju desain logical asalkan terdapat cukup informasi mengenai kebutuhan sistem. Tahap-tahap utama untuk memilih DBMS :

1. Mendefinisikan terminologi studi referensi.
2. Mendaftar dua atau tiga produk.
3. Evaluasi produk.
4. Rekomendasi pilihan dan laporan produk.

Desain Aplikasi (Application design)

Desain interface user dan program aplikasi yang menggunakan dan memproses database. Desain database dan aplikasi merupakan aktifitas paralel yang meliputi dua aktifitas penting, yaitu :

1. Transaction design

   Transaksi adalah satu aksi atau serangkaian aksi yang dilakukan oleh user tunggal atau program aplikasi, yang mengakses atau merubah isi dari database. Kegunaan dari desain transaksi adalah untuk menetapkan dan keterangan

karakteristik high-level dari suatu transaksi yang dibutuhkan pada database, diantaranya :

a. Data yang akan digunakan oleh transaksi.
b. Karakteristik fungsional dari suatu transaksi.
c. Output transaksi
d. Keuntungannya bagi user.
e. Tingkat kegunaan yang diharapkan.

2. Terdapat tiga tipe transaksi, yaitu :
   a. **retrieval transaction**, digunakan untuk pemanggilan (retrieve) data untuk ditampilkan di layar atau menghasilkan suatu laporan.
   b. **update transaction**, digunakan untuk menambahkan record baru, menghapus record lama, atau memodifikasi record yang sudah ada didalam database.
   c. **mixed transaction**, meliputi pemanggilan dan perubahan data.

3. User interface design**.**

   Beberapa aturan pokok dalam pembuatan user interface:
   a. *Meaningful title*, diusahakan pemberian nama suatu form cukup jelas menerangkan kegunaan dari suatu form/report.
   b. *Comprehensible instrctions,* penggunaan terminologi yang familiar untuk menyampaikan instruksi ke user dan jika informasi tambahan dibutuhkan, maka harus disediakan *helpscreen*
   c. *Logical grouping and sequencing of fields,* field yang saling berhubungan ditempatkan pada form/report yang sama. Urutan field harus logis dan konsisten.
   d. *Visually appealing layout of the form/report,* tampilan form/report harus menarik, dan sesuai dengan *hardcopy* agar konsisten.
   e. *Familiar field labels,* penggunaan label yang familiar.
   f. *Consistent terminology and abbreviation,* terminology dan singkatan yang digunakan harus konsisten.

g. *Consistent use of color.*

h. *Visible space and boundaries for data-entry fields,* jumlah tempat yang disediakan untuk data entry harus diketahui oleh user.

i. *Convinient cursor movement,* user dapat dengan mudah menjalankan operasi yang diinginkan dengan menggerakkan cursor pada form/report.

j. *Error correction for individual characters and entire fields,* user dapat dengan mudah menjalankan operasi yang diinginkan dan melakukan perubahan terhadap nilai field.

k. *Error messages for unacceptable values.*

l. *Optional fields marked clearly.*

m. *Explanatory messages for fields,* ketika user meletakkan cursor pada suatu field, maka keterangan mengenai field tersebut harus dapat dilihat.

Prototyping (optional)

Membuat model kerja suatu aplikasi database. Tujuan utama dari pembuatan prototyping adalah :

1. Untuk mengidentifikasi *feature* dari sistem yang berjalan dengan baik atau tidak.
2. Untuk memberikan perbakan-perbaikan atau penambahan feature baru.
3. Untuk klarifikasi kebutuhan user.
4. Untuk evaluasi feasibilitas (kemungkinan yang akan terjadi) dari desain sistem khusus.

Terdapat dua macam strategi prototyping yang digunakan saat ini, yaitu:

1. **Requirements prototyping**, menggunakan prototype untuk menentukan kebutuhan dari aplikasi database yang diinginkan dan ketika kebutuhan itu terpenuhi maka prototype akan dibuang.
2. **Evolutionary prototyping**, digunakan untuk tujuan yang sama. Perbedaannya, prototype tidak dibuang tetapi dengan

pengembangan lanjutan menjadi aplikasi database yang digunakan.

Implementation

Merupakan realisasi fisik dari database dan desain aplikasi. Implementasi database dicapai dengan menggunakan :

1. DDL untuk membuat skema database dan file database kosong.
2. DDL untuk membuat user view yang diinginkan.
3. 3GL dan 4GL untuk membuat program aplikasi. Termasuk transaksi database disertakan dengan menggunakan DML, atau ditambahkan pada bahasa pemrograman.

Data conversion and loading

Pemindahan data yang ada kedalam database baru dan mengkonversikan aplikasi yang ada agar dapat digunakan pada database yang baru. Tahapan ini dibutuhkan ketika sistem database baru menggantikan sistem yang lama. DBMS biasanya memiliki utilitas yang memanggil ulang file yang sudah ada kedalam database baru.Dapat juga mengkoversi dan menggunakan program aplikasi dari sistem yang lama untuk digunakan oleh sistem yang baru.

Testing

Suatu proses eksekusi program aplikasi dengan tujuan untuk menemukan kesalahan. Dengan menggunakan strategi tes yang direncanakan dan data yang sesungguhnya. Pengujian hanya akan terlihat jika terjadi kesalahan software. Mendemostrasikan database dan program aplikasi terlihat berjalan seperti yang diharapkan.

Operational maintenance

Suatu proses pengawasan dan pemeliharaan sistem setelah instalasi, meliputi :

1. Pengawasan performa sistem, jika performa menurun maka memerlukan perbaikan atau pengaturan ulang database.
2. Pemeliharaan dan pembaharuan aplikasi database (jika dibutuhkan).
3. Penggabungan kebutuhan baru kedalam aplikasi database.

Langkah ke-3 disebut juga perancangan basis data. Langkah 4 dan 5 merupakan bagian dari fase design dan implementation pada siklus kehidupan sistem informasi yang besar. Pada umumnya basis data pada organisasi menjalani seluruh aktifitas-aktifitas siklus kehidupan di atas. Langkah 5 dan 6 tidak berlaku jika basis data dan aplikasi-aplikasinya baru.

Sebelum adanya sistem basis data, sistem yang digunakan untuk mengelola data adalah sistem file atau dikenal juga dengan ***file-based system***.Menurut ***Connoly (2002,p7)***, file-based system adalah kumpulan dari program aplikasi yang berfungsi untuk menghasilkan laporan untuk pengguna. Tiap program mempunyai dan mengelola datanya masing-masing.

File-based system sebagai sistem penyimpanan dan pengurutan data dengan cara mengumpulkan data-data yang sejenis, memberi judul atau label dan melakukan index berdasarkan alfabet, untuk memudahkan proses pencarian data kembali.

Sistem ini menggunakan metode desentralisasi yang berarti masing-masing departemen menyimpan dan mengontrol datanya masing-masing.

File-based system menggunakan program aplikasi yang dapat memproses data sehingga dapat menghasilkan laporan yang dapat digunakan oleh masing-masing departemen yang mengelolanya.

Sistem ini dapat bekerja dengan baik apabila jumlah data yang disimpan tidak terlalu banyak, bahkan dapat bekerja dengan baik pada data dengan jumlah banyak tetapi hanya bila proses yang dilakukan adalah simpan dan ambil data. Sistem mulai tidak bekerja dengan baik saat diperlukan proses cek silang antar data, atau saat data berhubungan dengan data lain.

3.2 Prinsip-Prinsip Perancangan Basis Data

Prinsip dari database secara umum adalah integritas (menyatu), maka antara suatu file dengan file yang lain (table dengan table lain) harus saling berhubungan/table relationship. Perencanaan database selanjutnya adalah model data logika. Model data logika (logical data modeling) adalah satu teknik untuk menjelaskan dengan baik struktur informasi bisnis dan aturan-aturan sebagai masukan pada proses perancangan database.

Beberapa hal yag perlu diperhatikan dalam merancang database antara lain :

1. Miliki perspektif globalnya

Harus dipandang dari sisi keseluruhan sistem
Misal : database akademik
Bagian Jurusan (KHS, KRS, DOSEN, MK, MHS) Bagian Keuangan (pembayaran gaji karyawan, SPP MHS)
Dosen (KRS,MHS bimbingan, MK yang diajar)dll
Bila ada permasalahan pada sistem, harus dipandang dari keseluruhan sistem.

2. Gunakan perancangan atas bawah

   Perancangan harus dari atasan (pimpinan) terlebih dahulu, inovasi dari top, top ikut untuk kepentingan perusahaan, apabila ada permasalahan dari dalam (internal) organisasi maka (pimpinan) dapat menyimpulkan bahwa timbulnya permasalahaan dari sisi yg ia pandang. Kalau dari bawah ke atas maka setiap bagian akan cenderung mengemukakan permasalahannya sendiri-sendiri.

3. Berikan informasi secara selektif

   Untuk setiap level/tingkat, berikan informasi yang sesuai.
   Bahwa untuk tingkat manajer misalnya berikan laporan keuntungan saja. Sedangkan untuk laporan per hari, laporan absensi, kegiatan rutin untuk karyawan tidak perlu diberikan laporan tersebut.

4. Berikan database yang berbeda untuk manajemen yang beda

   Contoh:

   a. kepala program studi:

      Harus mengetahui perkembangan seluruh mhs dari kelas a→b, nama-nama orangnya.
      Mengetahui mata kuliah, memberikan mata kuliah kepada dosen-dosen

   b. Untuk dosen biasa:

      Sebagai dosen pembimbing saja
      Hanya menjalankan kebijakan kepala program studi

5. Jangan mengubah file yang ada

   Tidak boleh asal mengubah file

--oo0oo--

This page is intentionally left blank

BAB 4
PEMODELAN DATA

4.1 Basis Data Relasional

Model relasional terkait dengan 3 aspek:
struktur data, integritas data dan manipulasi data.

Variable vs Value

1. *Relation variable* adalah sebuah relasi yang skemanya terdefinisi.
2. *Relation value* adalah nilai yang dikandung suatu relasi pada suatu waktu disebut juga instans.

