SISTEM BASIS DATA

I Putu Dody S, Michael Sitorus, Rifka Widyastuti, Deddy Kurniawan, Satya Arisena H, Vivi Afifah, Dwipo Setyantoro, Akhmad Pandhu Wijaya, Immanuela Puspasari S, Neneng Rachmalia Feta, Diky Wardhani, I Gede Agus Suwartane, Ahmad Rosadi, Muhammad Anno Suwarno

Sistem Basis Data

I Putu Dody Suarnatha, Michael Sitorus, Rifka Widyastuti, Deddy Kurniawan, Satya Arisena H, Vivi Afifah, Dwipo Setyantoro, Akhmad Pandhu Wijaya, Immanuela Puspasari S, Neneng Rachmalia Feta, Diky Wardhani, I Gede Agus Suwartane, Ahmad Rosadi, Muhammad Anno Suwarno

PT. MIFANDI MANDIRI DIGITAL

Sistem Basis Data

Penulis:
I Putu Dody Suarnatha, Michael Sitorus, Rifka Widyastuti, Deddy Kurniawan, Satya Arisena H, Vivi Afifah, Dwipo Setyantoro, Akhmad Pandhu Wijaya, Immanuela Puspasari S, Neneng Rachmalia Feta, Diky Wardhani, I Gede Agus Suwartane, Ahmad Rosadi, Muhammad Anno Suwarno

ISBN: 978-623-09-2563-4

Editor:
Sarwandi

Penyunting:
Sinta Ulina Situmorang

Desain sampul dan Tata Letak:
Sarwandi

Penerbit:
PT. Mifandi Mandiri Digital

Redaksi:
Komplek Senda Residence Jl. Payanibung Ujung D
Dalu Sepuluh-B Tanjung Morawa
Kab. Deli Serdang Sumatera Utara

Distributor Tunggal:
PT. Mifandi Mandiri Digital
Komplek Senda Residence Jl. Payanibung Ujung D
Dalu Sepuluh-B Tanjung Morawa
Kab. Deli Serdang Sumatera Utara

Cetakan Pertama, Januari 2023

KATA PENGANTAR

Sistem basis data telah menjadi komponen penting dalam dunia teknologi informasi saat ini. Sebagai sebuah sistem yang menyimpan dan mengorganisir data secara terstruktur, sistem basis data memungkinkan pengguna untuk dengan mudah mengakses informasi yang diperlukan dengan cepat dan akurat.

Buku ini dirancang untuk memberikan pemahaman yang komprehensif tentang sistem basis data, mulai dari konsep dasar hingga teknologi terbaru yang digunakan dalam pengembangan dan pengelolaan basis data. Buku ini ditulis untuk pembaca yang ingin memperdalam pengetahuan mereka tentang sistem basis data, seperti mahasiswa teknologi informasi, pengembang perangkat lunak, dan profesional IT.

Dalam buku ini, pembaca akan belajar tentang berbagai jenis sistem basis data, arsitektur sistem basis data, teknik pemodelan data, bahasa pemrograman database, pengoptimalan kinerja database, dan banyak lagi. Selain itu, buku ini juga menampilkan beberapa studi kasus dan contoh penggunaan sistem basis data dalam berbagai bidang, seperti bisnis, kesehatan, dan keamanan.

Saya berharap buku ini dapat memberikan kontribusi yang signifikan bagi pembaca dalam memahami sistem basis data dan menerapkannya secara efektif dalam pengembangan perangkat lunak dan pemrosesan data. Terima kasih telah memilih buku ini dan semoga membantu Anda dalam memperoleh pemahaman yang lebih dalam tentang sistem basis data.

Medan, Januari 2023

Penulis

DAFTAR ISI

BAB

I

Konsep Dasar Basis Data

Pendahuluan

Basis Data (*database*), di zaman sekarang ini berperan dan memiliki dampak yang besar dalam kehidupan masyarakat. Basis data memiliki kaitan yang sangat penting dalam kemajuan berbagai bidang salah satunya bidang rekayasa perangkat lunak, serta basis data menjadi kerangka utama yang menjadi pondasi sistem informasi serta secara mendasar mengubah berbagai cara yang digunakan organisasi dalam beroperasi. Contoh penggunaan *database* pada program komputer seperti : aplikasi absensi karyawan, aplikasi peminjaman buku pada perpustakaan, aplikasi stok barang, dan lainnya.

Pada Bab ini diulas terkait konsep dasar dalam basis data, serta mengapa basis data itu penting dalam suatu organisasi/perusahaan.

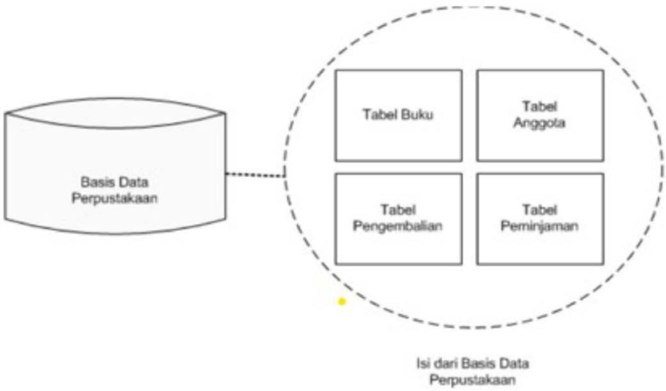
1.1 Definisi Basis Data

Basis data terdisi dari dua suku kata yaitu basis & data. Kata basis dapat bermakna markas, sebagai tempat berkumpul dan untuk data dapat bermakna pemaparan terhadap fakta yang terwakilkan terhadap suatu objek yang meliputi suatu peristiwa, keadaan dan lainnya yang dipaparkan kedalam format angka, simbol, huruf, gambar, atau sebagainya. Berasarkan pemaparan tersebut basis data (*database*) dapat diartikan sebagai kumpulan data yang memiliki hubungan serta tersimpan/terkumpulkan dalam memenuhi kebutuhan akan suatu informasi dalam perusahaan ataupun organisasi. Berikut merupakan pemaparan basis data menurut ahli diantaranya, (Sudarso, 2021), mengemukakakan basis data merupakan data yang dikumpulkan serta saling memiliki interaksi yang dibangun dengan maksud untuk memenuhi kebutuhan akan informasi dalam suatu perusahaan. Selain itu menurut (Alvin Dwi Hardiansyah & Dewi, 2020) *database* adalah desain dan integrasi yang dilakukan terhadap data sehingga kebutuhan pengguna dalam suatu organisasi/ perusahaan dapat terpenuhi. Menurut (Mulyana & Wahana, 2017) basis data atau *database* merupakan data logika yang dikumpulkan dan berelasi serta dirancang guna pemenuhan informasi yang diperlukan perusahaan. Berdasarkan pemaparan tersebut maka dapat ditarik kesimpulan bahwa basis data adalah sekumpulan data yang saling berinteraksi, didesain serta diintegrasikan dengan maksud/tujuan untuk pemenuhan kebutuhan informasi didalam perusahaan atau suatu organisasi.

1.2 Contoh Implementasi Basis Data

Basis Data (*Database*), sekarang ini mempunyai dampak besar pada berbagai sektor baik dari sektor perekonomian, sosial, sampai masyarakat. Sistem ini memiliki kaitan penting di dalam mengembangkan berbagai bidang rekayasa *software*, dan basis data menjadi komponen kerja yang menjadi dasar Sistem Informasi serta mendasari perubahan cara berbagai organisasi dalam bertindak.

Salah satu contoh penggunan basis data dalam suatu aplikasi/*software* seperti peminjaman suatu buku dalam perpustakaan. Ketika seseorang akan meminjam buku di perpustakaan, besar kemungkinan bahwa aka nada akses kedalam basis data. Admin akan meng-*entry* kode buku melalui sistem informasi perpustakaan yang terhubung dengan basis data untuk mengetahui data buku yang dicari. Sistem informasi perpustakaan selanjutnya akan mengurangi jumlah stok buku sesuai dengan yang di-input admin perpustakaan dan memunculkan jumlah stok yang tersedia. Seaindainya jumlah stok buku yang tersedia dibawah ambang batas, maka basis data secara langsung akan memberikan informasi kepada admin bahwa peminjaman buku yang diinputkan tidak bisa diproses. Atau, seandainya ketika seseorang bertanya ketersediaan sebuah buku, maka admin perpustakaan bisa mengecek ketersediaan buku yang diharapkan, jumlah stok dan lokasi buku disimpan dengan menjalankan Sistem Informasi perpustakaan yang terhubung dengan basis data buku perpustakaan.



Gambar 1. Contoh Siklus Basis Data Dalam Suatu Perpustakaan

1.3 Prinsip dan Tujuan Basis Data

Prinsip mendasar yang dimiliki basis data yaitu mempermudah dan mempercepat dalam mengambil, memproses, mengolah serta menyajikan data. Sedangkan tujuan utama basis data diantaranya :

1. Cepat dan mudah

   Dalam hal ini meliputi kecepatan didalam melakukan beberapa kegiatan seperti menambah, menyimpan atau mengedit data.

2. Space penyimpanan yang efisien

   Jumlah redudansi pada data dapat ditekan dengan cara mengimplementasikan kode-kode dengan menghasilkan relasi kedalam bentuk *file*.

3. Akurat

   Lebih ditekankan pada akurasi dalam proses *input* dan menyimpan data dengan mengimplementasikan tipe serta domain data

4. Ketersediaan

   Dalam hal ini menseleksi data utama *(master data)* maupun data yang kadaluarsa. Data yang sifatnya tidak lagi maupun jarang dipergunakan mampu dikontrol melalui sistem *database* aktif.

5. Kelengkapan

   Melakukan akomodasi kelengkapan data yang mengakibatkan perkembangan data melalui penambahan record baru serta perubahan struktur.

6. Aman

   Dapat melakukan penentuan hak akses terhadap *user* sehingga menjadi lebih aman.

7. Kebersamaan pengguna

   Basis data dapat dipergunakan bagi beberapa *user* dalam tempat yang berbeda-beda. Basis data yang dikendalikan *software*/aplikasi yang men-support berbagai *user* (*multi user*) mampu mendorong akan pemenuhan akan kebutuhan, namun harus menghindari data yang bersifat tidak konsisten.

1.4 **Jenis Basis Data**

Berdasarkan jenisnya, basis data dijabarkan kedalam 5 (lima) jenis, diantaranya :

1. Basis Data Operasional

Basis Data Operasional atau *Operational Database* dan sering disebut *database* OLTP (*Online Transaction Processing*) digunakan dalam memproses data dinamis dan bersifat *realtime.* Database ini memungkinkan *user* agar dapat melakukan modifikasi data yang meliputi *insert*, *edit*, *delete* terhadap data yang dilakukan secara langsung. Berikut merupakan contoh basis data operasional, yaitu :

a. JSON

JavaScript Object Notation atau disebut JSON adalah format file yng dalam proses mengirim data menggunakan teks. Format tersebut tergolong umum dipakai *user* dalam bertukar data seperti melakukan komunikasi cepat melalui *web browser*. Sinkron terhadap data dapat dilakukan *realtime*.

b. XML

Extensible Markup Language atau disebut XML merupakan bahasa program markup dimana terdapat *rule* dalam memberi 2 kode dokumen berbeda yang mampu terbaca bagi pengguna dan terbaca computer. Dengan XML, *output* data berupa *text* yang dapat dipakai dalam representasi struktur basis data.

Sinkron terhadap data dapat dilakukan *realtime*. XML memiliki kemiripan dengan JSON.

2. Basis Data Warehouse

Adalah sistem basis data yang umumnya digunakan dalam pembuatan laporan maupun analisa data. Basis data ini dianggap dalam bagian utama dalam intelijen bisnis. Basis Data *Warehouse* adalah *repository central data* terintegrasi oleh 1 atau lebih sumber. Data yang disimpan pada *warehouse* awalnya di-*upload* melalui *operation system*. Data tersebut dapat melalui *operational storage* dan memperkenankan untuk melakukan membersihkan data. Langkah itu dapat dikatakan dan diartikan sebagai operasi tambahan serta kualitas dari data sebelumnya dapat dipastikan serta dapat dijadikan pelaporan pada *warehouse*. Berikut merupakan contoh *database warehouse* yaitu :

a. Microsoft SQL Server

Merupakan sistem basis data yang diciptakan Microsoft. SQL Server adalah produk software yang memiliki fungsi dalam 'save and take' data sesuai permohonan *software* lain. Hal ini dirasa mampu dijalankan secara optimal melalui komputer yang sama/tidak menggunakan koneksi internet.

3. Database Terdistribusi

Merupakan basis data yang media penyimpanannya tidak dipasang di media komputer yang sama melainkan disimpan pada media komputer yang terdapat pada tempat sama ataupun letaknya tersebar melalui jaringan komputer yang sifatnya saling terhubung. Basis Data terdistribusi bukanlah sistem pararel yang mengkombinasikan dan bersistem data tunggal. Berikut merupakan contoh basis data terdistribusi, diantaranya :

a. Microsoft Access (Office)

Adalah sistem Basis Data Manajemen Sistem atau sering disingkat DBMS *(Database Management System)* yang menyimpan data menggunakan format yang dimiliki tersendiri. Melalui Microsoft Access, user dapat mengimport data yang telah disimpan pada basis data lainnya. Aplikasi ini sangatlah tepat diimplementasikan pada SI yang bersifat memerlukan basis data terdistribusi. Umumnya Microsoft Access dipergunakan oleh pebisnis mulai dari kecil maupun menengah dalam suatu organisasi kecil ataupun dipergunakan dalam perusahaan besar (Radiyyah et al., 2022).

4. Basis Data Relasional

Basis Data Relasional merupakan jenis *database* tergorganisir sesuai denan model relasi data. Sangat banyak aplikasi yang menerapkan basis data ini. Umumnya, sistem tersebut mempergunakan *Structured Query Language* (SQL) sebagai bahasa program dalam penggunaan *database* dan *query*. Berikut merupakan contoh dari Basis Data Relasional yaitu :

a. MySql

Merupakan jenis basis data yang popular serta umum dipergunakan serta dapat digolongkan kedalam basis data relasional (Maulana, 2016). Beberapa aplikasi yang menggunakan MySql diantaranya : Google, Youtube, Facebook WordPress, Joomla menggunakan MySql dalam memanajemen basis data.

b. PostgreSQL

PostgreSQL memiliki fungsi dalam penyimpanan data yang tergolong aman serta mampu mengembalikan data sebagai respon atas permintaan *software* lain. PostgreSQL ini umumnya digunakan pada sistem operasi macOS, dikarenakan pengaturan yang dimiliki sudah tersedia secara *default*. Basis data ini dapat dijadikan rekomendasi dan solusi bagi *user* basis data yang mendukung berbagai platform serta lisensi yang bebas (Munawaroh, 2005).

c. MongoDB

MongoDB merupakan basis data yang memiliki orientasi pada file *cross platform* dan *open sourc*e. Basis data ini memakai dokumen sejenis seperti skema JSON, karena itulah sistem MongoDB digolongkan kedalam program basis data NoSQL.

Masih banyak lagi jenis basis data relasional lainnya yang dapat digunakan seperti MariaDB, SapHANA, Oracle Database dan MemSQL.

5. End-User Database

Basis data yang digolongkan ke jenis ini adalah SQLite yang merupakan sistem manajemen *database* yang tertuang di *library* pemrograman C. SQLite terpasang pada program akhir sehingga tepat dipakai dalam men-support penyimpanan data akhir *end user*.

SQLite merupakan basis data favorit digunakan sebagai *software* basis data data menyimpan data lokal / klien melalui aplikasi seperti peramban web. SQLite sangat umum dipakai pada *operating system*, *website*, serta sistem *embedded* yang bersifat luas contohnya *handphone*.

BAB II

Lingkungan Basis Data

Pendahuluan

Lingkungan basis data merupakan sifat yang abstrak dalam sebuah basis data yang terdapat gambaran dari kebutuhan-kebutuhan informasi yang diperlukan. Tujuannya adalah menyediakan sudut pandangan abstrak sebuah data dengan dimanipulasikan dan disimpan untuk menyembunyikan detail informasinya. Basis data dipandang berbeda-beda akan data akan digunakan pengguna. Demi terpenuhi kebutuhan yang berbeda-beda ini, terdapat arsitektur basis data yang digunakan dan telah berkembang yaitu arsitektur ANSI-SPARC.

Arsitektur ANSI-SPARC (*American National Standards Institute - Standards Planning and Requirements Committee*) adalah standar abstrak untuk sistem manajemen basis data. Konsep gagasan data logis banyak dan luas, tetapi tidak ada *Data Base Management System* (DBMS) yang sepenuhnya didasari pada kemandirian independensi akses pengguna langsung untuk tingkat konseptual.

Terdapat 3 level arsitektur ANSI-SPARC yang bertujuan untuk perbedaan tingkat level dari *user* terhadap basis data, yaitu antara lain:

1. Tingkat Eksternal (*External Level*)

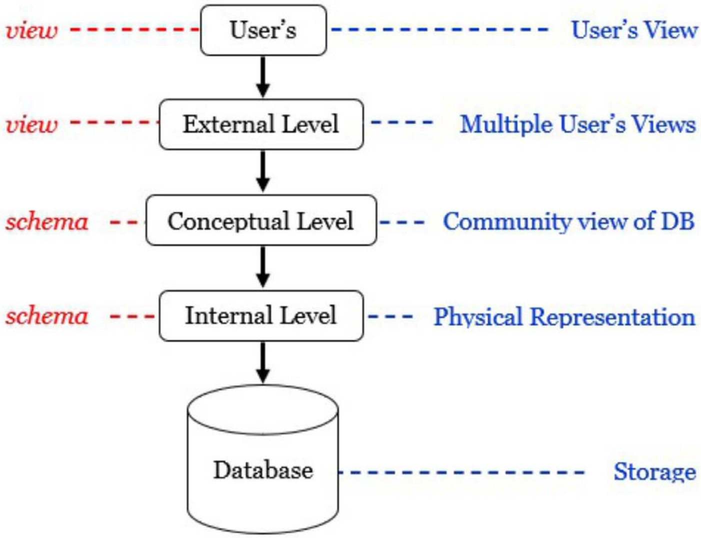
- **a.** Pandangan Pengguna (*User's View*)
- **b.** Pengumpulan Data (*Data Gathering*)

2. Tingkat Konseptual (*Conceptual Level*)

- **a.** Pandangan Entitas, Atribut & Relasi (*Entity, Attribute & Relationship*)
- **b.** Kendala dan Keamanan (*Constraints and Security*)
- **c.** *Entity Relationship Diagram* (ERD)
- **d.** Normalisasi (*Normalization*)

3. Tingkat Internal (*internal level*)

- **a.** Tampilan Presentasi (*Physical Presentation*)
- **b.** Penyimpanan, Indeks, Kompresi & Enkripsi (*Storage, Index, Compression & Encryption*)
- **c.** Organisasi File (*File Organization*)



Gambar : 3 Level Arsitektur *Database* (ANSI-SPARC)

2.1 Tingkatan Arsitektur ANSI-SPARC

Tingkatan Arsitektur ANSI-SPARC ini memisahkan DBMS ke dalam tiga tingkatan :

1. Tingkat Eksternal (*External Level*)

Dalam level ini digambarkan pada basis data yang relevan bagi seorang pengguna terhadap basis data dengan memiliki sejumlah pandangan yang berbeda-beda. Tiap pengguna melakukan representasi dalam bentuk yang sudah diketahuinya. Pola pandang eksternal ini hanya terbatas pada entitas, atribut dan relasi antar entitas yang diperlukan saja. Contohnya aktifitas dari kampus pada akses mata kuliah dan akses mahasiswa.

2. Tingkat Konseptual (*Conceptual Level*)

Dalam level ini digambarkan bagian pola terhadap sudut pandang data yang disimpan dan hubungan antara datanya. Tingkat ini tidak menentukan data disimpan secara fisik. Beberapa hal yang penting pada tingkatan ini sebagai berikut:

a. kumpulan semua entitas, atribut beserta relasinya

b. data yang telah disimpan dan keterkaitan hubungan antar data

c. proses yang dihadapai oleh DBA/ programer

d. menjelaskan struktur semua pengguna

e. keamanan informas & integritas informasi

Pada tingkat ini terdapat data yang dibutuhkan oleh pengguna yang harus mencakup data yang ada pada *database*. Data dari entitas hanya terdiri dari jenis data dan besar data atributnya tanpa memperdulikan besarnya penyimpanan dalam ukuran *byte*.

3. Tingkat Internal (*internal level*)

Pada tingkat ini menggambarkan bagian perwujudan basis data dalam komputer yang disimpan secara fisik. Tingkat internal melibatkan bagaimana basis data secara fisik diwakili pada sistem komputer untuk menjelaskan bagaimana data sebenarnya disimpan dalam basis data dan di perangkat keras komputer.

Berikut hal-hal yang perlu diperhatikan pada tingkat ini :

a. menggambarkan bagaimana data disimpan dalam basis data

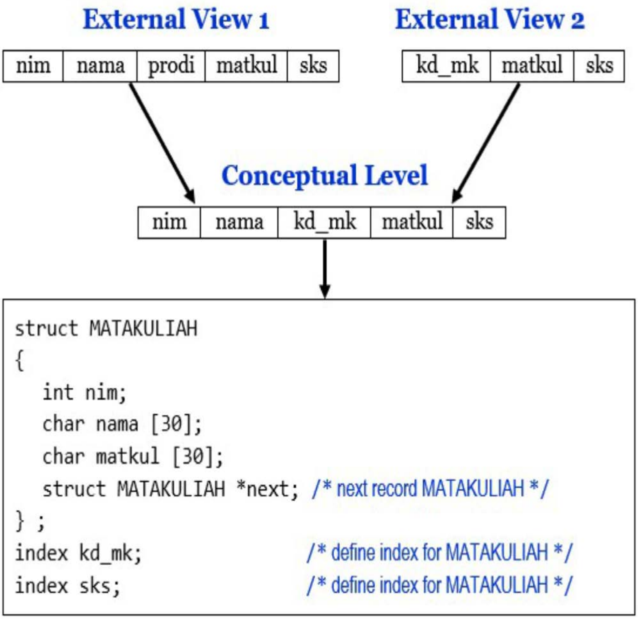
b. representasi indeks dan ruang penyimpanan data

c. *record* penyimpanan data elemen

Contoh pada tingkat ini seperti organisasi file secara sekuensial, relatif atau indeks sekuensial, penempatan *record* dan enkripsi data.

Tujuan arsitektur tiga tingkat adalah untuk memberi konsumen akses yang dipersonalisasi ke data yang sama. Pengguna tidak perlu memahami data yang disimpan secara fisik dalam *database* karena pemisahan antara level internal dan level ekternal. Selain itu, karena pemisahan level ini, *Administrator Database* (DBA) dapat memodifikasi struktur penyimpanan basis data tanpa memengaruhi tampilan pengguna.

Contoh perbedaan tiga tingkatan Arsitektur Basis Data (ANSI-SPARC) sebagai berikut :



Gambar : Perbedaan antara Tiga Tingkatan

2.2 Skema Basis Data

Ada tiga jenis skema berbeda yang sesuai dengan tiga level dalam arsitektur ANSI-SPARC:

1. Skema eksternal menjelaskan pandangan eksternal yang berbeda dari data dan mungkin ada banyak skema eksternal untuk database tertentu.
2. Skema konseptual menjelaskan item data terhadap hubungan diantaranya, bersama dengan batasan integritas (dikemudian

hari). Hanya ada satu skema konseptual per basis data.

3. Skema internal pada tingkat menjelaskan tentang definisi catatan yang disimpan, lalu dengan metode representasi, melalui bidang data, dan indeks datanya. Hanya ada satu skema internal per basis data.

2.3 Data Independence

Data Independence (data independensi) merupakan data perubahan pada tingkat rendah dan tidak mempengaruhi tingkat yang lebih tinggi. Ada 2 jenis data independensi, yaitu *Physical Data Independence* (PDI) dan *Logical Data Independence* (LDI).

1. PDI merupakan skema internal yang dapat dirubah DBA tanpa menggangu skema konseptual atau mengacu pada data statis pada skema konseptual terhadap perubahan skema internal. Contohnya:

    a. melakukan penambahan indeks

    b. merubah penyimpanan data

    c. merubah dari sekuensial ke indeks sekuensial pada organisasi file

    d. perubahan skema internal (mis. menggunakan organisasi file yang berbeda, struktur/perangkat penyimpanan)

    e. seharusnya tidak memerlukan perubahan pada skema konseptual atau eksternal

2. LDI merupakan skema konseptual dapat

dirubah oleh DBA tanpa menggangu skema eksternal atau mengacu pada kekebalan skema eksternal terhadap perubahan skema konseptual. Contohnya seperti :

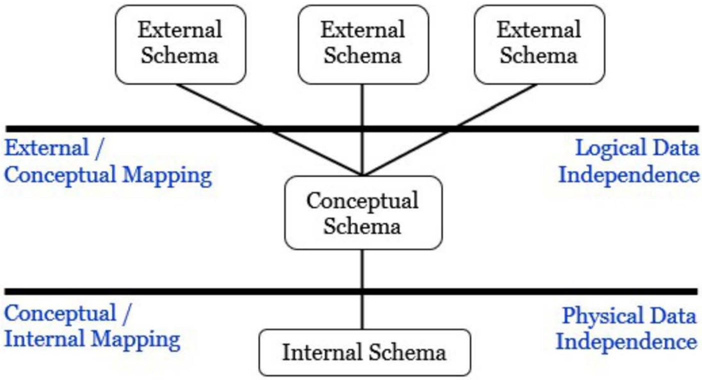
a. menghapus dan menambah suatu *type record* atau entitas

b. melakukan perubahan format data

c. seharusnya tidak memerlukan perubahan pada skema eksternal atau penulisan ulang program aplikasi.

Prinsip data independensi yaitu suatu hal yang harus diterapkan pada pengelolaan DBMS dengan pertimbangan:

1. DBA dapat melakukan perubahan isi dan tanpa mengganggu program yang sudah ada.
2. Aplikasi pengolahan suatu data dapat memberikan informasi tanpa mengganggu program yang sudah ada.
3. Migrasi basis data
4. Adanya *center controling* oleh DBA untuk keamanan data dengan menyesuaikan kebutuhan dari *user*.



Gambar : Data Independensi

Dari gambar diatas, dapat dipahami bahwa :

1. *Mapping* (Transformasi)

Proses pendefenisian informasi dari 1 level terhadap level lainnya.

2. *Internal Mapping / Conceptual*

Pendefinisian hubungan antara *conceptual view* dengan *database* di *internal level*. Bagaimana *record-record* atau *field-field* di dalam *conceptual level* didefinisikan di *internal level*.

3. *Conceptual Mapping / External*

Pendefinisian hubungan antara *conceptual view* dengan *external view*.

2.4 Database Languages

Database Language yakni suatu perantaraan oleh pengguna data untuk berinteraksi pada DBMS. Ada dua jenis bahasa yang digunakan untuk mengelola dan mengorganisasikan data yaitu *Data Definition Language (DDL) & Data Manipulation Language (DML)* (Dantes, Gede Rasben, dkk. 2019)

1. DDL

DDL adalah struktur basis data yang terdapat rangkaian desain basis data secara menyeluruh. Bahasa yang digunakan pada struktur basis data didalamnya terdapat kunci elemen, *record,* elemen data, dan relasinya. Dengan bahasa ini pengguna dapat membuat indeks, tabel baru, mengubah tabel, dan menentukan penyimpanan tabel. Hasil *compile* DDL yaitu kumpulan tabel yang tersimpan di dalam file khusus yang disebut *data dictionary*.

2. DML

DML adalah manipulasi terhadap data-data yang diperoleh dari basis data. Manipulasi data dapat berupa penambahan, pengurangan, pemetaan data di suatu *database*. Biasanya DBA melakukannya menggunakan *query* atau *coding*. *Query* adalah statement yang dirangkai untuk mengambil informasi.

Bagian dari DML ini berikut memiliki penanganan proses pengambilan informasi dari *query*. Terdapat 2 jenis DML, yakni sebagai berikut :

a. *Procedural DML*

Dilakukan untuk melakukan pendefenisian data yang diolah dan di perintah untuk dilaksanakan. Pengguna menentukan data apa yang diperlukan dan cara mendapatkan data tersebut (memungkinkan pengguna memberi tahu sistem dengan tepat cara memanipulasi data.) Contoh: Java

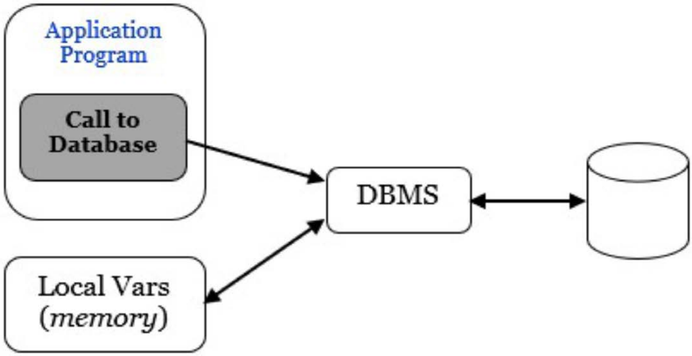
b. *Non Procedural*

Dilakukan untuk menerangkan dan menguraikan data yang ingin di ambil atau didapatkan. Pengguna menentukan data apa yang diperlukan tanpa menentukan cara mendapatkan data tersebut (memungkinkan pengguna untuk menyatakan data apa yang dibutuhkan daripada bagaimana data itu akan diambil.) Contoh: SQL

Setiap programmer memiliki bahasa pemrograman khusus seperti :

1. Program dengan terminal bahasa *Query* seperti MySQL yang metodenya dirancang **oleh programmer.**
2. DBMS memiliki fasilitas untuk menanamkan DDL & DML (sub-bahasa) dalam Bahasa Tingkat Tinggi (PHP, Python, Java, dll.

C, C++, Lips, …



Gambar : *Database Language*

2.5 Model Data

Model data merupakan kumpulan dari konsep untuk mendeskripsikan data lalu menghubungkan data, dan mengetahui kendala data dalam *database*. Untuk merepresentasikan data dengan cara yang dapat dimengerti. Model Data terdiri dari:

1. bagian struktural

2. bagian manipulatif

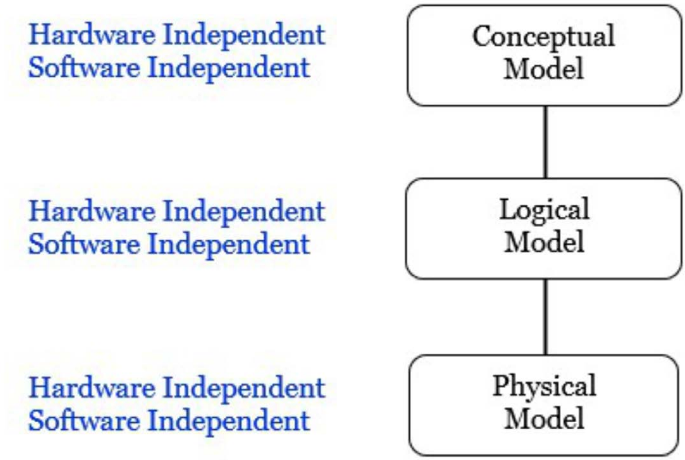
3. mungkin satu set aturan integritas

2.5.1 Kategori Model Data

Conceptual Data Models (Object-based) adalah konstruksi informasi perusahaan yang terlepas dari detail implementasi. Juga disebut model data berbasis entitas atau berbasis objek.

Logical Data Models (Record-based) adalah deskripsi logis dari informasi perusahaan dengan deskripsi implementasi tingkat tinggi. Juga disebut model data berbasis rekaman.

Physical Data Models adalah deskripsi fisik tentang bagaimana data disimpan di komputer.



Gambar : Kategori Model Data

Dari gambar diatas dapat di list kategori pemodelannya sebagai berikut :

a. Model Data Konseptual *(Object-based)* terdiri dari Entitas-Hubungan, Semantik, Fungsional, dan Berorientasi pada objek

b. Model Data Logis *(Record-based)* terdiri dari Data Relasional, Data Jaringan, dan Data Hirarkis

c. Model Data Fisik

2.5.2 Fungsi Model Data

Fungsi model data pada DBMS adalah sebagai berikut :

a. Data Storage, Retrieval, dan Update.

b. Katalog yang Dapat Diakses Pengguna.

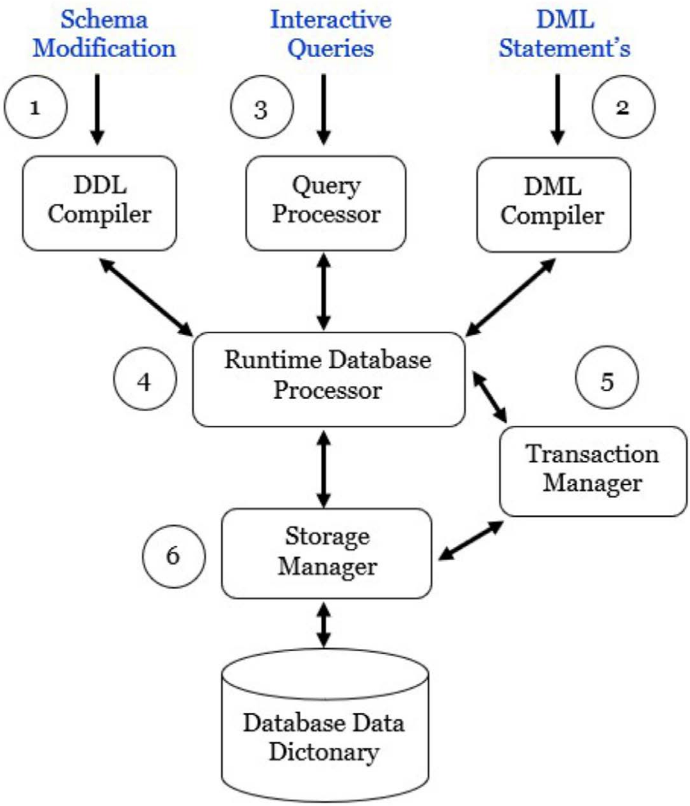
c. Dukungan Transaksi.

d. Layanan Kontrol Konkurensi.

e. Layanan Pemulihan.

2.6 Komponen Basis Data

Komponen basis data sangat berpengaruh pada sebuah informasi yang diharapkan oleh pengguna. Untuk itu komponen basis data dapat dilihat dibawah ini :



Gambar : Komponen Basis Data

Berikut keterangan dari gambar diatas : (Suryanto, 2015)

1. Skema modifikasi pada *DDL Compiler* menjadi kamus data *(data diction*ary).

2. *Statement* DML pada program aplikasi terdapat kode objek (cara melakukanya tergantung pada jenis dari *Aplication Programming Interface* (API). Saat *running* program, objek akan menghasilkan suatu panggilan untuk *Runtime Database Processor* (RDP).
3. *Query Interaktif* (adhoc) dioptimisasi untuk performa yang cepat. Dalam hal ini adalah tugas adhoc untuk optimalisasi di dalam basis data dimana hasilnya adalah suatu *query plan* yang saat dieksekusi mengakibatkan panggilan pada RDP.
4. RDP mengupdate urutan sebagai permintaan terhadap *storage manager* yang akan dilakukan oleh *transaction manager* dimana harus saling berhubungan dengan RDP untuk memonitor apa yang dibaca data dan yang di update terhadap permintaan dan kekuatan transaksi untuk menunggu jika terdapat konflik transaksi.
5. *Transaction Manager* berhubungan dengan log atau histori yang bertujuan untuk kontrol integrasi data, *backup data*, dan *recovery data*.
6. *Storage Manager* melakukan penerimaan permintaan dari RDP untuk halaman tertentu, harus memastikan apakah halaman yang diminta sudah berada dalam memori *buffer*. Jika berupa pesan, maka pesan tsb bisa dikirim ke halaman yg terisi dalam memori. Jika bukan, maka halaman diminta dari *file manager*.