Berdasarkan keberadaannya, relasi terbagi 3:

1. *Base Relation*: relasi yang skemanya terdefinisi & benar-benar ada pada basisdata.
2. *Derived Relation*: relasi yang diturunkan dari relasi lainnya dengan menggunakan ekspresi relasional.
3. *View*: *derived relation* yang memiliki nama.

Sebuah basis data yang dilihat oleh pemakai sebagai sekumpulan relasi yaitu variabel relasi, yang ternormalisasi dengan derajat yang beragam.

Bahasa pada basis data relasional

Bahasa yang digunakan adalah bahasa query sebagai pernyataan yang diajukan untuk mengambil informasi Terbagi 2:

1. Bahasa Formal

   Bahasa query yang diterjemahkan dengan menggunakan simbol-simbol matematis

   Contoh:

   a. Aljabar relasional

      Bahasa query prosedural: pemakai menspesifikasikan data apa yang dibutuhkan dan bagaimana untuk mendapatkannya

   b. Kalkulus relasional

Bahasa query non prosedural: pemakai menspesifikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana utntuk mendapatkannya.

2. Bahasa Komersial

   Bahasa query yang dirancang sendiri oleh programmer menjadi suatu program aplikasi agar pemakai lebih mudah menggunakannya (user friendly)

   Contoh:

   Select, Group By, Order By, dll.

Pemodelan Data

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya, penjelasan ini disebut skema. Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut.

Model data adalah sekumpulan konsep yang digunakan untuk menjelaskan struktur dari basis data (database structure) dan memberikan gambaran tingkat-tingkat abstraksi data (data abstraction). Selain itu, Model data juga dapat diartikan sebagai sekumpulan tool konseptual (logical/conceptual level) untuk mendeskripsikan data, relasi-relasi antar data, semantik (makna) data, dan konstrain.

Model data digunakan pada saat mendisain sistem basis data agar diperoleh basis data stabil dengan normalisasi penuh agar data terhindar dari kesalahan-kesalahan seperti tidak konsisten, tidak akurat, dll.

Tujuan model data adalah untuk menyajikan data agar mudah di modifikasi dan di mengerti.

Secara garis besar model data dikelompokkan dalam tiga macam yaitu:

1. Model data berbasis obyek (*object based data model*)
2. Model data berbasis record (*record based data model*)
3. Model data fisik (*physical based data model*)

Model Data Berbasis Obyek

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

Terdiri atas:

1. Entity relationship model
2. Semantic model
3. Binary model

Model Data Berbasis Record

Model ini berdasarkan record/rekaman untuk menjelaskan kepada pemakai mengenai hubungan logik antar data dalam basis data,

Terdiri dari:

1. Hierarchycal model
2. Network model
3. Relational model

Model Data Fisik

Model ini digunakan untuk menguraikan data di tingkat internal atau menjelaskan kepada pemakai bagaimana data-data dalam basis data disimpan dalam media penyimpanan secara fisik. Model ini jarang digunakan karena kerumitan dan kompleksitasnya yang justru menyulitkan pemakai.

Model ini terdiri dari:

1. Unifying model
2. Frame memory

4.2 Entiry Relasional Model

Entity Relationship Model pada hakekatnya perwujudan dari model relasional dalam bentuk diagram, yaitu E-R Diagram.

Berfungsi untuk menggambarkan relasi dari dua file atau dua tabel yang dapat digolongkan dalam tiga macam bentuk relasi, yaitu satu-satu, satu banyak, dan banyak-banyak. Penggambaran ini akan membantu analis sistem dalam melakukan perancangan proses yang nantinya akan dituangkan dalam bentuk baris-baris program.

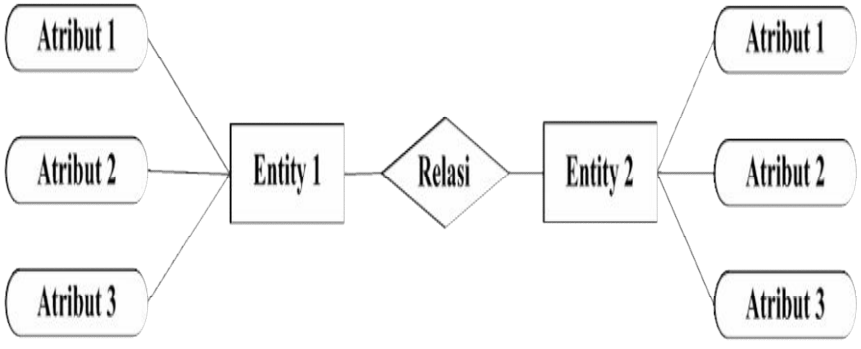
1. Domain data disebut juga sebagai himpunan entitas, diwakili oleh diagram kotak
2. Field data atau atribut diwakili oleh diagram lingkaran atau ellips.
3. Hubungan atau relasi antar domain diwakili oleh jajaran genjang

Simbol-Simbol Standar Entity Relationship:

SYMBOL	NAMA SIMBOL	KETERANGAN
	Entity	Symbol ini menyatakan identitas bisa berupa suatu elemen lingkungan, sumber daya atau transaksi, yang begitu pentingnya bagi perusahaan hingga didokumentasikan dengan data.
	Atribute	Symbol ini digunakan untuk menunjukkan nama-nama atribute yang ada pada entity.
	Primary key atribute	Symbol atribut yang digaris bawahi berfungsi sebagai kunci (key) diantara nama-nama atribute yang ada pada entity.
	Relation ship	Symbol ini menyatakan himpunan relasi. Ini digunakan untuk menunjukkan hubungan yang ada antara entity satu dengan yang lainnya.
	Link	Sebagai penghubung antara himpunan relasi antara entitas dan entitas dengan atribute.

Tabel 4.1 Simbol-Simbol Standar Entity Relationship

Contoh Diagram Hubungan antar Entity



Gambar 4.1 Diagram Hubungan antar *Entity*

Konsep Dasar ER-Modeling

Entity Types

Model relasi entiti didasarkan pada persepsi dunia nyata yang terdiri dari himpunan obyek dasar yang disebut entiti dan relasi antar entiti. Entiti adalah obyek yang dapat diidentifikasi secara unik. Entiti dikarakterisasi dan dipresentasikan dengan suatu gugus atribut. Sekelompok entiti yang memiliki karakterisasi entiti disebut entity set. Setiap entiti dari kelompok tersebut disebut member of set.

Setiap attribut terdapat suatu himpunan nilai yang dapat diberikan pada attribut tersebut yang dikatakan sebagai *domain* dari suatu attribut. Secara formal, suatu attribut adalah suatu fungsi yang memetakan dari suatu himpunan entitas kedalam suatu domain. Setiap entitas dijabarkan dengan suatu himpunan dari attribute dan nilai data, contoh suatu *customer* dijelaskan dengan himpunan {(*name*, Harris), (*social-security*, 890-12-3456), (*street*, North), (*city*, Rye)}.

Konsep Dasar ER-Modeling

Relationship Types

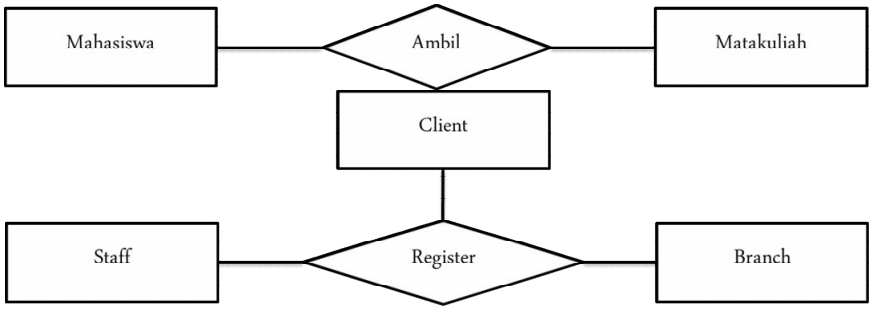
Relation adalah suatu assosiasi diantara beberapa entitas.

Reletionship Types adalah sekumpulan entitas yang berhubungan dan mempunyai arti antara tipe entitas yang ada.

Derajat Relationship adalah jumlah tipe entitas yang yang ada dalam suatu relasi. Derajat Relationship terdiri dari:

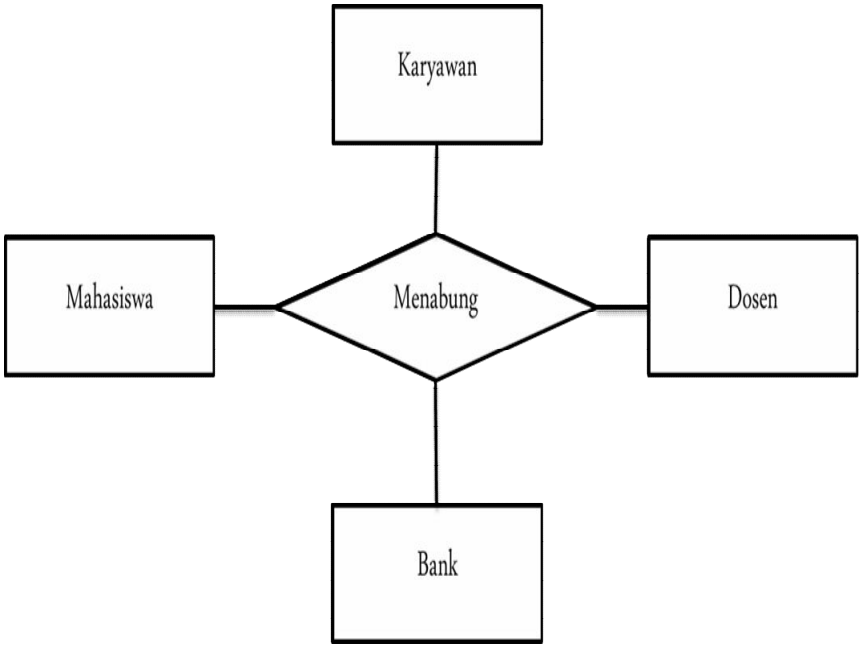
1. *Binary Relationship Set* adalah relasi antara dua entitas.