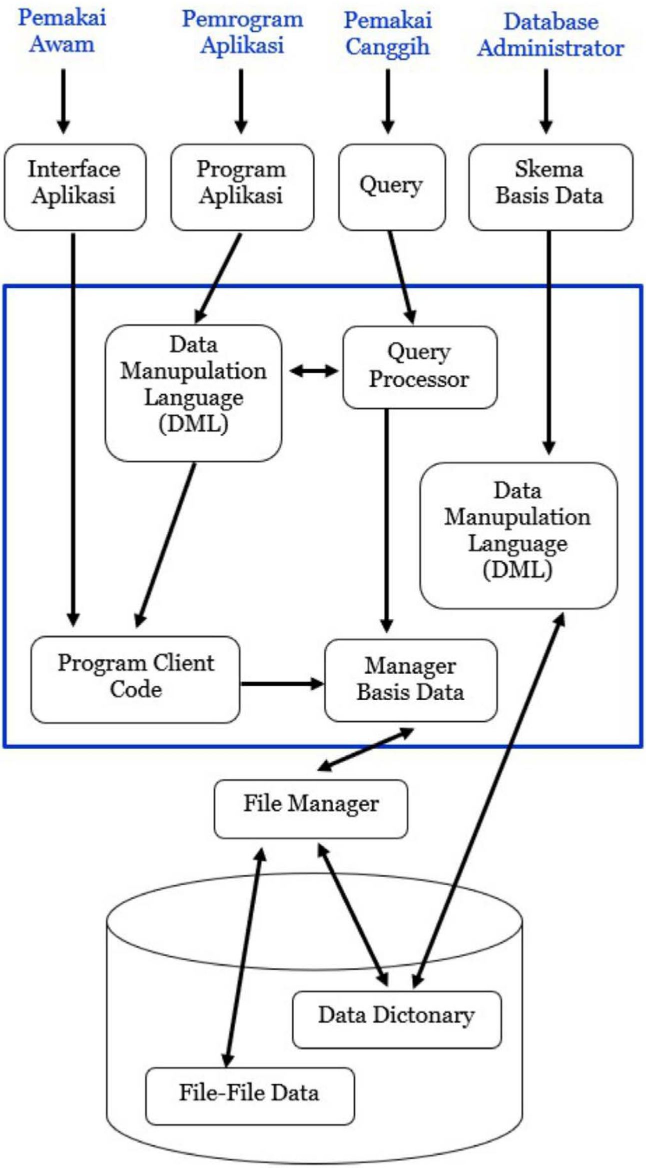
7. *Buffer Manager* menangani memori utama antara lain yaitu dengan menerima halaman data lalu mengalokasikannya pada data *buffer* ke dalam memori utama, lalu memutuskan untuk dikembalikan ke disk saat *memory buffers* penuh. Setelah itu memelihara tabel untuk memetakan *address* basis data pada memori.

2.7 Skema Blok Basis Data

Berikut skema blok basis data : (Suryanto, 2015)

:

1. *Data Dictionary* (DD) melakukan penyimpanan definisi di *database*.
2. Yang disimpan merupakan *field*, blok, struktur *data level record*, file-file dan tabel-tabel relasional.
3. Yang diolah merupakan informasi struktur data dan indek mengakses data secara cepat.



Gambar : Skema Blok Basis Data

2.8 Database Utility

Berikut utility-utiliti basis data :

1. *Loading*

   Proses file input ke *database*

2. *Backup*

   Melakukan penyalinan *database* jika terjadi kegagalan operasi data

3. *File Reorganization*

   Pengelompokan file *database* ke jenis *database* lainnya untuk meningkatkan performa informasinya

4. *Report Generation*

   Laporan untuk mengontrol *space, total, & summary.*

5. *Performance Monitoring*

   Memonitoring data dan membantu menampilkan statistik untuk DBA

BAB
III

Siklus Perancangan Basis Data

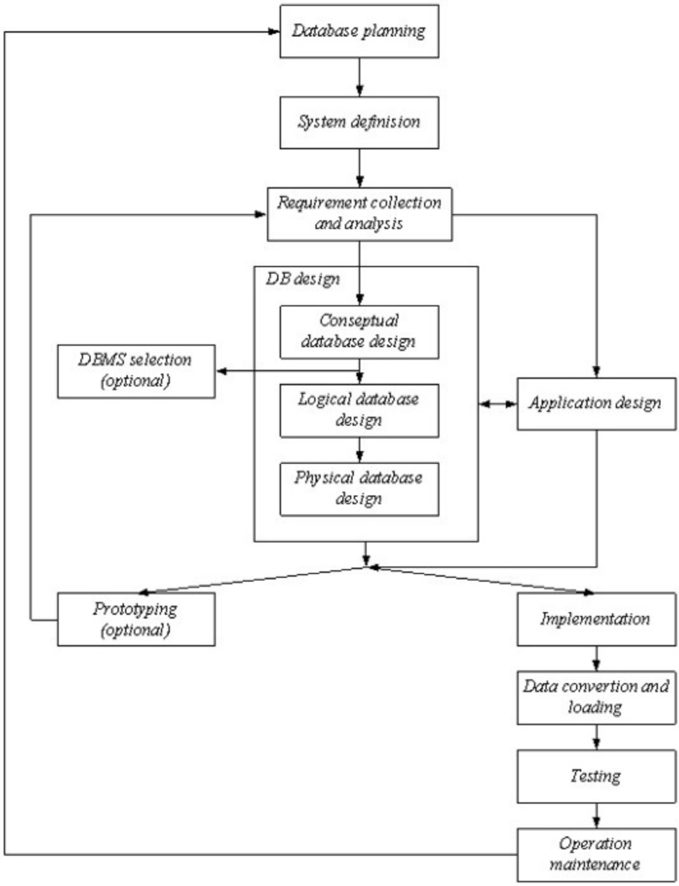
Pendahuluan

Data memiliki makna untaian fakta yang direkam dengan alat perekam dan disimpan ke memori komputer. Banyaknya data yang tersimpan dalam memori komputer tersusun secara *table-table* yang berelasi sering disebut dengan basis data. Objek basis data yang disimpan berupa suara, video, dokumen, foto dan file-file yang lainnya (Hoffer, 2002)

Setelah terbitnya suatu data yang telah dikumpulkan, maka muncul informasi yaitu data yang telah dikelola dalam suatu bentuk tertentu yang bermakna bagi penerimanya (Turban et al, 2005). Data dan informasi sangat penting dalam suatu perusahaan/organisasi karena hal tersebut dapat menjadi aset, tindakan, dan pengambilan keputusan untuk perusahaan/organisasi sehingga data dan informasi dapat membuat parameter kemajuan bagi perusahaan/organisasi.

Data dan informasi memiliki relasi yang keterkaitan sama halnya dengan basis data yang terdiri atas banyaknya data yang terkumpul dan memiliki relasi dengan yang lainnya. Basis data akan dibentuk dalam skema atau struktur tertentu yang akan disimpan dalan hardware komputer dan dapat dimanipulasikan dengan software komputer dengan tujuan untuk mendapatkan suatu informasi tertentu. Sebuah perusahaan/organsiasi membutukan sistem basis data dalam menjalankan sebuah organisasi tersebut dalam setiap bidang, contohnya bidang keuangan, bidang operasional, dan yang lainya.

3.1 Aktivitas Utama Siklus Hidup Perancangan Basis Data



Gambar 3.1 Siklus Hidup Perancangan Basis Data

Ringkasan Aktivitas Uatma

Berikut uraian ringakasan aktivitas utama yang ditunjukkan pada Tabel 3.1

Tabel 3.1 Ringkasan Aktivitas Utama

No.	Tahapan	Definisi
1.	*Database Planning*	Menentukan tujuan dari sistem basis data berypa gambaran secara luas seperti bagaimana pengumpulan data, design dan format data. Tahap ini memiliki dua metodologi yaitu *Mission Statement* dan *Mission Objectives*
2.	*System Definitions*	Mendeskripsikan ruang lingkungan dan batasan aplikasi.
3.	*Requirement Collection and Analysis*	Mengumpulkan dan menganalsis data yang akan dibutuhkan pada basis data
4.	*Databases Design*	Membuat rancangan *conceptual, physical* dan *logical*
5.	*DBMS Selection (Optional)*	Menentukan sistem DBMS yang lebih tepat digunakan
6.	*Application Design*	Menggambarkan rancangan antarmuka pengguna agar mudah dipahami.
7.	*Prototyping (Optional)*	Menggambarkan sebuah model basis data yang akan dibuat dengan tujuannya memperjelas dan mengevaluasi kelengkapan sistem basis data sebelum dijalankan.
8.	*Implementation*	Melakukan implementasi dengan menuliskan kode program berupa SQL ke dalam DBMS
9.	*Data Loading and Convertion*	Menginput data ke system dan pemindahan (migrasi) data ke sistem DBMS yang baru.

10.	*Testing*	Melakukan uji sistem secara menyeluruh.
11.	*Operational Maintenance*	Memonitor dan memelihara basis data selama fase operasi.

3.2 *Database Planning*

Database planning merupakan langkah awal dalam tahapan siklus hidup perancangan basis data. Tahap ini dilakukan untuk menentukan standar-standar yang akan dipakai dalam perancangan diantaranya bagaimana cara pengumpulan data, bagaimana standar bentuk data, dokumen apa saja yang diperlukan, dan bagaimana bentuk desain dan implemtasinya.

Untuk merancang suatu sistem basis data diperlukan strategi, berikut 3 pokok mengenai rancangan basis data yaitu:

1. Mengidentifikasi kebutuhan sistem, konsep rencana dan target tujuan sasaran pengembangan sistem
2. Menetapkan *Strong and Weekness* yang dimiliki dalam sistem informasi
3. Penafsiran teknologi informasi untuk memberikan keuntungan kompetitif.

Berdasarkan 3 hal pokok strategi tersebut dapat di atasi dengan metodologi. Tahap ini memiliki dua metodologi dengan tujuan untuk mengintegrasikan strategi sistem informasi. Berikut metodologi *database planning*:

1. Mission Statement

a. Menguraikan tujuan pokok kegunaan dari sistem basis data

b. Mendefinisikan perintah tugas dari sebuah projek

c. Menyediakan alur yang lebih jelas dengan tujuan terciptanya efektifitas dan efisisensi

2. Mission Objectives

Metodologi ini merupakan kelanjutan dari *mission statement*, dengan tujuan, yaitu:

a. Menguraikan tugas-tugas yang akan didukung sistem.

b. Menspesifikasikan pekerjaan yang dilakukan

3.3 System Definition

Definisi sistem menjelaskan bagian batasan, ruang lingkungan dan pandangan pengguna (*user viewer*) yang diterjemahkan secara perspektif sistem basis data, yaitu: yaitu:

- Mendefinisikan peraturan kerja yang khusus
- Mendefinisikan bagian-bagian aplikasi yang digunakan untuk kegiatan proses bisnis perusahaan/lembaga/individu
- Memastikan tidak ada user yang terlupakan untuk kebutuhan aplikasi
- Mengembangkan aplikasi yang kompleks dan kualitas.

3.4 Requirements Analysis and Collection

Bagian *requirement* ini adalah proses *collecting* data dan analisis data yang bersumber dari pengguna atau organisasi. Informasi yang dikumpulkan untuk identifikasi dan analisis kebutuhan pengguna terhadap sistem yang akan dikembangkan, diantaranya berupa:

1. Identifikasi data yang akan diolah
2. Rincian proses kelola data
3. Kebutuhan lainnya yang diperlukan sistem basis data

Penentuan analisa dari informasi yang telah dikumpulkan dapat dilakukan dengan tiga macam pendekatan berdasarkan *multiple user view*, yaitu: *centralized approach, view integration approach,* dan *combinantion of both approaches.* Perbedaan dari ketiga pendekatan ditunjukkan pada Tabel 3.2.

Tabel 3.2 Perbandingan pendekatan *centralized dan view integration*

No	*Centralized*	*View integration*
1.	Kebutuhan sudut pandang (*interface*) digabungkan jadi satu kumpulan.	Kebutuhan *interface* dipakai untuk rancang model data
2.	Model *Global data* merupakan penggabungan seluruh kebutuhan *interface* yang dapat direpresentasikan secara menyeluruh.	*Local data model* merupakan *user view* tunggal dan tersusun pada diagram-diagram sehingga dapat menggambarkan setiap *user view*.

Dalam tahapan ini dibutuhkan teknik yang tepat untuk mendapatkan semua informasi yang dibutuhkan. Pada umumnya teknik yang digunakan disebut *'fact finding technique'* diantaranya:

1. Memeriksa/mengevaluasi dokumen-dokumen
2. Wawancara
3. Jika ada wawancara diharapkan adanya kuesioner untuk memperdalam informasi
4. Terlibat dan mengamati jalannya kegiatan proses bisnis perusahaan
5. Penelitian melalui referensi luar perusahaan seperti di internet dan pengembang lain yang berpengalaman

Setiap informasi yang didapatkan pada tahapan ini akan digunakan untuk keperluan seluruh tahapan secara lengkap dari awal hingga akhir.

3.5 Database Design

Tahapan bagian rancangan basis data merupakan bagian proses desain yang sesuai dengan kebutuhan visi misi dan proses bisnis perusahaan. Sebuah sistem basis data yang baik adalah tidak adanya duplikat informasi atau disebut juga dengan redundant data.

Hal ini dapat memungkinkan kesalahan dan ketidakkonsistenan. Selain tidak adanya duplikat informasi, informasi yang juga harus benar dan lengkap. Hal ini dapat mengakibatkan setiap keputusan yang dibuat pada laporan salah karena laporan menarik informasi yang berasal dari sistem basis data.

Database design dirancang dengan tujuan menyediakan model dan mendukung bisnis proses yang ada pada perusahaan, merepresentasikan data dan relationshop antar data yang dibutuhkan pada perusahaan, dan menspesifikasikan desain untuk memenuhi kebutuhan performa perusahaan.

Menurut Fikry (2019), untuk membuat desain basis data yang dibaik dapat dilakukan pada beberapa pendekatan berikut, diantaranya:

1. Top Down

Desain ini menggunakan model *entitas relationship diagram* (ERD) yang dimulai dari identifikasi entitas, relasi entitas dan penentuan atribut entitas sebagai data pelengkap entitas tersebut. Pendekatan ini dimulai dari entitas *high-level* secara urutan yang bertujuan untuk identifikasi entitas lemah.

2. *Button Up*

Kebalikan dengan *top down*, pendekatan *Button Up* dimulai dari atribut dasar, kemudian digabungkan dan dianalisis. Selanjutnya, pembuatan kelompok relasi yang representasikan tipe entitas dan relasi.

3. *Inside Out*

Pendekatan ini mirip dengan pendekatan *button up*, hanya saja memiliki beda saat awal identifikasi entitas dan menyebar ke entitas lainnya serta atribut yang berelasi.

4. Mixed

Pada pendekatan ini merupakan gabungan dari pendekatan *buttom up* dan *top down*. Kegunaan dari model *mixed* adalah untuk membantu semantic data dan fasilitasi informasi yang dibutuhkan.

Ada tiga fase dalam mendesain basis data:

1. *Conceptual design*

Conceptual design digunakan pada perusahaan *enterprise* dimana keseluruhan aspek data akan dibangun dengan bersumber uraian spesifikasi kebebutuhan pengguna. Model ini digunakan untuk identifikasi entitas, atribut dan relasi serta direpresentasikan.

2. *Logical design*

Logical design merupakan kelanjutan dari *conceptual design* dimana proses pembentukan model dari informasi yang digunakan pada *enterprise* tanpa mempertimbangkan aspek fisik dari sistem basis data.

3. *Physical design*

Model ini menghasilkan deskripsi implementasi basis data pada tempat penyimpanan sekunder. Sasarannya adalah untuk membuat desain penyimpanan data yang dapat memberikan kinerja memadai dan memastikan integritas, keamanan, dan pemulihan data dapat bekerja dengan baik.

3.6 Database Management System (DBMS)

DBMS adalah perangkat lunak untuk mengelola atau manejemen basis data. Tujuannya adalah merancang, dan memungkinkan pembuatan, pembaharuan dan administrasi basis data. Tahapan ini dapat dilakukan kapan saja sebelum adanya *logical design* dimana informasi yang didapatkan cukup untuk kebutuhan sistem.

DBMS mempunyai banyak tipe dimana setiap tipe dapat disesuaikan dengan kebutuhan perusahaan. Berikut contoh DBMS dan model basis data yang ditunjukkan Pada Tabel 3.3

Tabel 3.3 Perangkat Lunak dan Model DBMS

No	**Perangkat Lunak DBMS**	**Model DBMS**
1.	Oracle	Relational DBMS
2.	MySQL	Relational DBMS
3.	MongoDB	Document store
4.	Cassandra	Wide coloumn store
5.	Redis	Key-Value Store
6.	SQLite	Relational DBMS

3.7 Application Design

Application design merupakan perancangan desain berupa antarmuka pengguna dan aplikasi yang siap intergasi dengan basis data. Desain basis data dan implementasi aplikasi memiliki sifat paralel dimana keduanya sama pentingnya. Menurut

Connoly (2002), rancangan antarmuka aplikasi terdiri dari 2 aspek yaitu:

a. Rancangan transaksi merupakan rancangan yang dilakukan pengguna tunggal atau perangkat aplikasi yang dapat menguraikan dan dokumentasikan data.

Rancangan transaksi terbagi atas tiga tipe, yaitu:

1. *Retrieval transaction*: transaksi untuk mengambil data yang akan ditampilkan untuk laporan

2. *Update transaction*: digunakan untuk input, *delete, edit* dan *update* data lama.