2. *Ternary Relationship Set* adalah relasi antara tiga entitas



Gambar 4.2 Diagram *Ternary Relationship*

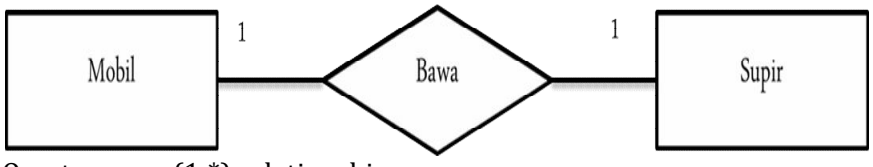
3. *N-ary Relationship Set* adalah relasi antara *n* entitas.



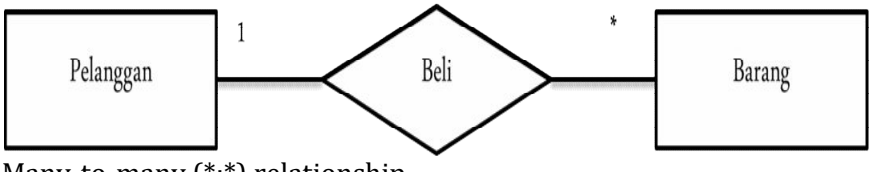
Gambar 4.3 Diagram *N-ary Relationship Set*

Derajat yang paling umum digunakan pada suatu relationship adalah:

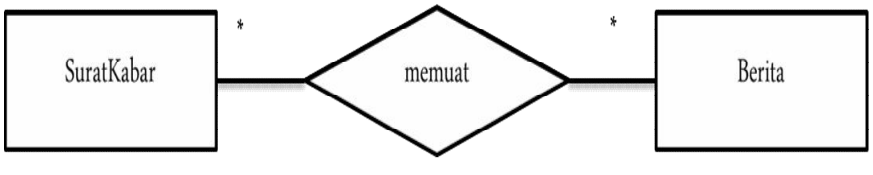
One-to-one (1:1) relationship



One-to-many (1:*) relationship

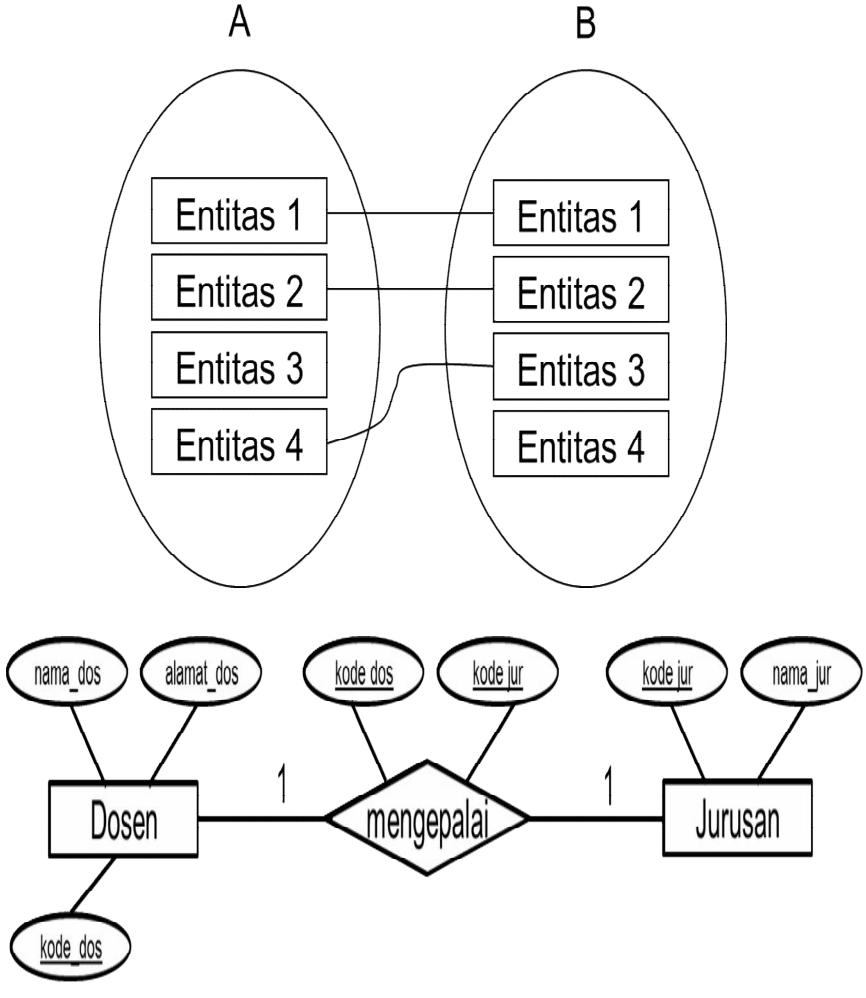


Many-to-many (*:*) relationship



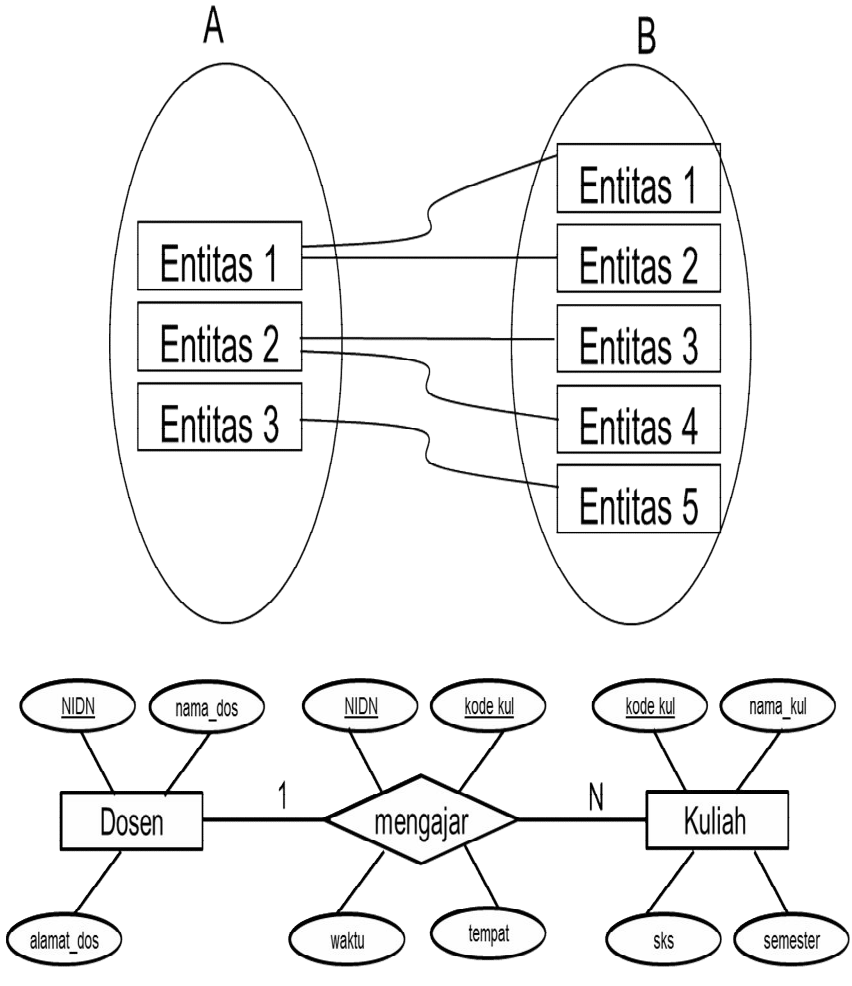
Gambar 4.4 Diagram Relationship

Derajat Relasi Satu ke satu:



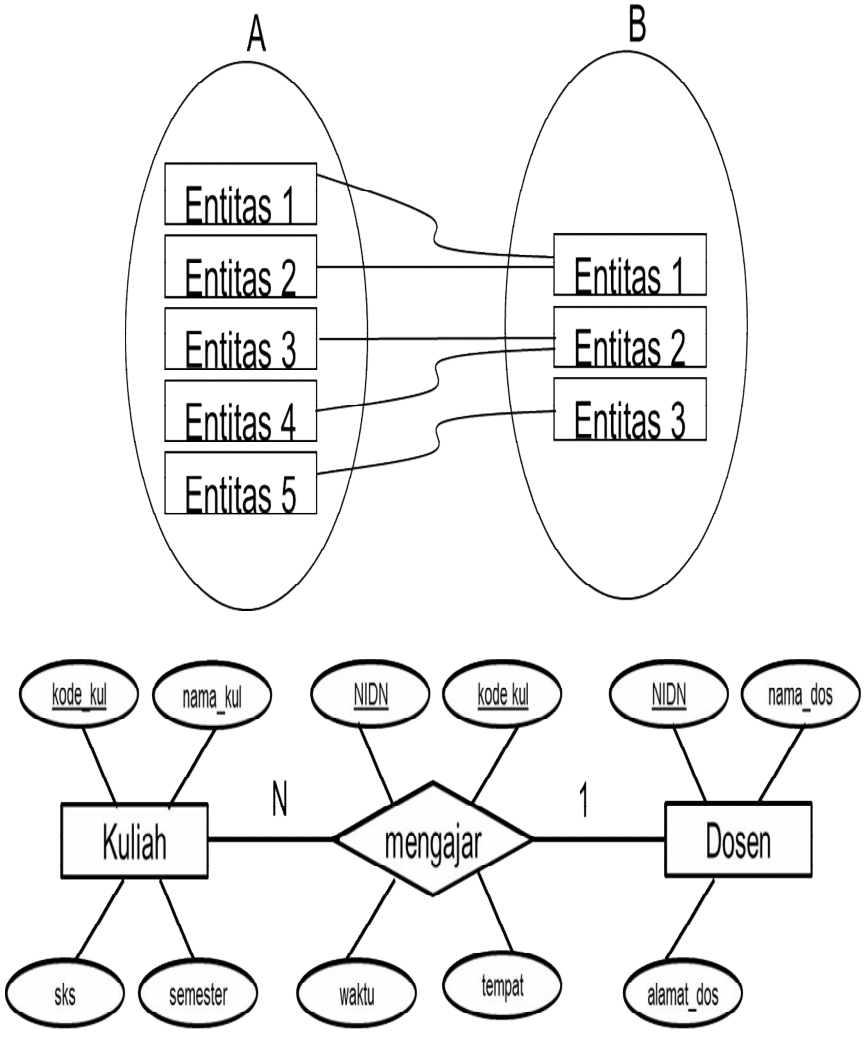
Gambar 4.5 Diagram Relasi Satu Kesatu

Derajat Relasi Satu ke Banyak:



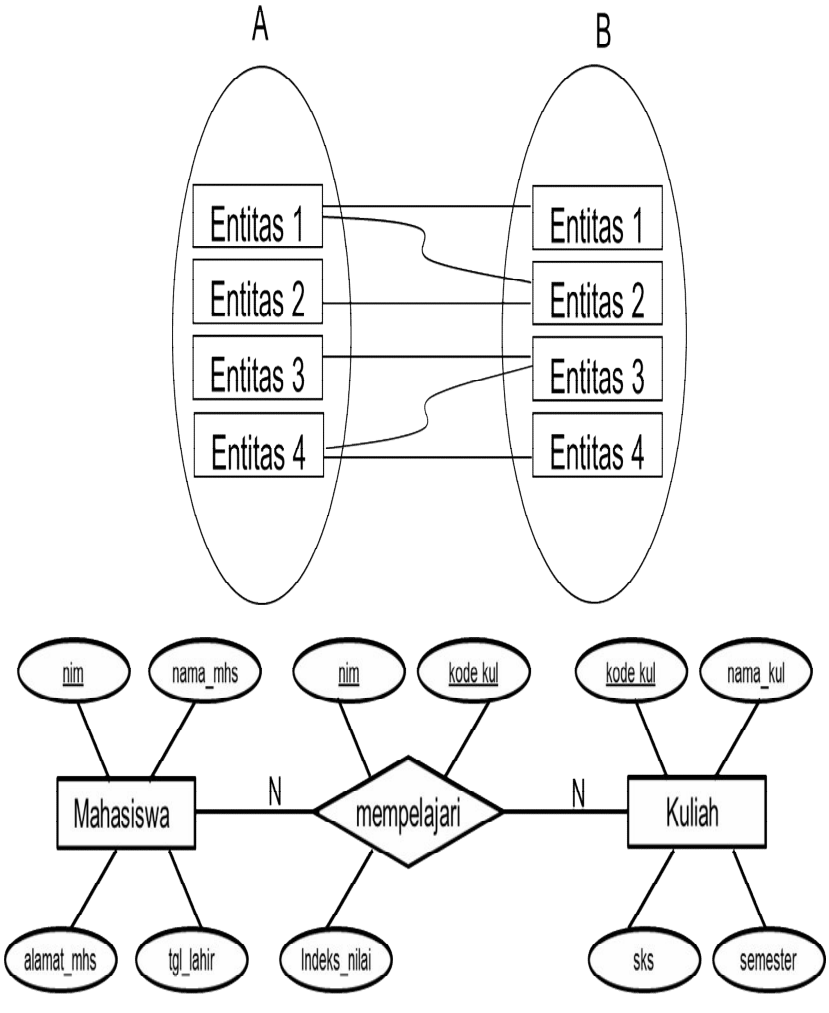
Gambar 4.6 Diagram Relasi Satu ke Banyak

Derajat Relasi Banyak ke Satu:



Gambar 4.7 Diagram Relasi Banyak ke Satu

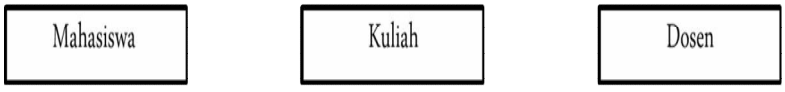
Derajat Relasi Banyak ke Banyak:



Gambar 4.8 Diagram Relasi Banyak ke Banyak

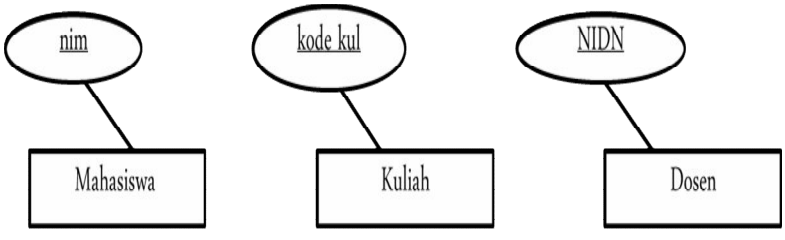
Langkah-langkah teknis yang dapat lakukan untuk membuat merancang Diagram Eentity Relationship:

1. Mengidetifikasi dan menetapkan seluruh himpunan entitas yang akan terlihat.



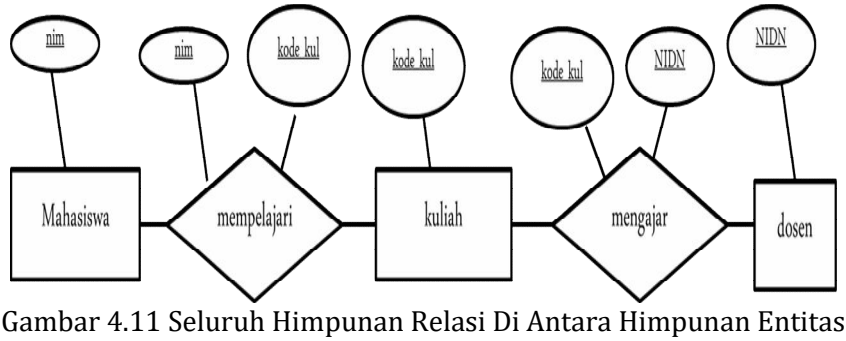
Gambar 4.9 Seluruh Himpunan Entitas

2. Menentukan atribut-atribut *key* dari masing-masing himpunan entitas.



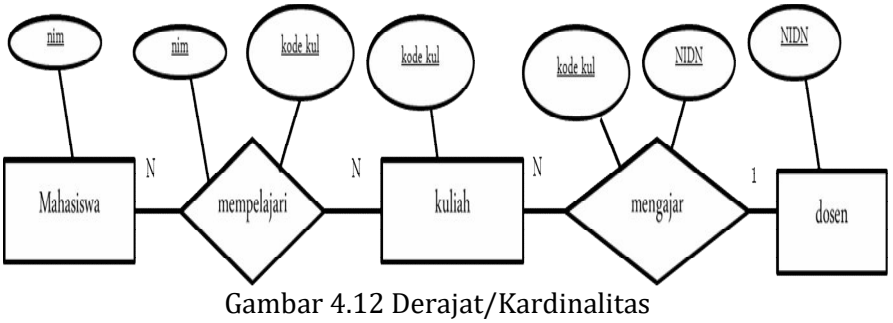
Gambar 4.10 Atribut-Atribut *Key*

3. Mengidetifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas yang ada beserta *foreign-key*nya



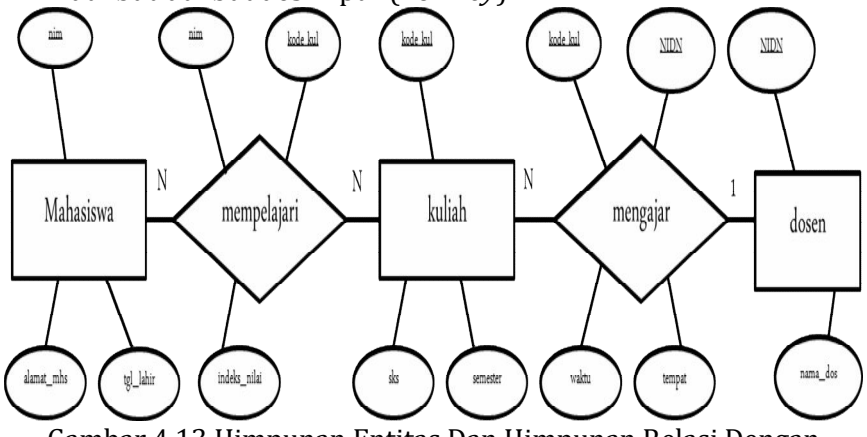
Gambar 4.11 Seluruh Himpunan Relasi Di Antara Himpunan Entitas

4. Menentukan derajat/kardinalitas untuk setiap himpunan relasi.



Gambar 4.12 Derajat/Kardinalitas

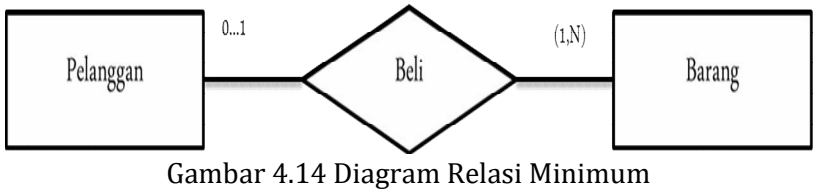
5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif (non *key*).



Gambar 4.13 Himpunan Entitas Dan Himpunan Relasi Dengan Atribut-Atribut Deskriptif

Derajat Relasi Minimum

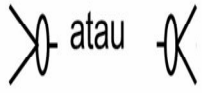
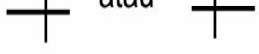
Derajat/Kardinalitas Relasi mewakili hubungan maksimum yang boleh terjadi antara himpunan entitas yang satu terhadap himpunan entitas yang lain. Di samping itu ada pula yang disebut dengan Derajat Relasi Minimum, yang menunjukkan hubungan minimum yang boleh terjadi dalam sebuah relasi antar himpunan entitas. Dalam Diagram Entity Relationship, Derajat Relasi Minimum boleh disertakan, tapi hanya bersifat opsional. Notasinya adalah (x,y) dimana x mewakili Derajat Relasi Minimum dan y mewakili Derajat Relasi Maksimum.