3. *Mixed transaction*: penggabungan antara *retrieval transaction* dan *update transaction*

b. *User Interface Design*

Desain antarmuka sangatlah penting dalam sebuah aplikasi dimana merupakan penghubung langsung antara aplikasi dan pengguna. Berikut aturan pokok pembuatan *user interface*:

a. Pemilihan dan penggunaan *form* yang jelas

b. Pemberian instruksi yang diberikan cukup jelas

c. Pengelompokan logika dan *field* yang terurut

d. Menampilkan *form* secara visual

e. Tabel *field* yang umum

f. Terminology dan singkatan harus konsisten

g. Konsisten dalam penggunaan warna

h. Ruang dan batasan yang terlihat untuk inputan data

j. Kursor yang nyaman

k. Operasi *field* mudah dijalankan oleh pengguna

l. Pesan kesalahan untuk data yang salah

m. Pilihan *optional* ditanjai jelas

n. Tanda kelengkapan dan penjelasan dari *field*

3.8 Prototyping (Optional)

Tahapan ini merupakan tahapan pilihan dimana membuat sebuah model kerja dari aplikasi basis data. Tujuannya untuk mempermudah identifikasi fitur-fitur dari sistem yang ada dan belum ada. Ada dua strategi prototyping, yaitu:

1. *Requirement prototyping*

Strategi ini menentukan seluruh kebutuhan sistem basis data yang diajukan, setelah kebutuhan terpenuhi maka tugas *prototyping* selesai.

2. *Evolutionary prototyping*

Strategi ini juga menentukan keseluruhan kebutuhan sistem basis data, namun setelah kebutuhan terpenuhi maka pengembangan lebih dalam dibandingkan sistem yang sudah jalan.

3.9 Implementation

Tahap implementasi merupakan tahapan lanjutan dari rancangan fisik basis data dan aplikasi. Implementasi dapat dilakukan dengan menggunakan:

1. *Data Definition Language* (DDL) dari DBMS yang telah dipilih.
2. Pembuatan *user view* dari DDL
3. Transaksi basis data diimplementasikan menggunakan *Data Manipulation Language* (DML) yang terdapat dalam bahasa pemrograman

3.10 Data Loading and Convertion

Pada tahap ini dilakukan ketika adanya pergantian sistem lama ke baru. Dalam pemindahan data tersebut terjadilah sebuah konversi aplikasi. Disini DBMS diperlukan untuk memanggil ulang file yang sudah ada kedalam sistem basis data yang baru.

3.11 Testing

Testing dibutukan untuk mengeksekusi program aplikasi yang bertujuan untuk mencari dan menemukan kesalahan pada sistem basis data. Testing biasanya memakai data real. Pada tahapan demontrasi sistem basis data dan program aplikasi dapat dijalankan.

3.12 Operational Maintenance

Operational maintenance merupakan tahapan terakhir dalam siklus perancangan basis data dimana proses pemantauan dan pengelolaan sistem yang akan diinstalasi. Setelah dilakukan instalasi, beberapa sistem basis data juga ada yang memerlukan pemeliharaan dan pembaharuan sistem basis data, dengan tujuan untuk pengawasan performa sistem ataupun nantinya akan terjadi penggabungan kebutuhan baru kedalam sistem basis data.

BAB IV

Database Manajement Sistem (DBMS)

Pendahuluan

Keputusan yang baik membutuhkan informasi yang baik yang berasal dari fakta mentah. Fakta mentah ini adalah dikenal sebagai data. Data cenderung dikelola paling efisien bila disimpan dalam sebuah basis data berbasis komputer yang dapat diorganisasikan melalui *software* atau perangkat untuk mendefinisikan, membuat dan memelihara data yang berada dalam *database* tersebut.

Dalam bab ini, Anda akan mempelajari apa itu *Database Management System* (DBMS), bagaimana sejarah awal cikal bakal hadinya DBMS, pengertian dari sebuah DBMS, fungsi, tujuan dan peran dari sebuah DMBS dalam membantu pengguna dalam melakukan aktivitas manajemen basis data secara terstruktur, komponen penyusun DBM hingga kelebihan dan kekurangan yang ada ketika DBMS diguankan dalam sebuah organisasi.

DBMS menjadi perangkat lunak yang menjadi sebuah perantara penting bari pengguna akhir dalam melakukan manajamen terhadap basis data yang mereka miliki. Mengapa dikatakan menjadi perantara bagi pengguna akhir, karena DBMS dibekali dengan bahasa pemrograman yang berguna dalam menelusuri sebuah basis data hingga melakukan aktivitas manipulasi data seperti memasukkan, memperbaharui, menghapus dan mengambil data pada sebuah *database.*

4.1 Sejarah *Database Management System*

Jauh sebelum dikenalnya sebuah sistem untuk melakukan manajemen *file* seperti yang ada pada saat ini, sistem berbasis *file* adalah hal yang umum dilakukan untuk dapat melakukan manajemen terhadap *file* yang dimiliki. Pendekatan dengan menggunakan basis data mulai disuarakan pada tahun 1960-an bertepatan setelah proyek pendaratan pesawat Apollo di bulan. Pada saat itu tidak ada sistem yang tersedia yang mampu untuk menangani dan mengelola sejumlah besar informasi yang akan dihasilkan dari proyek tersebut.

Permulaan tahun 1960 adalah waktu lahirnya cikal bakal pertama dari DBMS untuk generasi pertama yang prakarsai oleh Charles Bachman, DMBS generasi pertama tersebut disebut sebagai *Integrated Data Store* (IDS). Berangkat dari IDS tersebut terbentuklah semuah model data jaringan yang kemudian disepakati untuk distandarisasi oleh *Conference on Data System Language* (CODASYL).

Satu dekade berlalu, Edgar Cood mengajukan sebuah skema representasi data baru dengan sebuah model data relasional yang dikenal dengan istilah basis data relasional dan mengubah hampir sebagian besar pemahaman terhadap DBMS pada tahun 1980. Pada tahun yang sama bahasa *query* SQL (*Standard Query Language*) dikembangkan untuk mendukung basis data relasional dan distandarisasi oleh *American National Standards Institute* (ANSI) dan *International Standards Organization* (ISO).

Disepanjang tahun 1980 hingga memasuki permulaan tahun 1990 terjadi banyak perkembangan khususnya pada bidang basis data seperti peningkatan bahasa *query,* model data yang lengkap dan penyediaan fasilitas yang menunjang kebutuhan analisis data yang kompleks semua bagian organisasi.

4.2 Pengertian DBMS

Manajemen data yang efisien selayaknya membutuhkan adanya penggunaan *datasabe* dengan bantuan komputer. *Database* merupakan bentuk pengorganisasian dari sekumpulan data yang saling terhubung, terintegrasi oleh struktur komputer yang menyimpan kumpulan data yang terdiri dari data pengguna akhir dan metadata (informasi tentang data). Metadata berperan penting dalam memberikan data tentang data (data *about* data) dan serangkaian hubungan yang terdapat pada masing-masing tabel data yang ditemukan di dalam basis data. Singkatnya, metadata menyajikan gambaran yang lebih lengkap dari data dalam sebuah *database*.

Keberadaan *database* secara terkomputerisasi menjadi sebuah konsekuensi akan kebutuhan terhadap sebuah alat bantu yang dapat membantu pengguna dalam mengelola suatu basis data dan menjalankan berbagai macam operasi pengorganisasian terhadap data pada *database* yang kemudian dikenal sebuah perangkat lunak *Database Management System* (DBMS).

Untuk memperkuat pemahaman kita tentang apa aitu DBMS, penulis mengangkat beberapa definisi terkait DBMS yang dikemukakan oleh para ahli pada bidang basis data berikut ini:

a. Menurut (Considine dkk., 2012) DBMS merupakan sebuah struktur basis data terkomputerisasi yang dapat memungkin interaksi bersama dalam menangkap, menyimpan dan menghubungkan data dalam sebuah basis data.

b. Menurut (Elmasri & Navathe, 2016) DBMS merupakan sebuah sistem perangkat lunak yang bertujuan untuk memfasilitasi serangkaiaan aktivitas dalam mengorganisasi basidata yang meliputi proses mendefinisikan, membangun, memanipulasi, dan berbagi basis data di antara berbagai pengguna dan aplikasi.

c. Menurut (Hoffer dkk., 2016) Sistem manajemen basis data (DBMS) adalah sistem perangkat lunak yang memungkinkan pengguna dalam melakukan pendekatan terhadap basis data dengan menyediakan metode yang sistematis seperti membuat, memperbarui, menyimpan, dan mengambil data yang disimpan dalam *database*.

Berdasarkan beberapa definisi yangtelah dijelaskan sebelumnya dapat disimpulkan bahwa DBMS merupakan suatu perangkat lunak yang memiliki peran penting dalam memfasilitasi segala bentuk interaksi yang dilakukan oleh pengguna dalam mengakses basis data dalam rangka mengorganisasi data yang tersimpan dalam basis data tersebut.

4.3 Fungsi dan Tujuan DBMS

Dari definisi di atas terdapat lima fungsi utama yang dimiliki oleh DBMS yang dijabarkan sebagai berikut:

a. Mendefinisikan, membuat dan mengorganisasikan *database*, DBMS menetapkan hubungan logis antar elemen data yang berada pada *database* yang berbeda dan mendefinisikan skema struktur *database* dengan menggunakan DDL.

b. Memasukkan data, menjalankan fungsi untuk memasukkan data ke dalam *database* melalui perangkat lunak dengan bantuan pengguna.

c. Memproses data, menjalankan fungsi untuk memanipulasi dan memproses data dengan menggunakan DML.

d. Menjaga integritas dan keamanan data, memungkinkan akses *database* secara terbatas dari pengguna yang berwenang untuk menjaga integritas dan keamanan data.

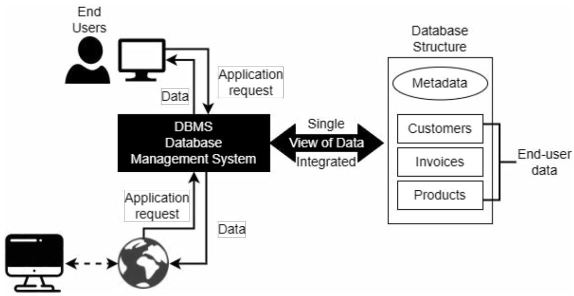
e. Database Query, memberikan informasi kepada pembuat keputusan yang mereka butuhkan dalam membuat keputusan penting dengan menjalankan serangkaian perintah menggunakan SQL.

Selain lima fungsi utama yang dimiliki oleh DMBS, bagi organisasi penerpan sebuah DBMS dalam sebuah *database* organisasi memiliki beberapa tujuan:

- **a.** Memberikan akses Bersama kedalam basis data
- **b.** Meningkatkan aksebilitas ke dalam basis data
- **c.** Menghemat ruang penyimpanan data
- **d.** Membantu menjaga keamanan data
- **e.** Menghilangkan resiko redudansi dan inskonsistentasi data
- **f.** Menangani data dalam jumlah yang besar

4.4 Peran DBMS

Dalam dunia nyata interaksi pengguna terhadap *database* yang mereka miliki akan selalu melibatkan dari keberadaan sebuah DBMS yang berfungsi sebagai fasilitator bagi pengguna terhadap basis data. (Coronel dkk., 2011). Pola interaksi pengguna dalam mengakses basis data melalui DBMS diilustrasikan pada Gambar 1 yang menggambarkan bagaimana DBMS menyajikan kepada pengguna akhir melalui program aplikasi dengan satu tampilan terintegrasi dari data dalam basis data. DBMS menerima semua permintaan aplikasi (*application request*) dan menerjemahkannya ke dalam operasi kompleks yang harus dipenuhi permintaan tersebut yang kemudian akan dikembalikan kembali kepada pengguna dalam bentuk data sesuai dengan permintaan pengguna. DBMS menyembunyikan banyak kompleksitas internal basis data dari program aplikasi dan pengguna.



Gambar 1. Interaksi Pengguna dengan DBMS dalam Mengakses *Database*

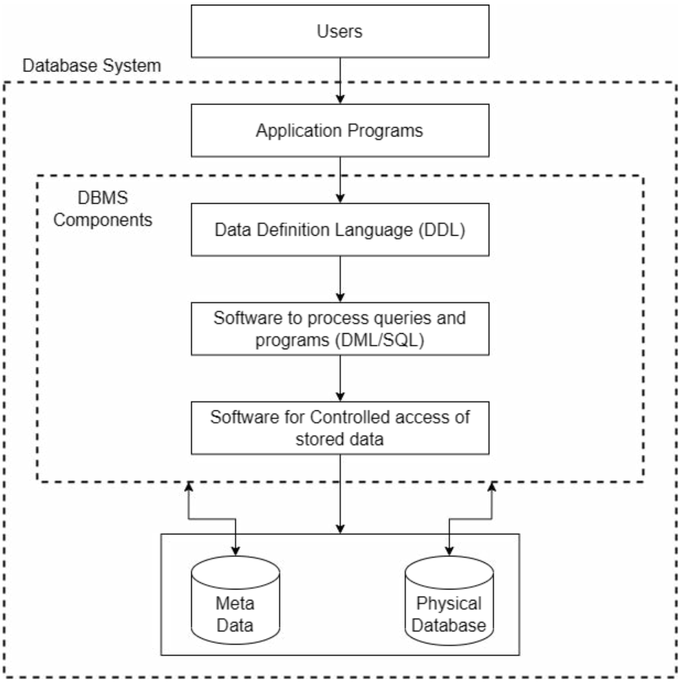
4.5 Komponen DBMS

DBMS merupakan sebuah perangkat lunak yang digunakan oleh para pengguna (*users*) untuk berinteraksi kedalam sebuah basis data (*database*) melalui sebuah program aplikasi sebagai *user interface* yang digunakan oleh pengguna. Secara umum DBMS menyediakan fasilitas seperti yang ilustrasikan pada Gambar 2 yang diikuti dengan definsi dari setiap komponen DBMS.

a. *Data Definition Language* (DDL) merupakan kumpulan perintah mendasar dalam dalam bahasa *Structured Query Language* (SQL) yang memungkinkan pengguna untuk membuat maupun memodifikasi tipe dan struktur data dari suatu objek berdasatkan batasan pada data yang akan di simpan dalam *database*.

b. *Data Manupulation Language* (DML) merupakan sederet perintah yang dapat digunakan oleh pengguna untuk melakukan manipulasi *database* seperti memasukkan, memperbaharui, menghapus dan mengambil data pada sebuah *database.*

c. *Structured Query Language* (SQL) merupakan bahasa pemrograman standar untuk menjalankan beragam perintah untuk menelusuri data pada *database*.



Gambar 2. Komponen DBMS

Perangkat lunak ini menyediakan fasilitas akses terkontrol dari *database* oleh pengguna, kontrol konkurensi untuk memungkinkan akses bersama dari *database* dan sistem kontrol pemulihan untuk memulihkan *database* jika terjadi kegagalan *hardware* atau *software*.

4.6 Kelebihan DBMS

Keberadaan DBMS yang menjadi jembatan bagi pengguna dalam mengelola *database* melalui sebuah aplikasi terintegrasi yang menampilkan banyak tampilan data pengguna yang bebeda bagi pengguna akhir dan *database* menawarkan kemudahan untuk berbagi data dalam basis kepada banyak pengguna lain. Mengingat keberadaan data yang merupakan bahan mentah yang sangat penting dari mana informasi diperoleh, Anda harus memiliki metode yang baik untuk mengelolanya data tersebut. Seperti yang akan Anda temukan dalam buku ini, DBMS membantu pengelolaan data menjadi lebih efisien dan efektif. Di dalam khususnya, DBMS memberikan keuntungan seperti:

a. Integrasi data yang baik: DBMS memastikan seluruh keberadaan data yang ada pada basis data terhindar dari adanya inskonsistensi, redudansi serta memiliki nilai akurasi yang tinggi yang dapat diakses oleh seluruh pengguna.

b. Keamanan data: DBMS memberikan jawaban akan isu keamanan data dengan hanya memberikan akses kepada pengguna yang berwenang dan diizinkan untuk mengakses *database* dan identitas mereka harus diautentikasi menggunakan nama pengguna dan kata sandi.

c. Akses data lebih cepat: DBMS membantu pengguna untuk menghasilkan jawaban dan permintaan masuk cepat melalui instruksi pada baris *query* yang diberikan pengguna atas pertanyaan yang membuat pengaksesan data menjadi akurat dan lebih cepat.

d. Dukungan pemulihan dan cadangan: DBMS menyediakan fasilitas dalam menangani proses pencadangan data secara berkala yang dapat berjalan secara terjadwal dan otomatis serta menangani skema mengembalikan *database* setelah kegagalan sistem atau *crash* untuk mengembalikan kepada kondisi sebelumnya.

e. Membantu pengambilan keputusan yang lebih baik: Konsistensi, akurasi dan validitas data menjadi jawaban yang diberikan oleh DBMS untuk mendukung dalam menghasilkan informasi dengan kualitas yang lebih baik, yang menjadi dasar pengambilan keputusan yang lebih baik.

f. Modifikasi yang mudah: DBMS menghadirkan sebuah sifat fleksibilitas dalam proses perubahan data pada sebuah sistem yang berjalan tanpa merubah atau merusak data yang sudah ada.

Disamping berbagai keuntungan yang akan diperoleh.

BAB
V

Entity Relationship Model

Pendahuluan

Basis data merupakan kumpulan catatan dari pengetahuan terstruktur yang didapatkan dari fakta (dunia nyata) nantinya akan disimpan dalam basis data, istilah ini pun disebut dengan nama skema atau model konseptual. Model kerangka konseptual bergantung dari apa yang menjadi kehendak keinginan pengguna yang mendasarkan spesifikasi kebutuhan yang dibutuhkan saat pengembangan sistem.

Entity Diagram Model digunakan untuk pemodelan data dalam bentuk objek-objek dunia nyata dimana data tersebut didapatkan dari kebutuhan pengembangan sistem secara spesifikasi (*specification requirement system*). Model ER ini direpresentasikan kepada pengguna sistem agar mudah dipahami dan diperbaiki apabila ada perubahan kebutuhan sistem.

Entitiy Diagram Model direpsentasikan dengan bentuk notasi diagram seperti bentuk persegi panjang, belah ketupat, *eclips*, garis lurus dan lain-lain. Dimana dikenal secara luas sebagai ERD

(*Entity Relationship Diagram*). Saat bangun ERD perlu elemen-elemen seperti entitas, relasi dan atribut.

5.1 Konsep Pemodelan Data

Kumpulan konsep yang menguraikan struktur basis data dan menggambarkan tingkatan abstraksi data disebut pemodelan data dan dapat juga sebagai model konseptual yang digunakan untuk deskripsi, relasi, semantik dan konstrain data. Seringnya, model data dirancang untuk dokumentasi dan memenuhi desain basis data (Fikry, 2019).

5.2 Tujuan dan Macam Model Data

Model data dirancang berdasarkan keinginan pengguna dan kebutuhan pengembangan aplikasi. Selain itu, digunakan untuk menyajikan rancangan data yang mudah di pahami oleh pengguna dan dapat di modifikasi apabila terjadinya perubahan kebutuhan pengembangan aplikasi. Adapun, model data memiliki 3 macam kelompok, diantaranya sebagai berikut (Bagui, 2022):

5.2.1 *Object Data Model*

Pemodelan data basis objek ini merupakan kumpulan data dan relasi/prosedur yang menerangkan hubungan logika-logika (*logic*) pada basis data didasarkan objeknya. *Object Data Model* terdiri dari (Baghui:

1. *Semantic Model*
2. *Binary Model*
3. *Entity Relationship Model*

5.2.2 *Record Data Model*

Model ini dirancang dasarkan atas rekaman yang menerangkan kepada pengguna mengenai relasi *logic* data dalam basis data yang terdiri dari 3 macam model, yaitu:

1. *Relational Model*
2. *Hierarchycal Model*
3. *Network Model*

5.2.3 *Physic Data Model*

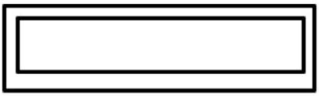
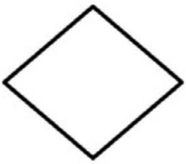
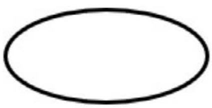
Model ini digunakan penguraian data tingkat internal atau menerangkan kepada pengguna bagaimana data disimpan ke penyimpanan basis data secara fisik. Jarang sekali model ini digunakan karena model yang sangat kompleks dan sulitkan pengguna. Model ini terdiri dari 2 macam model, yaitu :

- *Frame Memory*
- *Unifying Model*

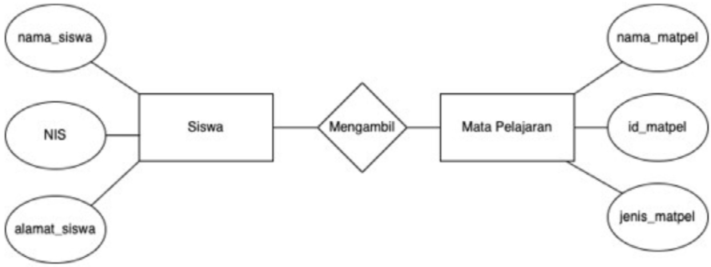
5.3 *Entity Relationship Model*

Pada umumnya, *entity relationship model* sebagai wujud dari model relasi dalam bentuk *entity relationship diagram* (ER-Diagram). ER diagram merupakan diagram yang menjelaskan relasi-relasi tabel data. Model ini bantu analisis sistem untuk uraikan apa yang dirancang dan jadi sebuah progress yang akan diimplementasikan ke dalam bahasa pemrograman. Berikut komponen atau notasi *entity relationship diagram* yang ditunjukkan pada Tabel 5.1 (Fikry, 2019).

Tabel 5.1 Notasi *Entity Relationship Diagram*

Simbol	Keterangan
	Entitas Sederhana / Kuat
	Entitas Lemah
	Relasi
	Atribut
	Penghubung

Contoh *Entity Relationship Diagram*:



Gambar 5.1. Contoh *Entity Relationship Diagram*

5.4 Notasi ER-Diagram

Sebuah relasi (hubungan) entitas atau ER Diagram memodelkan data sebagai entitas dan relasi. Entitas merupakan sesuatu yang akan disimpan dalam data, menurut Chen(1976) bahwa entitas dideskripsikan sesuatu yang dapat diidentifikasi, misalnya orang, tempat, objek, peristiwa dan sebagainya.

Entitas

Model entitas berlandasan persepsi dunia *real* (nyata) yang terdiri dari kumpulan objek yang disebut entitas (Bagui, 2022). Entitas digambarkan dalam bentuk persegi panjang yang tertera di Tabel 5.1. Entitas direpresentasikan dengan penjelas atau keterangan objek tersebut yang disebut dengan atribut.

Biasanya, entitas akan mewakili tipe atau kelas dari sesuatu dan diberi nama yang sesuai, contoh sebagai berikut:

a. Entitas orang : “Karyawan”, “Dokter”, “Pelajar” dan sebagainya.

b. Entitas tempat : “Bogor”, “Indonesia”, dan sebagainya.

c. Entitas objek : “Bangunan”, “Mobil”, “Motor” dan “Produk”

Contoh entitas :

Gambar 5.2. Contoh Entitas

Entitas mempunyai area cakupan yang besar atau luas sering dikenal dengan istilah *generalizations* (misalnya entitas Person), mungkin saja pembuatan entitas dipersempit lagi maknanya yang dikenal dengan istilah *specializations* (misal entitas Siswa). Namun, dalam praktik pembuatan entitas tidak ada grup *generalizations* atau *specializations*. Terdapat dua jenis entitas yaitu (Bagui, 2022):

a. entitas kuat (*entity strong*)

entitas kuat tidak memiliki ketergantungan terhadap entitas lainnya disimbol dengan persegi panjang.

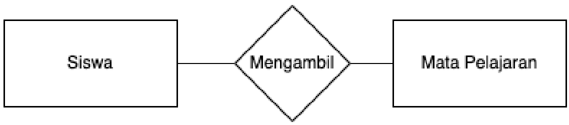
b. Entitas lemah (*entity weak*)

Entitas lemah yang keberadaannya bergantung terhadap keberadaan entitas lain dalam suatu relasi.

Relasi

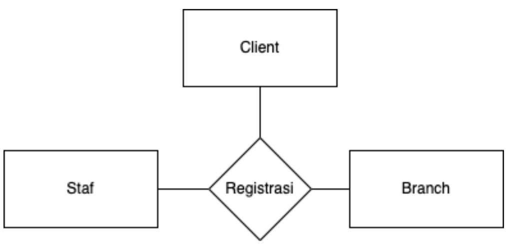
Relationship merupakan hubungan entitas yang berasal dari entitas yang lain dan berbeda. Disimbolkan dengan belah ketupat (Fikry, 2019). Derajat relasi merupakan jumlah entitas yang ada pada relasi, berikut 3 macam jenis derajat relasi (Bagui, 2022):

1. *Binary Relationship* merupakan dua entitas yang saling berelasi (hubungan).



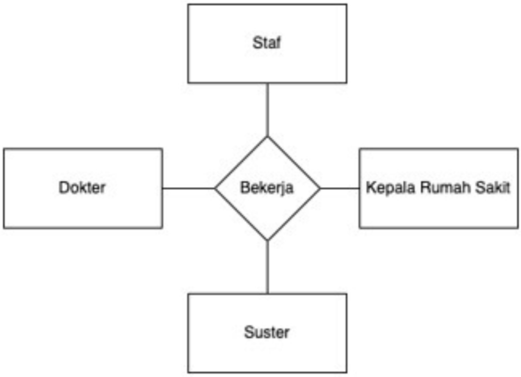
Gambar 5.3. *Binary Relationship Diagram*

2. *Ternary Relationship* merupakan tiga entitas yang saling berelasi (berhubungan)



Gambar 5.4. *Ternary Relationship Diagram*

3. *N-ary Relationship* merupakan banyaknya (dengan notasi n) entitas yang saling berelasi (hubungan).



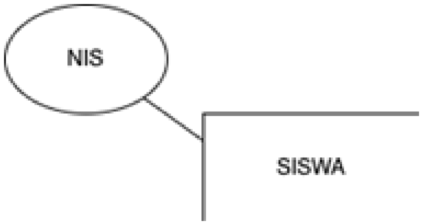
Gambar 5.5. *Ternary Relationship Diagram*

Atribut

Atribut adalah karakteristik entitas yang memberikan detail deskriptif tentang entitas. Terdapat beberapa jenis atribut yaitu sederhana atau atomik, komposit, multinilai, dan turunan. Disimbolkan dengan gambar elips.

Atribut Sederhana

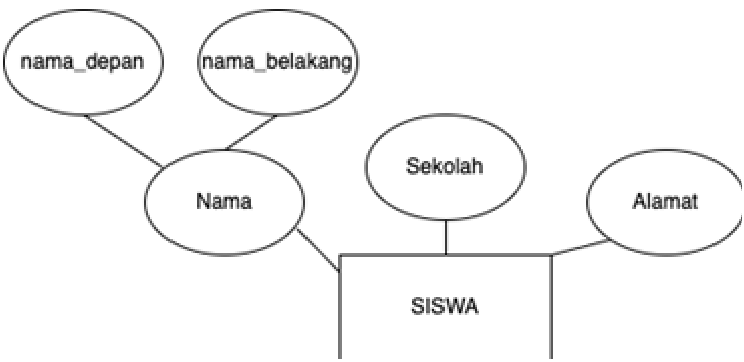
Atribut sederhana merupakan sebuah atribut yang tidak dapat dipecah (satu kesatuan yang utuh). Misalkan seorang Siswa memiliki Nomor Induk Siswa (NIS), dimana nomor tersebut memiliki keunikan sebagai identifikasi kesiswaan dan setiap siswa memiliki NIS yang berbeda-beda.



Gambar 5.6 Entitas Siswa memiliki Atribut Sederhana

Atribut Komposit

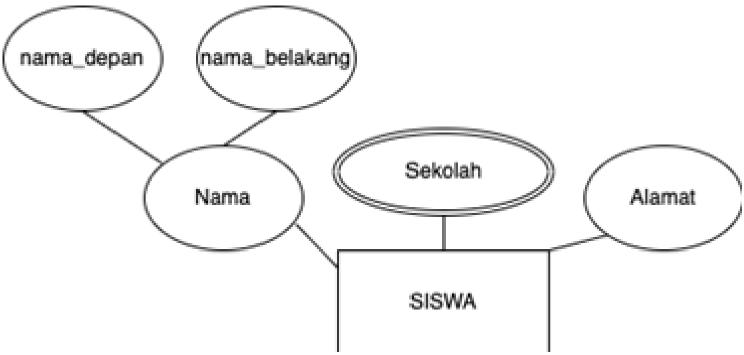
Atribut komposit yang sering dikenal sebagai *group attribute* yang merupakan menggabungkan atribut terkait, nama yang dipilih untuk digabungkan harus deskriptif dan umum dan jadi gambaran umum yang mudah dikenali serta pahami oleh pengguna. Mungkin saja suatu yang spesifik mengenai dari atribut berdasarkan keinginan dan kebutuhan pengguna. Misalkan, ada entitas Siswa yang memiliki atribut Nama, sebagaimana diketahui nama seseorang itu gabungan dari nama_depan, nama_tengah dan nama_belakang, yang menjadi sub atribut NAMA.



Gambar 5.7 Entitas Siswa dengan Atribut Komposit

Atribut multinilai

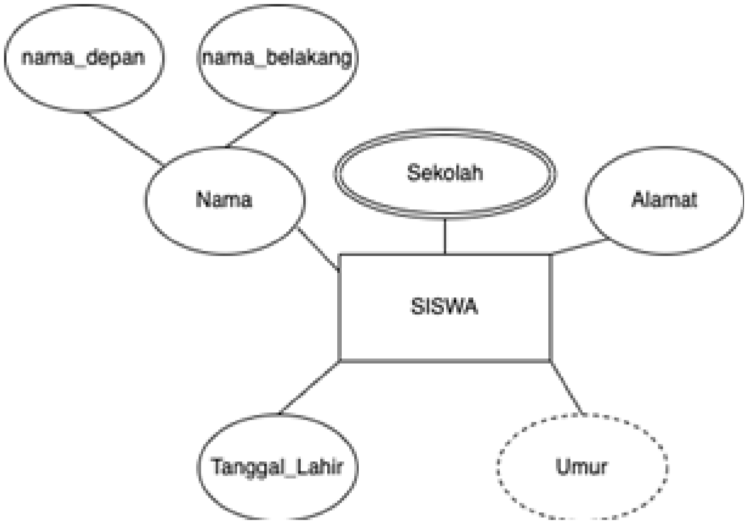
Tipe lain dari atribut non-sederhana yang akan dikelola disebut atribut multinilai. atribut multinilai, seperti namanya, dapat mengambil lebih dari satu nilai untuk kejadian entitas tertentu. Misalkan, atribut sekolah yang diketahui seseorang bersekolah dengan jenjang tingkat pendidikan dari sekolah dasar (SD), sekolah menengah pertama (SMP) dan sekolah menengah atas (SMA). Disimbolkan dengan oval berganda.



Gambar 5.8 Entitas Siswa dengan Atribut Multinilai

Atribut turunan

Atribut turunan adalah yang mungkin dibayangkan oleh pengguna tetapi mungkin tidak direkam, atribut turunan ini dapat dihitung dari data lain dalam database. contoh atribut turunan adalah usia, yang dapat dihitung setelah tanggal lahir disimpan. atribut turunan ditampilkan oval putus-putus.



Gambar 5.9 Entitas Siswa dengan Atribut Turunan.

5.5 Kardinalitas

Kardinalitas mewakilkan hubungan maksimal yang diperbolehkan antara himpunan atau kumpulan entitas terhadap entitas yang lainnya. Selain itu juga menunjukkan hubungan sederhana (minimum) yang diperbolehkan antara entitas dengan entitas lainnya. Notasi derajat relasi yaitu (*a,b*) dimana *a* akan mewakili derajat minimum dan *b* akan mewakili derajat maksimum (Soyusiawaty, et.all, 2020).

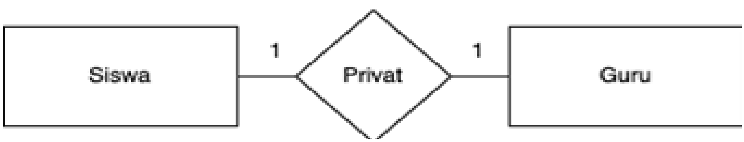
Gambar 5.10 Contoh Kardinalitas

Pada Gambar 5.10 dapat dijelaskan bahwa entitas dokter dalam hal ini dapat diwakilkan satu orang dokter memiliki relasi dengan kata kerja "periksa" dengan banyak pasien. Maka sangat penting sekali seorang analisis sistem secara detail menanyakan secara spesifik kebutuhan sistem yang akan dikembangkan karena ini akan mempengaruhi pada saat perancangan dan implementasi basis data.

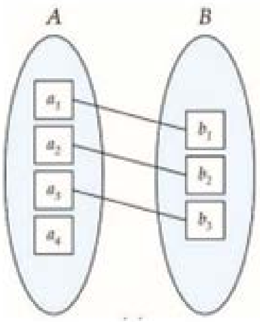
Model kardinalitas yang sering digunakan saat *relationship* yaitu

a. 1 : 1 (*One-To-One*) *Relationship*

Entitas A memiliki hubungan paling banyak hanya satu entitas B, begitu juga sebaliknya. Contoh relasi *one-to-one* adalah seorang siswa belajar privat dengan satu orang guru yang ditunjukkan pada Gambar 5.11. Apabila digambarkan ke dalam diagram himpunan A dan B ditunjukkan pada Gambar 5.12.



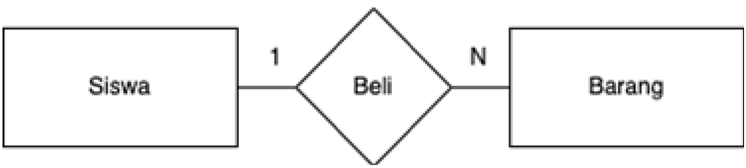
Gambar 5.11 *One-To-One Relationship*



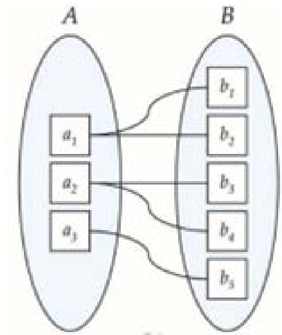
Gambar 5.12 Himpunan A ke B dengan *One-To-One*

b. 1 : N (*One-To-Many*) *Relationship*

Entitas A memiliki relasi banyaknya entitas B, dan tidak sebaliknya dimana entitas B hanya memiliki hubungan paling banyak satu entitas A. Contoh relasi *one-to-many* adalah seorang siswa membeli banyak barang jajanan diwarung yang ditunjukkan pada Gambar 5.13. Apabila digambarkan pada himpunan A dan B ditunjukkan pada Gambar 5.14.