Gambar 4.14 Diagram Relasi Minimum

Derajat Relasi dengan Notasi Lain

Disamping pemakaian notasi yang sudah dijelaskan sebelumnya, dalam berbagai literatur akan dapat dijumpai pula penggambaran Diagram Entity Relationship dengan sedikit perbedaan peggunaan notasi. Perbedaannya terletak pada penggambaran Derajat Relasi yang sekaligus juga telah mengakomodasi adanya Derajat Relasi Minimum.

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

Tabel 4.2 Diagram Relasi dengan Notasi Lain

∞

BAB 5
STRUCTURED QUERY LANGUAGE

SQL (*Structured Query Language*)

Sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

Sejarah

Sejarah SQL dimulai dari artikel seorang peneliti dari IBM bernama EF Codd yang membahas tentang ide pembuatan basis data relasional pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk mengakses data dalam basis data tersebut. Bahasa tersebut kemudian diberi nama **SEQUEL** (Structured English Query Language).

Setelah terbitnya artikel tersebut, IBM mengadakan proyek pembuatan basis data relasional berbasis bahasa SEQUEL. Akan tetapi, karena permasalahan hukum mengenai penamaan SEQUEL, IBM pun mengubahnya menjadi **SQL**. Implementasi basis data relasional dikenal dengan *System/R*.

Di akhir tahun 1970-an, muncul perusahaan bernama Oracle yang membuat server basis data populer yang bernama sama dengan nama perusahaannya. Dengan naiknya kepopuleran Oracle, maka SQL juga ikut populer sehingga saat ini menjadi standar de facto bahasa dalam manajemen basis data.

Standarisasi SQL dimulai pada tahun 1986, ditandai dengan dikeluarkannya standar SQL oleh ANSI. Standar ini sering disebut dengan SQL86. Standar tersebut kemudian diperbaiki pada tahun 1989 kemudian diperbaiki lagi pada tahun 1992. Versi terakhir dikenal dengan SQL92. Pada tahun 1999 dikeluarkan standar baru yaitu SQL99 atau disebut juga SQL99, akan tetapi kebanyakan implementasi mereferensi pada SQL92.

Pemakaian Dasar

SQL (tructured Query Language) adalah bahasa query yang standard yang digunakan sebagai suatu bahasa sederhana dan dasar, yang memungkinkan kita untuk berkomunikasi dengan database,

membaca, menulis, dan memperoleh informasi yang berguna dari database. Meskipun sifatnya non-procedural, lebih mudah bekerja dengan SQL daripada dengan kebanyakan bahasa pemrograman seperti PHP, PERL, Java dan lain-lain, namun kadangkala menyulitkan untuk beberapa kasus yang rumit bagi mereka yang baru mengenal SQL. Perintah atau statement SQL yang paling sederhana yang memungkinkan seorang user dapat menampilkan atau memperoleh data dari suatu tabel adalah perintah atau statement SELECT. Sesuai dengan namanya, dengan perintah SELECT seorang user dapat memilih data yang spesifik dari tabel untuk menampilkannya.

Secara umum, SQL terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap sistem manajemen basis data (SMBD), namun secara umum implementasi tiap bahasa ini memiliki bentuk standar yang ditetapkan ANSI.

SQL Statement

SQL STATEMENT	
SELECT	DATA RETRIEVAL
INSERT UPDATE DELETE MERGE	DML (DATA MANIPULATION LANGUAGE)
CREATE ALTER DROP RENAME TRUNCATE	DDL (DATA DEFINITION LANGUAGE)
COMMIT ROLLBACK SAVEPOINT GRANT REVOKE	DCL (DATA CONTROL LANGUAGE)

Tabel 5.1 SQL Statement

```
CREATE DATABASE `basisdata` ;

CRATE TABLE `basisdata`.`barang` (
`kde_barang` VARCHAR( 7 ) NOT NULL ,
`ama_barang` VARCHAR( 50 ) NOT NULL ,
satuan` VARCHAR( 50 ) NOT NULL ,
`harga` INT NOT NULL ,
`stok` INT NOT NULL ,
PRIMARY KEY ( `kode_barang` )
)

INERT INTO `basisdata`.`barang` (`kode_barang`, `nama_barang`,
    `satuan`, `harga`, `stok`) VALUES
('001', 'Laptop Acer', 'Unit', '5000000', '11'),
(B002', 'Laptop Axio', 'Unit', '3800000', '23'),
'B003', 'Laptop Tosibha', 'Unit', '6000000', '8'),
('B004', 'Keyboard', 'Unit', '30000', '41'),
('B005', 'Mouse', 'Unit', '25000', '29'),
('B006', 'FlashDisc 8 GB', 'Unit', '90000', '53'),
('B007', 'FlashDisc 16 GB', 'Unit', '120000', '55');

CEATE TABLE `basisdata`.`konsumen` (
`ode_konsumen` VARCHAR( 7 ) NOT NULL ,
nama_konsumen` VARCHAR( 50 ) NOT NULL ,
`alamat` VARCHAR( 50 ) NOT NULL ,
`kota` VARCHAR( 25 ) NOT NULL ,
`negara` VARCHAR( 30 ) NOT NULL ,
`telepon` VARCHAR( 20 ) NOT NULL ,
`tanggal_gabung` DATE NOT NULL ,
PRIMARY KEY ( `kode_konsumen` )
```

```
)

INSERT INTO `basisdata`.`konsumen` (`kode_konsumen`, `nama_konsumen`, `alamat`, `kota`, `negara`, `telepon`, `tanggal_gabung`) VALUES
(K001, 'Indra M, Jati','Padang, Indonesia, 081356342188, 2012-11-04),
(K002,'Deddy F, Jati,'Padang,'Indonesia,'081378906734,'2012-11-07),
(K003,'Joe M,'Belimbing,'Padang,'Indonesia,'081298664589,'2012-11-07),
(K004,'Fadlan A,'Sawahan,'Padang,'Indonesia,'081576898900,'2012-11-07),
(K005,'Jona S,'Selayo,'Solok,'Indonesia,'085289008899,'2012-11-10),
(K006,'Mulya,'Selayo,'Solok,'Indonesia,'081363545789,'2012-11-10),
(K07,'Jimmy,'Lima Kaum,'BatuSangkar,'Indonesia,'087856432689,2012-11-11),
(K008,'Reymon,'Jambu Air,'Bukittinggi','Indonesia,'081908567800,2012-11-12);

CEATE TABLE `basisdata`.`supplier` (
`ode_supplier` VARCHAR( 7 ) NOT NULL ,
nama_supplier` VARCHAR( 50 ) NOT NULL ,
`alamat` VARCHAR( 50 ) NOT NULL ,
`kota` VARCHAR( 25 ) NOT NULL ,
`negara` VARCHAR( 30 ) NOT NULL ,
`telepon` VARCHAR( 20 ) NOT NULL ,
`tanggal_gabung` DATE NOT NULL ,
PRIMARY KEY ( `kode_supplier` )
)
```

```
INSERT INTO `basisdata`.`supplier` (`kode_supplier`, `nama_supplier`, `alamat`, `kota`, `negara`, `telepon`, `tanggal_gabung`) VALUES
(S001,'JBros,'Kampung Cina,'Padang,'Indonesia,'075198066,'2012-10-18'),
(S002,'Maxindo,'Kampung Cina,'Padang,'Indonesia,'07517788009,'2012-10-18),
(S003,'Katulistiwa,'Andalas,'Padang,'Indonesia,'0751679011,'2012-10-18),
(S004,'Axio,'Damar,'Padang,'Indonesia,'075188124,'2012-10-18),
(S005,'Kingston,'Jalan Sudirman,'Jakarta,'Indonesia,'021345687,'2012-10-20);

CEATE TABLE `basisdata`.`penjualan` (
`omor_penjualan` INT NOT NULL ,
ode_barang` VARCHAR( 7 ) NOT NULL ,
`kode_konsumen` VARCHAR( 7 ) NOT NULL ,
`jmlah_jual` INT NOT NULL ,
`tanggal_jual` DATE NOT NULL ,
PRIMARY KEY ( `nomor_penjualan`, `kode_barang`, `kode_konsumen` )
)

INERT INTO `basisdata`.`penjualan` (`nomor_penjualan`, `kode_barang`, `kode_konsumen`, `jumlah_jual`, `tanggal_jual`) VALUES
('120001001', 'B001', 'K001', '1', '2012-11-03'),
(2120001002', 'B001', 'K002', '2', '2012-11-03'),
'2120001003', 'B005', 'K004', '5', '2012-11-03'),
('2120001004', 'B004', 'K006', '2', '2012-11-03'),
('2120001005', 'B001', 'K001', '1', '2012-11-04'),
('2120001006', 'B007', 'K008', '5', '2012-11-04'),
('2120001007', 'B002', 'K003', '2', '2012-11-04'),
('2120001008', 'B003', 'K004', '1', '2012-11-04'),
```

('2120001009', 'B001', 'K007', '1', '2012-11-04'),

('2120001010', 'B004', 'K005', '1', '2012-11-05');

CREATE TABLE `basisdata`.`pembelian` (

`nomor_pembelian` INT NOT NULL ,
`nomor_faktur` VARCHAR(25) NOT NULL ,
`kode_barang` VARCHAR(7) NOT NULL ,
`kode_supplier` VARCHAR(7) NOT NULL ,
`jumlah_beli` INT NOT NULL ,
`tanggal_beli` DATE NOT NULL ,
PRIMARY KEY (`nomor_pembelian`, `kode_barang`, `kode_supplier`)
)