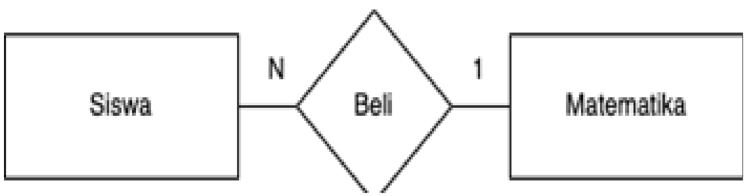
Gambar 5.13 *One-To-Many Relationship*



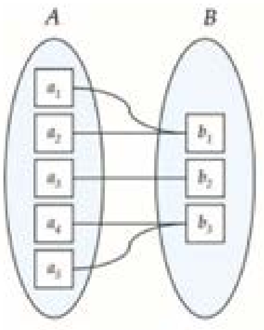
Gambar 5.14 Himpunan A dan B dengan *One-To-Many*

c. N : 1 (*Many-To-One*) *Relationship*

Entitas A memiliki hubungan paling banyak hanya satu ke entitas B, namun tidak sebaliknya, dimana entitas B dapat memiliki hubungan yang banyak ke entitas A. Contoh relasi *many-to-one* adalah banyak siswa kelas 6 belajar matematika dalam kelas yang ditunjukkan pada Gambar 5.15. Apabila digambarkan pada himpunan A dan B ditunjukkan pada Gambar 5.16.



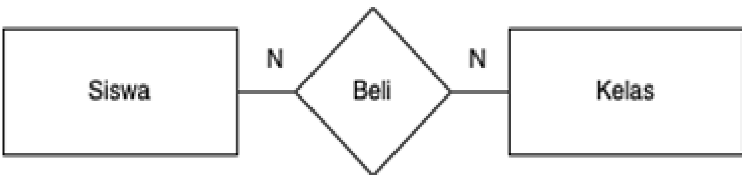
Gambar 5.15 *Many-To-One Relationship*



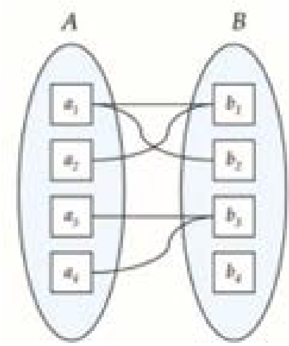
Gambar 5.16 Himpunan A dan B dengan *Many-To-One*

d. N : N (*Many-To-Many*) *Relationship*

Entitas A memiliki hubungan dengan banyaknya entitas B dan sebaliknya. Contoh relasi *many-to-many* adalah para siswa menempati kelasnya masing-masing yang ditunjukkan pada Gambar 5.17. Apabila digambarkan pada himpunan A dan B ditunjukkan pada Gambar 5.18.



Gambar 5.17 *Many-To-Many Relationship*



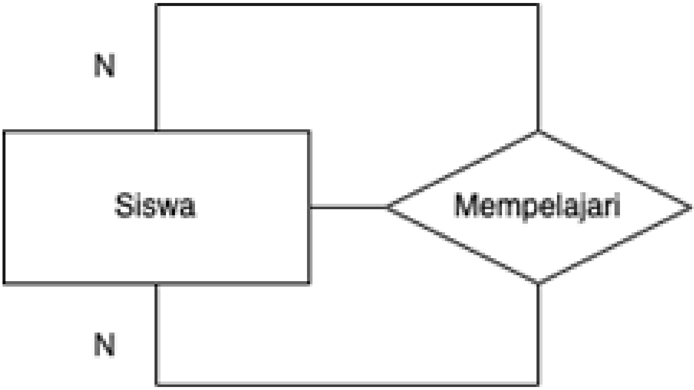
Gambar 5.18 Himpunan A dan B dengan *Many-To-Many*

5.6 Derajat Relasi

Satu atau lebih entitas yang saling berhimpun dan partisipasi dalam suatu hubungan (relasi). Misalkan: relasi Siswa belajar Mata pelajaran, relasi sebanyak 2 entitas yang terlibat yaitu entitas Siswa dan Mata Pelajaran. Pada umumnya, derajat relasi ERD dikelompokkan menjadi 3 bagian yaitu (Putri, 2022):

α. *Unary ERD*

Relasi ini hanya memiliki satu relasi saja (*unary relationship*). *Unary ERD* dikenal juga sebagai *recursive relationship*. Misalkan, Siswa berteman dengan siswa lainnya artinya satu siswa dapat berteman satu siswa atau lebih. Notasi *Unary ERD* ditunjukkan pada Gambar 5.19.



Gambar 5.19 Contoh *Unary ERD*

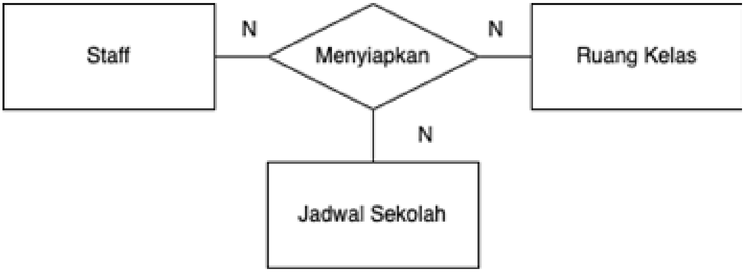
β. *Binary ERD*

Relasi ini hanya memiliki dua relasi saja (*binary relationship*). Misalkan, satu orang guru memberikan pelajaran tambahan ke satu siswa. Notasi *Binary ERD*.

Gambar 5.20 Contoh *Binary ERD*

χ. *Ternary ERD*

Relasi ini memiliki tiga atau lebih relasi (*ternary relationship*). Misalkan, staf sekolah mempersiapkan jadwal sekolah ke banyak kelas.



Gambar 5.21 Contoh *Ternary ERD*

5.7 Batasan Kardinalitas

Banyaknya instansiasi yang dilakukan dengan relasi entitas. Misalkan, entitas Guru dengan Siswa dengan relasi mengajar. Namun seorang Guru bisa saja memberikan pengajaran ke satu siswa saja atau banyak siswa dalam kelas. Hal ini perlu notasi yang tepat untuk gambarkan jangkauan relasi kardinalitas (Putri, 2022).

a. *Minimum Cardinality*

Jumlah yang sedikit relasi asosiasi antar entitas. Misalkan, Guru hanya memberikan pengajaran ke satu Siswa saja. Sehingga *Minimum Cardinality* dinotasikan bentuk bulat berongga, pada sisi Siswa hanya satu siswa yang diberikan pelajaran oleh tidaknya satu orang guru.

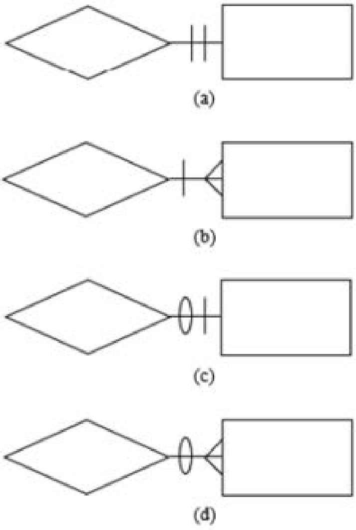
Gambar 5.22 Batasan *Minimum Cardinality*

b. *Maximum Cardinality*

Banyaknya relasi asosiasi antar entitas. Misalkan, Banyaknya Siswa mendapatkan banyak mata pelajaran pada satu semester.

Gambar 5.23 Batasan *Maximum Cardinality*

Berikut ini bentuk macam kardinalitas, yaitu:



Gambar 5.24 Relasi Kardinalitas : (a) *Mandatory One;* (b) *Mandatory Many* ; (c) *Optional One* ; (d) *Optional Many*

5.8 Rancangan *Entity Relationship Diagram*

Langkah teknis untuk membuat rancangan *Entity Relationship Diagram,* sebagai berikut (Bagui, 2022):

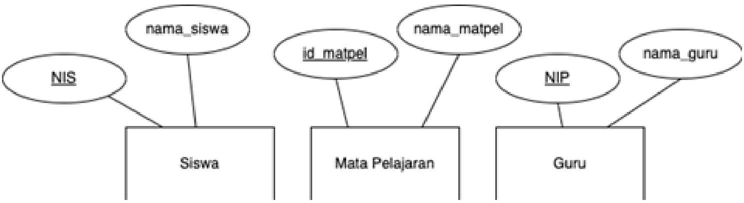
a. Identifikasi dan tentukan objek yang berdasarkan dari analisis kebutuhan sistem sebagai entitas.

   Misalkan: Budi sedang melakukan analisis kebutuhan dan perancangan sistem terkait pengadaan aplikasi Sistem Informasi Manajemen Sekolah (SIMS) dimana sistem dapat mengakomodasikan setiap proses kebutuhan manajemen sekolah seperti data nilai, mata pelajaran, siswa, guru dan sebagainya.

Gambar 5.25 Contoh Entitas SIMS

b. Tentukan atribut dan *key* dari objek (entitas) yang telah ditetapkan.

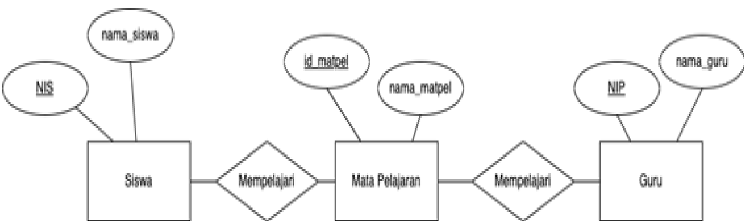
   Atribut sebagai data keterangan yang menjelaskan mengenai entitas tersebut. Misalkan: Himpunan entitas pada SIMS yang terlibat dalam pemodelan basis data yaitu entitas Siswa, Mata Pelajaran dan Guru. Sebagaimana diketahui bahwa entitas Siswa memiliki Nomor Induk Siswa (NIS) dan Nama Siswa. Maka, NIS menjadi *key identifier* yaitu hal yang menjadikan atribut tersebut memiliki keunikan dan atribut id_ matpel dan NIP (Nomor Induk Pegawai) juga sebagai *key identifier*.



Gambar 5.26 Contoh Atribut dari Entitas SIMS

c. Tetapkan dan lengkapi semua atribut yang terlibat pada entitas dan bangun relasi antar entitas.

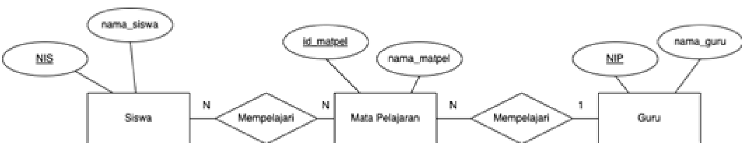
Data atribut entitas didapatkan dari catatan dokumentasi kebutuhan sistem, analisis sistem harus mencatat secara lengkap data pelengkap (atribut) dari entitas yang ditetapkan. Bangun relasi antar entitas menggunakan kalimat kerja.



Gambar 5.27 Contoh Relasi antar Entitas SIMS

d. Tentukan derajat relasi (kardinalitas) setiap relasi antar entitas.

penentuan derajat relasi dibangun dengan persepsi apa yang ada dalam dunia nyata (*real*). Selanjutnya, pemodelan ERD akan jadi bahan referensi dalam implementasi ke bahasa pemrograman. Misalkan : pada SIMS terdapat entitas Siswa, Mata Pelajaran dan Guru, proses keberlangsungan pada dunia nyata bahwa satu kelas terdiri banyaknya Siswa yang belajar banyaknya mata pelajaran dan banyaknya Guru dalam sekolah dimana 1 individu guru dapat mengajar lebih dari satu mata pelajaran atau hanya mengajar satu mata pelajaran saja. Selanjutnya, memutuskan bahwa relasi entitas Siswa dengan Mata Pelajaran dengan derajat kardinalitas *many-to-many* dan derajat relasi *many-to-many* atau *many-to-one* pada relasi entitas Mata Pelajaran dengan Guru.



Gambar 5.28 Contoh Relasi Derajat Kardinalitas

BAB VI
Relational Database Management System (RDBMS)

Pendahuluan

Sekitar tahun 1960-an, pengembang masih menggunakan perangkat lunak hardcode untuk mengelola data. Jika pengembang menggunakan tabel hash untuk menentukan struktur data, mereka harus menulis ulang sistem saat database berkembang, dan apabila data tidak sesuai dengan arsitektur standar yang sangat kaku, perangkat lunak harus didesain ulang setiap saat untuk jenis data yang disimpan dan jenis metode yang digunakan dalam manipulasinya.

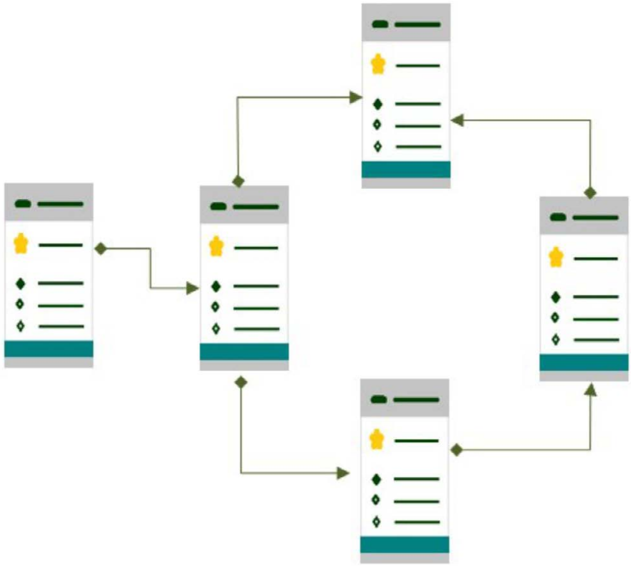
Seorang ilmuwan bernama Edgar F. Codd dari IBM mengakui perlunya arsitektur untuk memenuhi kebutuhan data yang terus berkembang. Pada tahun 1970, Edgar F. Codd dalam makalah penelitiannya dengan judul *"A Relational Model of Informasi for Large Shared Informasi Banks"*, berteori tentang model basis data relasional, makalah ini memungkinkan sistem itu sendiri menemukan representasi dan metode terbaik untuk menyimpan, mengambil, dan memanipulasi data, untuk memberikan solusi pada keterbatasan penggunaan DBMS.

Dalam model RDBMS yang diusulkan ini memungkinkan penggabungan data antar tabel, menyajikan informasi kepada pengguna yang dipresentasikan dalam bentuk tabel dimana tiap tabel terdiri dari sekumpulan baris serta kolom. Kemudian IBM mengembangkan System R, sebuah proyek penelitian pengembangan prototipe RDBMS untuk menguatkan teori relasional sebagai implementasi kekuatan industri yang menjadi tempat pengujian SQL, sehingga memungkinkan RDBMS dapat diadopsi secara lebih luas dalam waktu singkat. Edgar F. Codd merupakan penemu model RDBMS untuk memodelkan aspek realitas dalam mendukung proses yang membutuhkan informasi saat ini. Dimana saat Relational Database Management System (RDBMS) belum ada, industri masih menggunakan Sistem Manajemen Basis Data (DBMS) dengan struktur tabel hierarkis yang memungkinkan pengguna mengendalikan informasi dalam jumlah besar, tetapi memiliki berbagai keterbatasan dalam mengelola dan menghasilkan informasi diperlukan.

6.1. Konsep RDBMS

RDBMS adalah jenis basis data dengan objek pasif yang menyimpan informasi berbasis fakta, menyediakan akses ke titik data terkait, dan mengidentifikasi hubungan antara objek dan data yang disimpan di dalamnya. RDBMS didasarkan pada model relasional yang secara intuitif dapat merepresentasikan data dalam tabel, di mana setiap baris memiliki catatan dengan pengidentifikasi unik yang disebut kunci. Kolom di bagian tabel berisi atribut data, di mana setiap record biasanya memiliki nilai untuk setiap atribut agar lebih mudah menjalin hubungan antar data.

RDBMS memungkinkan pengguna untuk berinteraksi, membuat, memperbarui, mengelola, dan mengonfigurasi akses ke database menggunakan database relasional. RDBMS menyimpan data dalam format tabel, dan sebagian besar sistem manajemen database terkait bisnis saat ini menggunakan SQL (Structured Query Language) untuk mengakses database. RDBMS memungkinkan pengguna sistem untuk mendefinisikan, membuat, mengelola, dan mengontrol akses ke database yang dibutuhkan pengguna untuk mengakses sistem. Sistem RDBMS pertama merupakan implementasi dari model relasional, yaitu. kumpulan data dengan hubungan yang telah ditentukan, data tersebut disusun dalam bentuk tabel berupa kolom dan baris. Perkembangan teknologi informasi menjadi salah satu pendorong munculnya istilah database relasional berdasarkan model RDBMS relasional.



Gambar 6.1. Contoh Model Basis Data Relasional

6.2. Tabel RDBMS

RDBMS mengatur data ke dalam baris dan kolom yang bersama-sama membentuk tabel. Sebagian besar basis data yang digunakan secara luas saat ini didasarkan pada model RDBMS. Data biasanya terstruktur di beberapa tabel yang dapat ditautkan menggunakan primary key atau asing. Pengidentifikasi unik ini menjelaskan hubungan berbeda yang ada di antara tabel, hubungan ini biasanya diwakili oleh berbagai jenis model data.

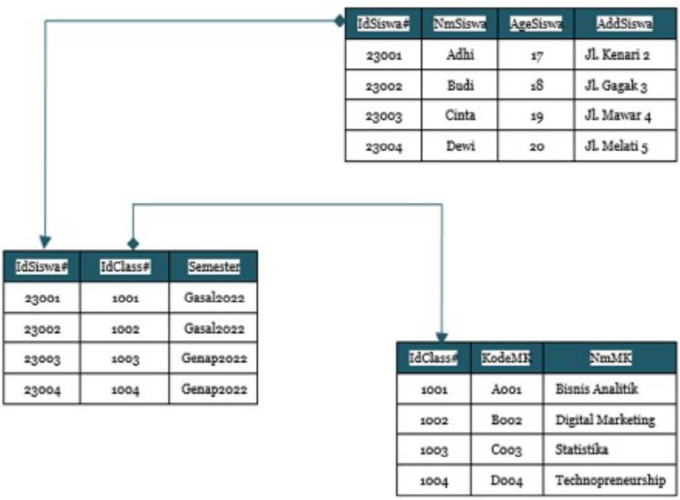
RDBMS menggunakan ekspresi aljabar relasional dasar ini untuk menentukan metode yang paling efisien. Itu matematika relasional. Komponen yang sangat kompleks dari sistem yang disebut pengoptimal queri.