INSERT INTO `basisdata`.`pembelian` (`nomor_pembelian`,
`nomor_faktur`, `kode_barang`, `kode_supplier`, `jumlah_beli`,
`tanggal_beli`) VALUES

('1110001001,'F09876,'B001,'S001,'12','2012-10-22),

('1110001001,'F09876,'B002,'S001,'15','2012-10-22),

('1110001001,'F09876,'B003,'S001,'10','2012-10-22),

('1110001002,'KS99102012,'B007,'S005,'15,'2012-10-24),

('1110001002,'KS99102012,'B006,'S005,'30,'2012-10-24),

('1110001003,'556/9906/12,'B004,'S002,'20,'2012-10-25),

('1110001003,'556/9906/12,'B005,'S002,'20','2012-10-25);

SQL Select Statements

SELECT * |{[DISTINCT] column| expression [alias],....} FROM table

1. Melihat seluruh isi tabel

   Input:

   SELECT * FROM barang

   Output:

kode_barang	nama_barang	satuan	harga	stok
B001	Laptop Acer	Unit	5000000	11
B002	Laptop Axio	Unit	3800000	23
B003	Laptop Tosibha	Unit	6000000	8
B004	Keyboard	Unit	30000	41
B005	Mouse	Unit	25000	29
B006	FlashDisc 8 GB	Unit	90000	53
B007	FlashDisc 16 GB	Unit	120000	55

Tabel 5.2 Contoh 1

2. Melihat sebagian dari isi tabel

   Input:

   SELECT kode_barang, nama_barang, satuan FROM barang

   *Output*i *:*

kode_barang	nama_barang	satuan
B001	Laptop Acer	Unit
B002	Laptop Axio	Unit
B003	Laptop Tosibha	Unit
B004	Keyboard	Unit
B005	Mouse	Unit
B006	FlashDisc 8 GB	Unit
B007	FlashDisc 16 GB	Unit

Tabel 5.3 Contoh 2

3. Menggunakan Operator Aritmatik

   Prioritas: * / + -

   input*:*

   SELECT kode_barang, nama_barang, harga+250 FROM barang

   Output:

kode_barang	nama_barang	harga
B001	Laptop Acer	5000250
B002	Laptop Axio	3800250
B003	Laptop Tosibha	6000250
B004	Keyboard	30250
B005	Mouse	25250
B006	FlashDisc 8 GB	90250
B007	FlashDisc 16 GB	120250

Tabel 5.4 Contoh 3

4. Menggunakan kolom alias

   Input:

   SLECT kode_barang AS 'KODE BARANG', nama_barang NAMA FROM barang

   Output:

KODE BARANG	NAMA
B001	Laptop Acer
B002	Laptop Axio
B003	Laptop Tosibha
B004	Keyboard
B005	Mouse
B006	FlashDisc 8 GB
B007	FlashDisc 16 GB

Tabel 5.5 Contoh 4

5. Duplikasi Baris

   Input:

   SLECT alamat FROM konsumen

   Output:

alamat
Jati
Jati
Belimbing
Sawahan
Selayo
Selayo
Lima Kaum
Jambu Air

Tabel 5.6 Contoh 5

6. Menghindari Duplikasi Baris

   Input:

   SELECT DISTINCT alamat FROM konsumen

   Output:

alamat
Jati
Belimbing
Sawahan
Selayo
Lima Kaum
Jambu Air

Tabel 5.7 Contoh 6

7. Menampilkan struktur dari Tabel

   Input:

   DESC barang

   Output:

Field	Type	Null	Key	Default	Extra
kode_barang	varchar(7)	NO	PRI		
nama_barang	varchar(50)	NO			
satuan	varchar(50)	NO			
harga	int(11)	NO			
stok	int(11)	NO			

Tabel 5.8 Contoh 7

8. Menggunakan Rangkaian Operator*

   Input:

   Select kode_barang || nama_barang from barang

9. Menggunakan Karakter String*

   Input:

   Select kode_barang || 'dan' || nama_barang from barang

   *: hanya berjalan pada DBMS Oracle

Restricting and Sorting Data

1. Menggunakan ketentuan WHERE
   SELECT * |{[DISTINCT] column| expression [alias],....}
   FROM table
   WHERE condition

Kondisi pembanding

Operator	Keterangan
=	Sama dengan
>	Lebih besar
>=	Lebih besar atau sama dengan
<	Lebih kecil
<=	Lebih kecil atau sama dengan
<>	Tidak sama dengan

Tabel 5.9 Kondisi Pembanding

Input A:
SELECT * FROM konsumen WHERE kota = 'Padang'

Output A:

kode_konsumen	nama_konsumen	alamat	kota	negara	telepon	tanggal_gabung
K001	Indra M	Jati	Padang	Indonesia	081356342188	2012-11-04
K002	Deddy F	Jati	Padang	Indonesia	081378906734	2012-11-07
K003	Joe M	Belimbing	Padang	Indonesia	081298664589	2012-11-07
K004	Fadlan A	Sawahan	Padang	Indonesia	081576898900	2012-11-07

Tabel 5.10 Contoh 8

Input B:
SELECT nama_barang, harga FROM barang WHERE harga > 90000

Output B:

nama_barang	harga
Laptop Acer	5000000
Laptop Axio	3800000
Laptop Tosibha	6000000
FlashDisc 16 GB	120000

Tabel 5.11 Contoh 9

Input C:
SELECT nama_barang, harga FROM barang WHERE harga >= 90000

Output C:

nama_barang	harga
Laptop Acer	5000000
Laptop Axio	3800000
Laptop Tosibha	6000000
FlashDisc 8 GB	90000
FlashDisc 16 GB	120000

T

abel 5.12 Contoh 10

Input D:
SELECT nama_barang, harga FROM barang WHERE harga < 90000

Output D:

nama_barang	harga
Keyboard	30000
Mouse	25000

Tabel 5.13 Contoh 11

Input E:
SELECT nama_barang, harga FROM barang WHERE harga <= 90000

Output E:

nama_barang	harga
Keyboard	30000
Mouse	25000
FlashDisc 8 GB	90000

Tabel 5.14 Contoh 12

Input F:
SELECT * FROM konsumen WHERE kota <> 'Padang‘

Output F:

kode_konsumen	nama_konsumen	alamat	kota	negara	telepon	tanggal_gabung
K005	Jona S	Selayo	Solok	Indonesia	085289008899	2012-11-10
K006	Mulya	Selayo	Solok	Indonesia	081363545789	2012-11-10
K007	Jimmy	Lima Kaum	Batu Sangkar	Indonesia	087856432689	2012-11-11
K008	Reymond	Jambu Air	Bukittinggi	Indonesia	081908567800	2012-11-12

Tabel 5.15 Contoh 13

Kondisi Pembanding Lainnya

Operator	Keterangan
BETWEEN	Diantara dua nilai
IN	Didalam suatu nilai
LIKE	Memiliki suatu unsur karakter
IS NULL	Bernilai NULL
IS NOT NULL	Tidak bernilai NULL

Tabel 5.16 Kondisi Pembanding Lainnya

Input A:
SELECT * FROM barang WHERE stok BETWEEN 25 AND 50

Output A:

kode_barang	nama_barang	satuan	harga	stok
B004	Keyboard	Unit	30000	41
B005	Mouse	Unit	25000	29

Tabel 5.16 Contoh 14

input B:
SELECT * FROM barang WHERE stok IN(11, 41, 55)

Output B:

kode_barang	nama_barang	satuan	harga	stok
B001	Laptop Acer	Unit	5000000	11
B004	Keyboard	Unit	30000	41
B007	FlashDisc 16 GB	Unit	120000	55

Tabel 5.17 Contoh 15

Input C:
SELECT nama_barang FROM barang WHERE nama_barang LIKE '%D%‘

Output C:

nama_barang
Keyboard
FlashDisc 8 GB
FlashDisc 16 GB

Tabel 5.18 Contoh 16

Input D:
SELECT nama_barang, satuan FROM barang WHERE satuan IS NULL

Output D:

nama_barang	satuan
FlashDisc 16 GB	NULL

Tabel 5.19 Contoh 17

Input E:
SELECT nama_barang, satuan FROM barang WHERE satuan IS NOT NULL

Output E:

nama_barang	satuan
Laptop Acer	Unit
Laptop Axio	Unit
Laptop Tosibha	Unit
Keyboard	Unit
Mouse	Unit
FlashDisc 8 GB	Unit

Tabel 5.20 Contoh 18

Kondisi logika

Operator	Keterangan
AND	Jika semua kondisi benar
OR	Jika salah satu kondisi benar
NOT	Jika kondisi salah

Tabel 5.21 Kondisi logika

Input A:
SELECT nama_supplier, kota, alamat FROM supplier WHERE kota = 'Padang' AND alamat = 'Kampung Cina'

Output A:

nama_supplier	kota	alamat
JBros	Padang	Kampung Cina
Maxindo	Padang	Kampung Cina

Tabel 5.22 Contoh 19

Input B:
SELECT nama_supplier, kota, alamat FROM supplier WHERE kota = 'Surabaya' OR kota = 'Jakarta'

Output B:

nama_supplier	kota	alamat
Kingston	Jakarta	Jalan Sudirman

Tabel 5.23 Contoh 20

Input C:
SELECT nama_supplier, kota, alamat FROM supplier WHERE kota = 'Surabaya' OR kota = 'Jakarta'

Output C:

kode_barang	nama_barang	satuan	harga	stok
B002	Laptop Axio	Unit	3800000	23
B003	Laptop Tosibha	Unit	6000000	8
B005	Mouse	Unit	25000	29
B006	FlashDisc 8 GB	Unit	90000	53

Tabel 5.24 Contoh 21

Urutan Ascending

Input A:

SELECT * FROM supplier ORDER BY nama_supplier ASC

Output A:

kode_supplier	nama_supplier	alamat	kota	negara	telepon	tanggal_gabung
S004	Axio	Damar	Padang	Indonesia	075188124	2012-10-18
S001	JBros	Kampung Cina	Padang	Indonesia	075198066	2012-10-18
S003	Katulistiwa	Andalas	Padang	Indonesia	0751679011	2012-10-18
S005	Kingston	Jalan sudirman	Jakarta	Indonesia	021345687	2012-10-20
S002	Maxindo	Kampung Cina	Padang	Indonesia	07517788009	2012-10-18

Tabel 5.25 Contoh 22

Urutan Descending

Input A:
SELECT * FROM supplier ORDER BY nama_supplier DESC

Output A:

kode_supplier	nama_supplier	alamat	kota	negara	telepon	tanggal_gabung
S002	Maxindo	Kampung Cina	Padang	Indonesia	07517788009	2012-10-18
S005	Kingston	Jalan sudirman	Jakarta	Indonesia	021345687	2012-10-20
S003	Katulistiwa	Andalas	Padang	Indonesia	0751679011	2012-10-18
S001	JBros	Kampung Cina	Padang	Indonesia	075198066	2012-10-18
S004	Axio	Damar	Padang	Indonesia	075188124	2012-10-18

Tabel 5.26 Contoh 23

Single-Row Functions

SELECT * | Function_name [(arg1, arg2, ...)]
FROM table

General Function

Operator	Keterangan
NVL	Mengubah nilai NULL dengan nilai yang aktual*
NVL2	Mengubah nilai NULL dengan nilai yang bertipe karakter*
NULLIF	Membandingkan 2 nilai dan jika nilai tersebut sama maka akan menghasilkan NULL, tapi jika nilai tersebut berbeda maka nilai pertama yang akan ditampilkan
COALSECE	Jika hasil pertama adalah NULL, maka akan ditampilkan hasil dari nilai kedua. Jika hasl kedua adalah NULL, maka yang tampil nilai ketiga dan seterusnya
CASE	Membuat ekspresi aturan (IF-THEN-ELSE)
DECODE	Membuat ekspresi aturan (IF-THEN-ELSE)*

Tabel 5.27 General Function
*: hanya berjalan pada DBMS Oracle

SELECT * | Function_name [(arg1, arg2, ...)]
FROM table

1. General Function

NVL
Input:
SELECT nama_barang, NVL(stok, 0) FROM barang

NVL2
Input:
SELECT nama_barang, stok, NVL2(stok, 'NOL','KOSONG') FROM barang

NULLIF
Input:
SELECT nama_konsumen, LENGTH(nama_konsumen) "NAMA", kota, LENGTH(kota) "KOTA", NULLIF(LENGTH (nama_konsumen), LENGTH(kota)) "HASIL" FROM konsumen

Output:

nama_konsumen	NAMA	kota	KOTA	HASIL
Indra M	7	Padang	6	7
Deddy F	7	Padang	6	7
Joe M	5	Padang	6	5
Fadlan A	8	Padang	6	8
Jona S	6	Solok	5	6
Mulya	5	Solok	5	*NULL*
Jimmy	5	Batu Sangkar	12	5
Reymond	7	Bukittinggi	11	7

Tabel 5.27 Contoh 24

Coalesce

Input:
SELECT nama_barang, stok, harga, COALESCE(stok, harga, 10) "HASIL" FROM barang

Output:

nama_barang	stok	harga	HASIL
Laptop Acer	11	5000000	11
Laptop Axio	23	3800000	23
Laptop Tosibha	8	6000000	8
Keyboard	41	30000	41
Mouse	29	25000	29
FlashDisc 8 GB	53	90000	53
FlashDisc 16 GB	*NULL*	120000	120000
Laptop Acer	11	5000000	11

Tabel 5.28 Contoh 25

CASE

Input:
SELECT nama_barang, stok, CASE stok WHEN 8 THEN stok+5 WHEN 11 THEN stok+10 ELSE 0 END "HASIL" FROM barang

Output:

nama_barang	stok	harga	HASIL
Laptop Acer	11	5000000	11
Laptop Axio	23	3800000	23
Laptop Tosibha	8	6000000	8
Keyboard	41	30000	41
Mouse	29	25000	29
FlashDisc 8 GB	53	90000	53
FlashDisc 16 GB	*NULL*	120000	120000
Laptop Acer	11	5000000	11

Tabel 5.29 Contoh 26

DECODE

Input:
SELECT nama_barang, stok, DECODE(stok, 8, stok+5, 11, stok+10, 0) "HASIL" FROM barang

Output :

Multiple Table

Salah satu fitur SQL yang paling berguna adalah kemampuan untuk mengabungkan tabel on-the-fly dengan query-query yang mendapatkan kembali data, tetapi sebelum mempelajari ini kita harus terlebih dahulu memahami tabel relasional dan rancangan database relasional, karena jika tidak akan menyebabkan terjadinya Cartesian Product.

Cartesian Product adalah Hasil yang dikembalikan oleh sebuah tabel relasional tanpa kondisi JOIN. Jumlah baris yang didapatkan kembali akan menjadi jumlah baris dalam tabel pertama dikalikan dengan jumlah baris pada tabel kedua.

Input:
SELECT * FROM barang, penjualan

Output:

kode_barang	nama_barang	satuan	harga	stok	nomor_penjualan	kode_barang	kode_konsumen	tanggal_jual
B001	Laptop Acer	Unit	5000000	11	2120001001	B001	K001	1
B002	Laptop Axio	Unit	3800000	23	2120001001	B001	K001	1
B003	Laptop Tosibha	Unit	6000000	8	2120001001	B001	K001	1
B004	Keyboard	Unit	30000	41	2120001001	B001	K001	1
B005	Mouse	Unit	25000	29	2120001001	B001	K001	1
B006	FlashDisc 8 GB	Unit	90000	53	2120001001	B001	K001	1
B007	FlashDisc 16 GB	Unit	120000	55	2120001001	B001	K001	1
B001	Laptop Acer	Unit	5000000	11	2120001002	B001	K002	2

Tabel 5.30 Contoh 27

Jika dilihat dari hasil diatas terdapat 100 rows selected padahal didalam contoh datanya hanya sebanyak 10 rows selected, hal diatas terjadi karena tidak adanya Filter sehingga dalam prosesnya komputer mengkombinasikan kedua tabel diatas.

Input:

```
SELECT * FROM barang, penjualan where penjualan.kode_barang
= barang.kode_barang
```

Output:

kode_barang	nama_barang	satuan	harga	stok	nomor_penjualan	kode_barang	kode_konsumen	tanggal_jual
B001	Laptop Acer	Unit	5000000	11	2120001001	B001	K001	1
B001	Laptop Acer	Unit	5000000	11	2120001002	B001	K002	2
B005	Mouse	Unit	25000	29	2120001003	B005	K004	5
B004	Keyboard	Unit	30000	41	2120001004	B004	K006	2
B001	Laptop Acer	Unit	5000000	11	2120001005	B001	K001	1
B007	FlashDisc 16 GB	Unit	120000	55	2120001006	B007	K008	5
B002	Laptop Axio	Unit	3800000	23	2120001007	B002	K003	2
B003	Laptop Tosibha	Unit	6000000	8	2120001008	B003	K004	1
B001	Laptop Acer	Unit	5000000	11	2120001009	B001	K007	1
B004	Keyboard	Unit	30000	41	2120001010	B004	K005	1

Tabel 5.31 Contoh 28

Menampilkan sebagian dari multiple table
Input:
SELECT penjualan.nomor_penjualan, barang.nama_barang, penjualan.jumlah_jual FROM barang, penjualan where penjualan.kode_barang = barang.kode_barang

Output:

nomor_penjualan	nama_barang	jumlah_jual
2120001001	Laptop Acer	1
2120001002	Laptop Acer	2
2120001003	Mouse	5
2120001004	Keyboard	2
2120001005	Laptop Acer	1
2120001006	FlashDisc 16 GB	5
2120001007	Laptop Axio	2
2120001008	Laptop Tosibha	1
2120001009	Laptop Acer	1
2120001010	Keyboard	1

Tabel 5.32 Contoh 29

Menggunakan kolom alias

Input:
SELECT t1.nomor_penjualan AS NOMOR, t2.nama_barang AS BARANG, t1.jumlah_jual AS JUMLAH FROM penjualan t1, barang t2 where t1.kode_barang = t2.kode_barang

Output:

NOMOR	BARANG	JUMLAH
2120001001	Laptop Acer	1
2120001002	Laptop Acer	2
2120001003	Mouse	5
2120001004	Keyboard	2
2120001005	Laptop Acer	1
2120001006	FlashDisc 16 GB	5
2120001007	Laptop Axio	2
2120001008	Laptop Tosibha	1
2120001009	Laptop Acer	1
2120001010	Keyboard	1

Tabel 5.33 Contoh 30

Menggunakan kondisi logika

Input:

```
SELECT penjualan.nomor_penjualan, barang.nama_barang,
penjualan.jumlah_jual FROM barang, penjualan WHERE
penjualan.kode_barang = barang.kode_barang AND
penjualan.jumlah_jual = '5'
```

Output:

nomor_penjualan	nama_barang	jumlah_jual
2120001003	Mouse	5
2120001006	FlashDisc 16 GB	5

Tabel 5.34 Contoh 31

Menampilkan sebagian isi tabel dengan menggunakan kolom alias

Input:
SELECT t1.nomor_penjualan AS NOMOR, t3.nama_konsumen AS KONSUMEN, t2.nama_barang AS BARANG, t2.harga AS HARGA, t1.jumlah_jual AS JUMLAH FROM penjualan t1, barang t2, konsumen t3 where t1.kode_barang = t2.kode_barang AND t1.kode_konsumen = t3.kode_konsumen

Output:

NOMOR	KONSUMEN	BARANG	HARGA	JUMLAH
2120001001	Indra M	Laptop Acer	5000000	1
2120001002	Deddy F	Laptop Acer	5000000	2
2120001003	Fadlan A	Mouse	25000	5
2120001004	Mulya	Keyboard	30000	2
2120001005	Indra M	Laptop Acer	5000000	1
2120001006	Reymond	FlashDisc 16 GB	120000	5
2120001007	Joe M	Laptop Axio	3800000	2
2120001008	Fadlan A	Laptop Tosibha	6000000	1
2120001009	Jimmy	Laptop Acer	5000000	1
2120001010	Jona S	Keyboard	30000	1