Data membentuk hubungan dalam bentuk tupel, tabel data yang dikenal sebagai relasi, setiap sel memiliki tipe data skalar sederhana dan dihubungkan sebagai tupel yang membentuk relasi terpisah di setiap baris dan tiap entitas memiliki kode unik yang disebut primary key.

Tabel dapat berkomunikasi satu sama lain melalui kunci ini disebut foreign key, membentuk hubungan khusus dengan tujuh operasi dasar. Berikut ini operasi dasar dan sintaks SQL:

1. *Select*: digunakan untuk memilih dan mengembalikan relasi baru yang merupakan subset dari tupel yang memenuhi predikat bersyarat tertentu.
2. *Projection*; digunakan untuk mengembalikan relasi baru dengan tupel berisi atribut yang ditentukan, dan dapat mengatur ulang pengurutan atribut, serta dapat memanipulasi nilai.
3. *Join*; digunakan untuk mengembalikan relasi baru dari tupel yang ada di kedua relasi input, data dan atribut tidak perlu sama, selama ada satu kecocokan di dalam tupel.
4. *Union*: digunakan untuk mengembalikan semua tupel dalam dua relasi input sebagai satu relasi baru, berfungsi jika tipe dan atribut data input sama di keduanya

5. *Intersection*; digunakan untuk mengembalikan relasi tupel baru yang ada di kedua relasi input tetapi hanya berfungsi jika tipe dan atribut data input sama di keduanya.
6. Difference; digunakan untuk mengembalikan relasi baru dengan tupel yang ada di input pertama tetapi tidak di input kedua tetapi hanya berfungsi jika tipe dan atribut data input sama di keduanya.
7. *Split, Agregat, Sort*, dll dapat digunakan untuk mengevaluasi hubungan menggunakan teknik yang disebut aljabar relasional.



IdSiswa#	NmSiswa	AgeSiswa	AddSiswa
23001	Adhi	17	Jl. Kenari 2
23002	Budi	18	Jl. Gagak 3
23003	Cinta	19	Jl. Mawar 4
23004	Dewi	20	Jl. Melati 5

IdSiswa#	IdClass#	Semester
23001	1001	Gasal2022
23002	1002	Gasal2022
23003	1003	Genap2022
23004	1004	Genap2022

IdClass#	KodeMK	NmMK
1001	A001	Bisnis Analitik
1002	B002	Digital Marketing
1003	C003	Statistika
1004	D004	Technopreneurship

Gambar 6.2 Contoh Tabel Basis Data Relasional

6.3. Cara Kerja RDBMS

Data RDBMS tersimpan didalam bentuk table, jumlah tabel yang berbeda dan kunci utama yang unik dimiliki setiap table dalam sistem. Selanjutnya kunci utama melakukan identifikasi pada setiap table. Dalam tabel satu baris/lebih, disebut record atau unit horizontal, berisi informasi untuk record individual, sedangkan kolom disebut unit vertikal. Cara kerja RDBMS secara prinsipnya harus memeriksa batasan dalam membuat tabel, antara lain:

1. Integritas data (*data integrity*); langkah ini dilakukan karena integritas data harus diperiksa sebelum data dihasilkan.
   a. Integritas entitas memastikan bahwa baris dalam tabel tidak diduplikasi.
   b. Integritas domain mengisi tabel menurut kriteria tertentu.
   c. Integritas referensial menjamin pengguna tidak menghapus baris yang berkaitan dengan tabel lainnya.
   d. Integritas khusus digunakan untuk mengecek tabel telah memenuhi semua ketentuan khusus.

2. Check; langkah ini dilakukan untuk memastikan setiap entri dalam kolom/baris talah memenuhi kondisi yang benar dimana masing-masing kolom berisikan data unik.

3. Notnull; langkah ini dilakukan untuk memastikan bahwa nilai di setiap kolom tidak kosong/nol.

4. Kunci utama (*primary key*); memastikan sebuah tabel hanya memiliki satu kunci utama.
5. Kunci asing (*foreign key*); untuk menggabungkan tabel, foreign key disimpan dalam satu tabel mereferensikan primary key yang terkait dengan tabel lain.

6.4. Perbedaan DBMS Versus RDBMS

Tabel 6.1 Perbedaan DBMS versus RDBMS

Database Management System **(DBMS)**	*Relational Database Management System* **(RDBMS)**
1. Data disimpan dalam bentuk file 2. Elemen data perlu diakses secara individual 3. Tidak ada hubungan antar tabel data 4. Menyimpan data dalam bentuk hirarki 5. Sering terjadi redundansi data 6. Digunakan untuk organisasi untuk menangani data kecil 7. Mendukung pengguna tunggal 8. Pengambilan data lebih lambat untuk jumlah data yang besar 9. Tingkat keamanan yang rendah dalam manipulasi rendah	1. Data disimpan dalam bentuk-tabel 2. Elemen data dapat diakses secara bersamaan 3. Tabel data saling terkait satu sama lain. 4. Menggunakan struktur tabular kolom dan baris 5. Kunci dan indeks tidak mengizinkan redundansi data 6. Digunakan untuk menangani sejumlah besar data 7. Mendukung banyak pengguna 8. Pengambilan data lebih cepat 9. Terdapat tingkatan keamanan data dalam RDBMS

6.5. Software RDBMS

Berikut ini adalah software RDBMS yang sering digunakan:

1. Microsoft SQL Server
   a. Microsoft SQL Server mudah digunakan.
   b. Microsoft SQL Server dapat menyediakan fungsi serta memiliki beberapa fitur seperti recovery, security dan database

management.

c. Microsoft SQL mengunakan sistem operasi Windows, tidak dapat berjalan dengan sistem operasi lain.

2. MySQL
   a. Diperkenalkan pada tahun 1979 oleh Michael Widenius seorang ahli computer dari Swedia.
   b. MySQL bersifat open source, saat ini menjadi salah satu software basis data yang paling sering dipakai oleh pengguna.
3. IBM DB2
   a. Sistem operasi basis data yang menyediakan platform perangkat lunak basis data yang fleksibel dan hemat biaya.
   b. Sangat cocok digunakan untuk penggunaan aplikasi bisnis dengan skala besar dan dapat dioptimalkan untuk meningkatkan kinerja dan efisiensi bisnis.
4. SAP Adaptive Server Enterprise (ASE)
   a. SAP Sybase ASE, adalah sistem database berkinerja tinggi yang berfokus pada pengurangan biaya dan risiko operasional.
   b. Banyak digunakan di industri perbankan.
   c. SAP Sybase ASE dipasarkan sebagai RDBMS yang dirancang untuk aplikasi berbasis transaksi berkinerja tinggi yang berisi data volume besar dengan banyak pengguna secara bersamaan.

5. Oracle
   a. Salah satu basis data relasional yang paling canggih dengan fitur terlengkap.
   b. Software ini dapat membatu mengolah data secara cepat, akurat, memiliki fungsionalitas lengkap, namun dengan spesifikasi dan fitur yang tinggi berpengaruh pada spesifikasi hardware yang dibutuhkan.
   c. Database Oracle cocok digunakan untuk instansi, organisasi atau perusahaan berskala besar dengan beberapa perusahaan, serta server dan database yang sangat besar dan besar.

BAB VII

Normalisasi

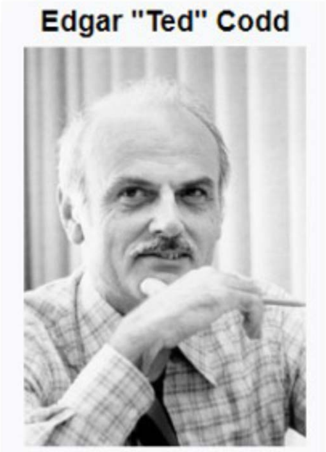
Pendahuluan

Salah satu tahapan pengembangan basis data adalah perancangan basis data, yang terdiri dari perancangan konseptual, lojikal, dan fisik. Pada perancangan secara konseptual digunakan tool pemodelan basis data seperti diagram ER. Perancangan lojik akan menghasilkan skema relasi yang merupakan hasil pemetaan dari diagram ER, dan rancangan fisik berupa struktur data yang akan akan diimplementasikan ke dalam basis data menggunakan DBMS.

Rancangan basis data yang baik adalah rancangan yang tidak akan menghasilkan **redundansi data** dan **inkonsistensi data**. Redundansi data terjadi jika terdapat beberapa kemunculan data yang sama di berbagai tabel. Contoh : pada tabel Pegawai terdapat data Nama, Alamat dan No_Telepon. Di tabel Anggota_Koperasi terdapat data Nama, Alamat dan No_Telepon. Terdapat redundansi dimana data Alamat dan No_Telepon ada di dua tabel berbeda. Hal ini dapat berpeluang menghasilkan data yang tidak konsisten dimana jika update data pada satu tabel tidak diikuti dengan update data yang sama di tabel yang berbeda. Redundansi dapat mengakibatkan penggunaan media penyimpanan data tidak efisien dan inkonsistensi memerlukan usaha ekstra dalam pengembangan aplikasi basis data untuk menjaga basis data selalu valid dan akurat.

Pada tahun 1970 , dua ilmuwan berkebangsaan Inggris yaitu Raymod F. Boyce dan Edgar Frank Codd menyampaikan ide untuk membuat semacam mekanisme dan ketentuan yang mengevaluasi dan memperbaiki struk-tur skema relasi yang terdapat pada sebuah rancangan basis data, agar skema-skema relasi tersebut tidak memiliki unsur redundansi dan tidak berpotensi menjadi tidak konsisten pada saat implementasi basis data tersebut, yang disebut sebagai **Normalisasi Basis Data**. Mereka membuat ketentuan berdasarkan status dan persyaratan yang harus dipenuhi oleh skema-skema relasi secara bertahap menurut tingkatannya (mulai dari 1NF, kemudian 2NF, lalu 3NF, dan seterusnya).

Sebelum membahas secara rinci tentang tahapan normalisasi ini, perlu dipahami terlebih dahulu tentang konsep ketergantungan fungsional antar atribut pada sebuah skema relasi berikut.

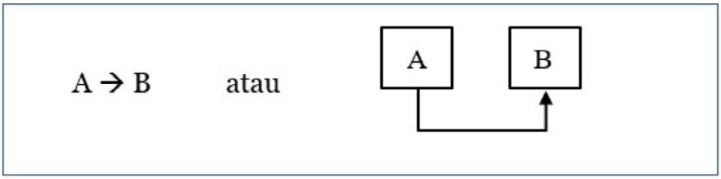


Sumber : Wikipedia

Gambar 1 Foto Edgar Frank “Ted” Codd

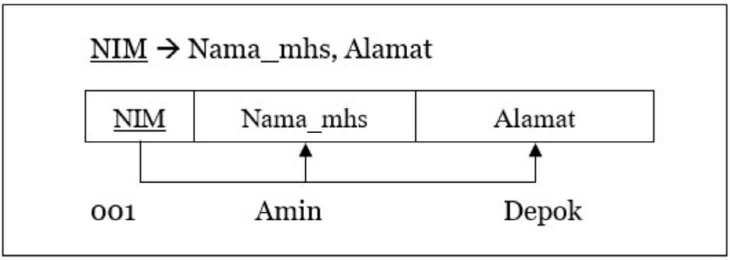
7.1 Ketergantungan Fungsional (*functional depen-dency*)

Pada sebuah skema relasi, ada bentuk ketergan-tungan antar atribut yang ada. Satu (atau lebih) atribut dapat tergantung secara fungsional pada satu (atau lebih) atribut lainnya. Himpunan atribut B dikatakan tergantung secara fungional pada himpunan atribut A dalam sebuah tabel, jika pada saat kemunculan nilai atribut AX1, AX2, dan seterusnya pada himpunan atribut A akan disertai dengan kemunculan nilai atribut BY1,BY2, dan seterusnya pada himpunan atribut B. Berikut adalah notasinya :



Gambar 2. Notasi ketergantungan fungsional

Contoh :



Gambar 3. Contoh ketergantungan fungsional

Pada bentuk ketergantungan di atas, keberadaan nilai Nama_mhs dan Alamat tergantung pada NIM. Artinya, jika pada sebuah tuple/record terdapat nilai NIM adalah "001" dan Nama adalah "Amin" dengan alamat "Depok", maka jika pada record lainnya terdapat NIM = "001", nilai atribut Nama dan Alamat haruslah "Amin" dan "Depok", bukan nilai lainnya.

Jika ada nama Amin dan alamat Depok di sebuah record sementara NIM bukan "001", berarti mahasiswa ini adalah mahasiswa yang berbeda dengan Amin dengan NIM = "001".

7.1.1 Ketergantungan fungsional penuh dan parsial

Ketergantungan fungsional dapat bersifat penuh atau parsial. Perhatikan bentuk ketergantungan fungsional berikut :

NIM → Nama, Alamat, Tgl_lahir, Hobby, Kelas

Atribut-atribut Nama, Alamat, Tgl_lahir, Hobby dan Kelas dinyatakan tergantung penuh terhadap NIM

Sedangkan pada bentuk ketergantungan dibawah ini, dimana No_Trans dan Kd_Brg menjadi primary key dan sisanya adalah atribut non key :

Gambar 4. Contoh ketergantungan parsial dan penuh

terdapat 3 buah bentuk ketergantungan fungsional, yaitu :

1. Fd1 : No_Trans → Tgl_Trans
2. Fd2 : Kd_Brg → Nama_brg
3. Fd3 : No_Trans,Kd_Brg → Total_Hrg

Pada **fd1** (*functional dependency* 1), Tgl_Trans hanya ter-gantung pada No_Trans.

Pada **fd2**, Nama_brg hanya tergantung pada Kd_Brg, apapun no transaksinya.

Sedangkan pada **fd3**, Total_hrg ditentukan oleh No_Trans dan Kd_Brg.

Kita dapat melihat bahwa di **fd1** dan **fd2**, atribut non-key tergantung hanya pada **salah satu** (sebagian) atribut key nya, dan ini disebut sebagai ketergantungan **parsial**. Sedangkan pada **fd3**, atribut-atribut non-key tergantung pada **semua** atribut key nya, sehingga disebut sebagai ketergantungan **PENUH**.

Sifat ketergantungan penuh atau parsial ini akan sangat menentukan pada saat dilakukan normalisasi nanti.

7.1.2 Ketergantungan transitif

Atribut-atribut non key dapat saja tergantung pada atribut non-key lainnya, bukan pada atribut key atau primary keynya. Perhatikan contoh berikut :

Pada skema relasi :

Karyawan(NIP, Nama, Alamat, Kode_Bag, Nama_Bag)

Primary key adalah NIP, dan atribut yang lainnya se-harusnya tergantung pada Primary Key nya (NIP). Tetapi ternyata Nama_Bag hanya tergantung pada Kode_Bag, siapapun yang menjadi karyawan. Berikut contohnya :

Tabel 1. Karyawan

NIP	Nama	Alamat	Kode_Bag	Nama_Bag
001	Andi	Bekasi	A1	Keuangan
002	Umar	Bekasi	A2	SDM
003	Hendra	Jakarta	A1	Keuangan

Sehingga bentuk ketergantungan fungsionalnya menjadi :

Fd1 : NIP → Nama, Alamat, Kode_Bag

Fd2 : Kode_Bag → Nama_Bag

Karena Nama_Bag tidak tergantung langsung pada NIP sebagai primary key, maka Nama_Bag tergantung secara **transitif** pada NIP melalui Kode_Bag.

7.2 Tahapan Normalisasi

Edgar Codd melihat bahwa seharusnya rancangan tabel pada sebuah basis data dapat dijaga konsistensinya dan menghindari kerangkapan data atau redundansi data. Oleh sebab itu, beliau menyarankan diterapkannya proses Normalisasi yang bertujuan agar rancangan skema relasi/tabel mengikuti aturan-aturan tentang bentuk-bentuk normal yang terdiri mulai dari bentuk normal pertama hingga bentuk normal ke lima, yang dikenal sebagai :

1st Normal Form (1NF)

2nd Normal Form (2NF)

3rd Normal Form (3NF)

wBoyce-Codd Normal Form (BCNF)

4th Normal Form (4NF)

5th Normal Form (5NF)

Di mana semakin tinggi tingkatannya, semakin ketat pula aturan bentuk normal nya.

Dalam prakteknya, kegiatan normalisasi hanya dilakukan pada bentuk 1NF, 2NF, dan 3 NF saja. Hal ini disebabkan karena pada setiap level normalisasi, skema relasi yang ada berpeluang untuk dipecah-pecah menjadi beberapa skema yang lebih kecil. Seperti pada contoh di Gambar 4, sebelumnya ada 1 (satu) skema relasi, setelah diperbaiki bentuk ketergantungan fungsionalnya bertambah jumlahnya menjadi 3 (tiga) skema relasi. Dan pada tingkat 5NF, hasil normalisasinya adalah semua skema relasi/ tabel hanya terdiri dari 1 atau 2 primary key dan sebuah atribut non key. Hal ini cukup merepotkan pada saat implementasi dan penggunaan basis data tersebut dimana untuk membuat laporan perlu digunakan banyak tabel yang dihubungkan dengan proses join.

Oleh sebab itu, dalam pembahasan di Bab ini hanya dibatasi pada bentuk 1NF hingga 3NF saja.

7.2.1 Bentuk Normal Pertama (1st Normal Form)

Ide dasar dari bentuk 1NF ini adalah menghindari kerangkapan data (redundansi data) pada sebuah tabel. Persyaratannya adalah setiap nilai pada atribut sebuah record/tuple haruslah **atomic (bernilai tunggal)** atau **tidak terdapat multivalued attribute**. Contohnya pada tabel Karyawan di tabel 1 sebelumnya, dimana setiap baris data hanya bernilai tunggal pada setiap atributnya.

Perhatikan tabel berikut :

Tabel 2. Siswa

NIS	Nama	Alamat	No_Telp
001	Samsul	Jakarta	021222322
002	Silvia	Jakrta	08193382382, 0 2 1 9 8 4 4 8 , 0859662626

Terlihat bahwa atribut No_Telp pada data siswa dengan NIS = "002" memiliki 3 (tiga) nilai (*multivalue*), dan hal ini tidak memenuhi persyaratan 1NF.