Tabel 5.35 Contoh 32

Natural Join

Input:
SELECT kode_barang, nama_barang, jumlah_beli FROM pembelian NATURAL JOIN barang

Output:

kode_barang	nama_barang	jumlah_beli
B001	Laptop Acer	12
B002	Laptop Axio	15
B003	Laptop Tosibha	10
B006	FlashDisc 8 GB	30
B007	FlashDisc 16 GB	15
B004	Keyboard	20
B005	Mouse	20

Tabel 5.36 Contoh 33

Group Function

Pada kasus seleksi data seringkali diminta untuk menampilkan atau memilih sekumpulan data berdasarkan kelompok data tertentu. Untuk menyelesaikan masalah tersebut, SQL menyediakan perintah atau sintax Group By. Pada pengelompokan data biasanya disertakan bersama Aggregate Function. Dalam hal ini implementasinya, kumpulan Function harus diikuti group by bila terdapat field lain yang dijadikan sebagai kriteria pengelompokkan.

```
SELECT column, group_function(column)
FROM table [WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

```
SELECT column, group_function(column)
FROM table [WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

Tipe dari Group Function

Tipe dari Group Function

Operator	Keterangan
AVG()	digunakan untuk mencari nilai rata-rata dari sekumpulan data
COUNT()	digunakan untuk mencari cacah atau banyaknya data
MAX()	digunakan untuk mencari nilai maximum dari sekumpulan data
MIN()	digunakan untuk mencari nilai minimum dari sekumpulan data
STDDEV()	digunakan untuk menghitung standar deviasi dari sekumpulan data
SUM()	digunakan untuk mencari nilai total dari sekumpulan data
VARIANCE()	digunakan untuk menghitung nilai variance dari sekumpulan data

Tabel 5.37 Tipe dari Group Function

1. AVG()
Input:
SELECT nomor_faktur, AVG(jumlah_beli) FROM pembelian GROUP BY nomor_faktur

Output:

nomor_faktur	AVG(jumlah_beli)
556/9906/12	20.0000
F09876	12.3333
KS99102012	22.5000

Tabel 5.38 Contoh 34

2. COUNT()
Input:
SELECT nomor_faktur, COUNT(kode_supplier) FROM pembelian GROUP BY nomor_faktur

Output:

Tabel 5.39 Contoh 35

nomor_faktur	COUNT(kode_supplier)
556/9906/12	2
F09876	3
KS99102012	2

3. MAX()
Input:
SELECT MAX(harga) FROM barang GROUP BY satuan

Output:

MAX(harga)
6000000

Tabel 5.40 Contoh 36

4. MIN()
Input:
SELECT MIN(harga) FROM barang GROUP BY satuan

Output:

MIN(harga)
25000

Tabel 5.41 Contoh 37

5. STDDEV()
Input:
SELECT STDDEV(stok) , MAX(stok) FROM barang GROUP BY satuan

Output:

STDDEV(stok)	MAX(stok)
17.5488	55

Tabel 5.42 Contoh 38

6. SUM()
Input:
SELECT MIN(harga), MAX(harga), SUM(harga) FROM barang GROUP BY satuan

Output:

MIN(harga)	MAX(harga)	SUM(harga)
25000	6000000	15065000

Tabel 5.43 Contoh 39

7. VARIANCE()
Input:
SELECT VARIANCE(stok) , MAX(stok)FROM barang GROUP BY satuan
Output:

VARIANCE(stok)	MAX(stok)
307.9592	55

Tabel 5.44 Contoh 40

Having
Input:
SELECT nama_barang, Max(harga) FROM barang GROUP BY kode_barang HAVING max(harga) > 150000 ORDER BY nama_barang DESC

Output:

nama_barang	Max(harga)
Laptop Tosibha	6000000
Laptop Axio	3800000
Laptop Acer	5000000

Tabel 5.45 Contoh 41

Subqueries
Subquery adalah Query didalam query.
Artinya seleksi data berdasarkan hasil seleksi data yang telah ada.
Sintax SQL nya sama syntax SQL pada umumnya, tetapi kondisi setelah where diikuti dengan query baru atau subquery.

Syntaxnya :
SELECT nama_field-1,.....,nama_field-n
FROM nama_tabel
WHERE kriteria (SELECT nama_field-1, ..., nama_field-n
FROM nama_tabel
WHERE kriteria)

Input:
SELECT nomor_pembelian, nomor_faktur, kode_barang FROM pembelian WHERE kode_barang = (select kode_barang FROM barang WHERE stok = 11)

Output:

nomor_pembelian	nomor_faktur	kode_barang
1110001001	F09876	B001

Tabel 5.46 Contoh 42

Input:
SELECT nomor_pembelian, nomor_faktur, (select sum(stok) FROM barang GROUP BY satuan)
FROM pembelian

Output:

nomor_pembelian	nomor_faktur	(select sum(stok) FROM barang GROUP BY satuan)
1110001001	F09876	220
1110001001	F09876	220
1110001001	F09876	220
1110001002	KS99102012	220
1110001002	KS99102012	220
1110001003	556/9906/12	220
1110001003	556/9906/12	220

Tabel 5.47 Contoh 43

Initial

Input:

SELECT konsumen.nama_konsumen, barang.nama_barang, barang.satuan, barang.harga, penjualan.jumlah_jual, penjualan.tanggal_jual FROM barang, penjualan, konsumen WHERE barang.kode_barang = penjualan.kode_barang AND konsumen.kode_konsumen = penjualan.kode_konsumen

Output:

nama_konsumen	nama_barang	satuan	harga	jumlah_jual	tanggal_jual
Indra M	Laptop Acer	Unit	5000000	1	2012-11-03
Deddy F	Laptop Acer	Unit	5000000	2	2012-11-03
Fadlan A	Mouse	Unit	25000	5	2012-11-03
Mulya	Keyboard	Unit	30000	2	2012-11-03
Indra M	Laptop Acer	Unit	5000000	1	2012-11-04
Reymond	FlashDisc 16 GB	Unit	120000	5	2012-11-04
Joe M	Laptop Axio	Unit	3800000	2	2012-11-04
Fadlan A	Laptop Tosibha	Unit	6000000	1	2012-11-04
Jimmy	Laptop Acer	Unit	5000000	1	2012-11-04
Jona S	Keyboard	Unit	30000	1	2012-11-05

Tabel 5.48 Contoh 44

Input:
SELECT k.nama_konsumen, b.nama_barang, b.satuan, b.harga, p.jumlah_jual, p.tanggal_jual FROM barang b, penjualan p, konsumen k WHERE b.kode_barang = p.kode_barang AND k.kode_konsumen = p.kode_konsumen

Output:

nama_konsumen	nama_barang	satuan	harga	jumlah_jual	tanggal_jual
Indra M	Laptop Acer	Unit	5000000	1	2012-11-03
Deddy F	Laptop Acer	Unit	5000000	2	2012-11-03
Fadlan A	Mouse	Unit	25000	5	2012-11-03
Mulya	Keyboard	Unit	30000	2	2012-11-03
Indra M	Laptop Acer	Unit	5000000	1	2012-11-04
Reymond	FlashDisc 16 GB	Unit	120000	5	2012-11-04
Joe M	Laptop Axio	Unit	3800000	2	2012-11-04
Fadlan A	Laptop Tosibha	Unit	6000000	1	2012-11-04
Jimmy	Laptop Acer	Unit	5000000	1	2012-11-04
Jona S	Keyboard	Unit	30000	1	2012-11-05

Tabel 5.49 Contoh 45

Pembuatan View

Input:
CREATE VIEW v_beli AS SELECT s.nama_supplier AS SUPPLIER, b.nama_barang AS BARANG, b.satuan AS SATUAN, b.harga AS HARGA, p.jumlah_beli AS JUMLAH, b.harga*p.jumlah_beli AS TOTAL FROM supplier s, barang b, pembelian p WHERE p.kode_supplier = s.kode_supplier AND p.kode_barang = b.kode_barang
Output:

Tabel 5.50 Contoh 46

SUPPLIER	BARANG	SATUAN	HARGA	JUMLAH	TOTAL
JBros	Laptop Acer	Unit	5000000	12	60000000
JBros	Laptop Axio	Unit	3800000	15	57000000
JBros	Laptop Tosibha	Unit	6000000	10	60000000
Maxindo	Keyboard	Unit	30000	20	600000
Maxindo	Mouse	Unit	25000	20	500000
Kingston	FlashDisc 8 GB	Unit	90000	30	2700000
Kingston	FlashDisc 16 GB	Unit	120000	15	1800000

∞

This page is intentionally left blank

DAFTAR PUSTAKA

Batini, Carlo, Stefano Ceri, and Shamkant B. Navathe. *Conceptual database design: an Entity-relationship approach*. Vol. 116. Redwood City, CA: Benjamin/Cummings, 1992.

Indrajani. *Database Design All in One: Theory, Practice, and Case Study*. Elex Media Komputindo, 2018

RIWAYAT PENULIS

Basis adalah Gudang / markas / tempat berkumpul / tempat bersarang. Data adalah Representasi fakta dunia nyata yang mewakili suatu obyek (manusia, benda, kejadian, dll) yang disimpan dalam bentuk teks, angka, gambar, bunyi, simbol, atau kombinasinya

Basis Data adalah Kumpulan dari item data yang saling berhubungan satu dengan lainnya yang diorganisasikan berdasar sebuah skema atau struktur tertentu, tersimpan di hardware komputer dan dengan software digunakan untuk melakukan manipulasi data (diperbaharui, dicari, diolah dengan perhitungan-perhitungan tertentu, dan dihapus) dengan tujuan tertentu

ISBN 978-602-464-078-1

9 786024 640781

UNIMAL PRESS