Solusinya adalah tabel tersebut dipecah dengan memisah-kan unsur atribut yang tidak bersifat atomic menjadi tabel baru menjadi sebagai berikut :

1. Siswa(NIS, Nama, Alamat) dan
2. Siswa_Telepon(NIS, No_Telp)

Contoh kasus lainnya seperti pada tabel berikut :

Tabel 3. Penjualan

No	Tgl	Cust	KdItem	HrgSat	Juml	TotalHrg
001	12/02	Ali	B003 B001 B010	50000 10000 20000	5 2 10	470000

Pada record dengan no transaksi "001", atribut-atribut KdItem, HrgSat dan Juml memiliki 3 (tiga) nilai. Untuk menghindari multivalued attribute diatas, seharusnya tabel tersebut dibuat menjadi seperti berikut :

Tabel 4. Penjualan

No	Tgl	Cust	KdItem	HrgSat	Juml	TotalHrg
001	12/02	Ali	B003	50000	5	470000
001	12/02	Ali	B001	10000	2	470000
001	12/02	Ali	B010	20000	10	470000

Namun hasilnya adalah terdapat nilai atribut yang sama pada beberapa baris (terjadi kerangkapan data), yaitu No, Tgl, Cust, TotalHrg. Hal ini berpeluang menjadi inkonsisten karena untuk menjaga konsistensi, update data harus dilakukan pada semua data yang memiliki nilai yang sama.

Solusinya adalah menghindari kerangkapan data dengan memisah tabel di atas menjadi dua, yaitu menjadi :

1. PenjualanMaster(No, Tgl, Cust, TotalHrg) dan
2. PenjualanDetail(No, KdItem, HrgSat, Juml)

Tabel 5. PenjualanMaster

No	Tgl	Cust	TotalHrg
001	12/02	Ali	470000

Tabel 6. PenjualanDetail

No	KdItem	HrgSat	Juml
001	B003	50000	5
001	B001	10000	2
001	B010	20000	10

7.2.2 Bentuk Normal Kedua (2^{nd} Normal Form)

Persyaratan untuk bentuk normal 2NF adalah :

- Sudah 1NF
- Setiap atribut non-key **harus tergantung penuh** (*full functional dependent*) pada setiap atribut key nya

Yang artinya setiap skema relasi yang ada harus memenuhi 1NF terlebih dahulu sebelum dilakukan normalisasi 2NF

Contoh :

Pada bentuk ketergantungan berikut :

Kode_Barang → Nama_brg, Jenis_brg, Satuan, Harga

Setiap atribut non-key, yaitu Nama_brg, Jenis_brg, Satuan, dan Harga bergantung penuh pada Kode_Barang (sudah memenuhi 2NF).

Perhatikan contoh skema relasi NilaiMHS berikut :

NilaiMHS (NIM, Kode_MK, Nama_Mhs, Nama_MK, NilaiUAS)

Seharusnya ada 3 (tiga) bentuk ketergantungan, yaitu :

1. fd1 : NIM → Nama_Mhs
2. fd2 : Kode_MK → Nama_MK
3. fd3 : NIM,Kode_MK → NilaiUAS

Hanya NilaiUAS yang termasuk bentuk ketergantungan fungsional PENUH (pada fd3) dimana setiap atribut non-key tergantung pada semua atribut keynya (NIM dan Kode_MK), sementara Nama_Mhs dan Nama_Mk hanya tergantung pada sebagian atribut keynya.

Kesimpulannya : Skema **NilaiMHS** belum memenuhi 2NF.

Solusinya : skema relasi di atas dipecah menjadi :

1. Mahasiswa(NIM, Nama_Mhs)
2. MataKuliah(Kode_MK, Nama_MK)
3. NilaiUAS(NIM, Kode_MK,NilaiUAS)

Dan masing-masing skema di atas sudah memenuhi 2NF.

7.2.3 Bentuk Normal Ketiga (3rd Normal Form)

Persyaratan untuk bentuk normal 3NF adalah :

1. Sudah 2NF
2. **Tidak ada** atribut non-key yang **tergantung transitif** terhadap atribut keynya.

Yang berarti skema-skema relasi harus sudah memenuhi 2NF sebelum dilakukan normalisasi 3NF.

Perhatikan skema relasi berikut :

Pegawai(NIP, Nama_Pegawai, Kd_Bagian, Nm_Bagian)

Terlihat Nm_Bagian tergantung pada Kd_ Bagian, bukan pada NIP. Berarti Nm_Bagian tergantung transitif pada NIP, dan hal ini tidak memenuhi 3NF.

Berikut bentuk ketergantungan fungsionalnya :

1. Fd1 : NIP $\rightarrow$ Nama_Pegawai, Kd_Bagian
2. Fd2 : Kd_Bagian $\rightarrow$ Nm_Bagian

Solusinya adalah : Skema relasi di atas dipecah dengan memisahkan bentuk ketergantungan transitifnya menjadi :

1. Pegawai(<u>NIP</u>, Nama_Pegawai, Kd_Bagian) dan
2. Bagian(<u>Kd_Bagian</u>, Nm_Bagian)

BAB
VIII

Bahasa Basis Data

Pendahuluan

Pengguna basis data dan data pada disk dihubungkan oleh sebuah perantara yang disebut dengan DBMS (*Database Management System*) yaitu kumpulan data yang saling terkait yang memiliki bahasa dalam pengoperasiannya (Rizki and Amijaya 2019). Pada bab Bahasa basis data akan dibahas mengenai bagaimana cara berkomunikasi atau berinteraksi antara pengguna dengan basis data itu sendiri, hal tersebut telah diatur tersendiri oleh perusahaan pembuat DBMS. Hal – hal yang menjadi pembahasan pada bahasa basis data diantaranya meliputi sejumlah *statement* atau perintah yang diformulasikan serta dapat digunakan untuk melakukan suatu aksi.

8.1 Deskripsi Bahasa Basis Data

Bahasa basis data dapat diartikan sebagai perantara bagi pengguna dengan basis data yang ada pada mesin, sehingga dapat digunakan sebagai alat berinteraksi (Hasanah 2019). Bahasa ini dapat disisipkan kedalam bahasa pemrograman lain sebagai sebuah konstruksi yang kompleks dan dapat dijalankan sebagai sebuah sistem dengan fitur operasi Basis Data. Bahasa yang menjadi induk penyisipan bahsa basis data disebut dengan inang (*host language*), seperti halnya bahasa pemrograman Java, PHP, C/ C++, Basic, Python, dan lain sebagainya.

Kode yang disusun kemudian dikompilasi menggunakan *compiler* sesuai dengan bahasa yang digunakan sebagai *host language* sehingga perintah yang dilakukan sesuai instruksi dapat dilaksanakan dan menghubungkan pengguna dengan DBMS. Istruksi atau perintah ditangkap oleh *host language* sesuai dengan keperluannya. Fasilitas interaktif dengan bahasa SQL juga dimiliki oleh DBMS sebagai fasilitas langsung bagi operator atau pengguna untuk menyelesaikan tugas – tugas terkait database secara keseluruhan.

8.2 Komponen Bahasa Basis Data

a. Data Definition Language (DDL)

DDL memiliki fungsi sebagai pembuat data baru, pembuatan indeks, mengubah tabel, menentukan struktur tabel dan alin sebagainya. Spesifik memberikan skema atau struktur basis data dan dapat dapat juga disebut sebagai himpunan definisi data yang secara khusus disimpan pada *directory*.

Contoh : perintah "create"

Run SQL query/queries on server "127.0.0.1":

```sql
CREATE DATABASE karyawan;
```

Gambar 8.1 *Create Database*

```sql
CREATE TABLE data_karyawan (
    nid int (9),
    nama_d char (30),
    tempat_lhr char (10),
    tgl_lhr date,
    jkelamin char (6),
    alamat varchar (50),
    kota varchar (20),
    kodepos int (5),
    gaji int (8)
```

Gambar 8.2 *Create* Tabel

Perintah untuk membuat *database* ditunjukkan pada gambar 8.1, pengguna membuat sebuah database yang nantinya akan berisi tabek – tabel yang diinginkan untuk mentimpan *record*. Perintah *create table* diikuti dengan struktur tabel beserta tipe data pada masing – masing atributnya, agar pengguna dapat memberikan input sesuai dengan tipe yang diinginkan. Tipe data secara berurutan dapat dilihat pada gambar

8.3 yaitu : *integer, character, date,* dan *varchar*, maka isian yang akan ditambahkan harus menyesuaikan dengan tipe data tersebut.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	nid	int(9)			Yes	NULL			Change Drop More
☐	2	nama_d	char(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	3	tempat_lhr	char(10)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	4	tgl_lhr	date			Yes	NULL			Change Drop More
☐	5	jkelamin	char(6)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	6	alamat	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	7	kota	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	8	kodepos	int(5)			Yes	NULL			Change Drop More
☐	9	gaji	int(8)			Yes	NULL			Change Drop More

Gambar 8.3 *Type Data* Tabel

b. Data Manipulation Language (DML)

Bahasa untuk melakukan manipulasi serta mengambil data pada suatu basis data, manipulasi yang dimaksud berupa :

a. Penambahan atau penyisipan data pada file / tabel.

b. Melakukan penghapusan data pada file / tabel.

c. Melakukan perubahan data pada file / tabel.

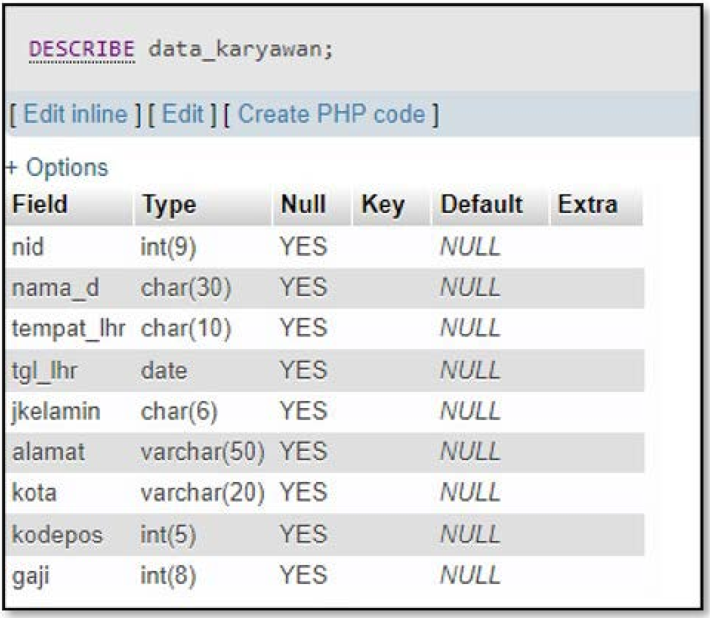
d. Melakukan penelusuran data pada file / tabel.

DMLmerupakansebuahbahasayang memiliki tujuan untuk mempermudah penggunanya dalam mengakses data pada model data yang ada. Jenis DML dibagi menjadi dua, sebagai berikut :

1. **Prosedural,** membutuhkan peran mengguna untuk menentukan secara spesifik data apa yang digunakan, serta bagaimana mendapatkan data tersebut.
2. **Nonprosedural,** pengguna menentukan secara spesifik data yang dibutuhkan tanpa mengetahui bagaimana cara mendapatkannya.

Contoh paket bahasa prosedurah DML : FoxBase,dBase, sedangkan untuk Nonprosedural DML : SQL (Structure Query Language), QBE (Query By Example).

Query merupakan sebuah pernyataan yang digunakan untuk mengambil informasi yang ada pada suatu *basis data,* artinya segala sesuatu yang tersimpan pada basis data dapat dikelola menggunakan sebuah perantara yang dinamakan *query*. Proses pengambilan data menggunakan *query* disebut juga dengan *query language* yang merupakan sebuah DML. Fungsi dasar query dalam mengolah table dapat dilihat pada gambar berikut :



```
DESCRIBE data_karyawan;
```

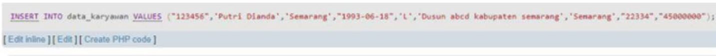
[Edit inline] [Edit] [Create PHP code]

+ Options

Field	Type	Null	Key	Default	Extra
nid	int(9)	YES		NULL	
nama_d	char(30)	YES		NULL	
tempat_lhr	char(10)	YES		NULL	
tgl_lhr	date	YES		NULL	
jkelamin	char(6)	YES		NULL	
alamat	varchar(50)	YES		NULL	
kota	varchar(20)	YES		NULL	
kodepos	int(5)	YES		NULL	
gaji	int(8)	YES		NULL	

Gambar 8.4 *Query* deskripsi tabel

Gambar 8.4 merupakan *query* dengan perintah DESCRIBE namatebel atau juga bisa disingkat dengan DESC namatebel. Fungsi "describe" adalah untuk menampilkan struktur table yang telah dibuat sebelumnya, agar pengguna dapat melihat format table yang dimaksud untuk menghindari kesalahan memasukkan *type data*.



Gambar 8.5 *Query* menyisipkan *value* pada tabel

Gambar 8.6 *Query* melihat *value* pada table

Perintah mengisi tabel pada database ditunjukkan pada gambar 8.5 dengan satu *record*, dan dilanjutkan dengan melihat isi dari data yang diinputkan menggunakan perintah "select * from namatabel" seperti pada gambar 8.6. Mengisi tabel menggunakan *query* dapat dilakukan dengan lebih dari satu *record* seperti pada gambar 8.7 berikut.

Gambar 8.7 *Query* menyisipkan banyak *value* pada table

Perintah *insert* pada gambar 8.7 adalah memasukkan *value* lebih dari satu baris *record* dengan format sebagai berikut :

mysql> INSERT INTO table_name VALUES
-> ('value11','value12','value13',...,'value n'),
-> ('value21','value22','value23',...,'value n'),
-> ('value n1','value n2','value n3','...','value nn');

Dengan perintah tersebut pengguna dapat mengisi tabel dengan sekali perintah sebanyak *record* yang diinginkan seperti Nampak pada gambar 8.7 berikut.

```
SELECT * FROM data_karyawan;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

nid	nama_d	tempat_lhr	tgl_lhr	jkelamin	alamat	kota	kodepos	gaji
123456	Putri Dianda	Semarang	1993-06-18	L	Dusun abcd kabupaten semarang	Semarang	22334	45000000
223434	Putra Wardhana	Bandung	1993-12-06	L	Jalan Duren No.5	Bandung	33224	5000000
234543	Lidya Putri	Jakarta	1990-11-11	P	Jalan Nangka No.10	Jakarta	12445	5000000
223434	Leni Mardianti	Palembang	1992-09-05	P	Jalan Melati No.30	Jakarta	33224	5000000

Gambar 8.7 *Query* menyisipkan banyak *value* pada table

Gambar 8.7 menujukkan pengambilan data secara keseluruhan tanpa syarat dengan perintah *select*, bahasa *query* secara lanjut akan dibahas pada bab berikutnya.

a. Data Control Language

Konsep ini digunakan untuk merubah hak akses, serta memberikan *role* (manajerial), serta iso dan topik lain yang berhubungan dengan keamanan database.

Contoh DCL

- Beberapa perintah SQL yang termasuk dalam DCL yaitu GRANT dan REVOKE.

1. **GRANT** SELECT, UPDATE ON Nama_database TO user1, user2;

2. **REVOKE** SELECT, UPDATE ON Nama_ database FROM user1, user2;

   Penjelasan :

   - Perintah **GRANT** dipergunakan untuk memberikan kewenangan terhadap pengguna dalammelakukan pengoperasian pada suatu database.
   - Perintah **REVOKE** memiliki fungsi untuk menghilangkan hak perintah pada user terhadap objek atau skema database.

8.3 Pengguna *Database*

Database diciptakan memiliki unsur utama berupa tabel - tabel yang memiliki keterkaitan satu dengan yang lainnya, keterkaitan tersebut menjadikan database memiliki informasi yang dapat dimanfaatkan oleh penggunanya. Pengguna dalam pemanfaatan basis data dibagi menjadi beberapa jenis :

1. Database Manager

Basis Data doperasikan menggunakan sebuah *tools* atau perangkat lunak yang menjadi interface sebagai perantara antara penyimpanan data dalam *database* dengan pengguna, dikarenakan database merupakan informasi yang disimpan dalam koputer secara sistematis (Duggan, Roderick, and Sieburg 1970).

2. *Database Administrator*

Merupakan seorang pengguna yang memiliki akses kontrol penuh terhadap sistem yang ada baik dari segi data yang tersimpan, maupun program yang mengakses dan mengolah data yang ada. Adapun fuingsi *database administrator* antara lain :

a. Mendefinisikan pola struktur *database*.
b. Mendefinisikan stuktur penyimpanan serta metode akses.
c. Memiliki kemampuan memodifikasi pola serta organisasi phisik.
d. Memberikan hak akses pada user untuk mengakses data.
e. Melakukan analisa terspesifik terhadap integritas data.

3. *Database User*

Pengguna database memiliki *rule* yang dapat disesuaikan dengan peruntukan pengaksesan, 4 macam *rule* pengguna berdasarkan keperluan dan cara aksesnya, antara lain : 1. *Programmer* Aplikasi, 2.

Casual User (mahir), 3. *User* umum (*end user*), 4. *User* Khusus (*specialized user*).

8.4 Basis Data Relasional

Relasi database merupakan kumpulan dari data dengan ketentuan hubungan yang telah ditentukan sebelumnya. Penyusunannya menggunakan set tabel dengan format kolom dan baris untuk menyimpan informasi tentang objek tertentu dan media penyimpanan nilai atribut tertentu.

1. Ilustrasi Model Data Relasional

Pengetahuan mengenai contoh data yang akan digunakan sangatlah penting, hal tersebut harus dilakukan sebelum menerapkan basis data. Karekteristik data menjadi hal yang harus diketahui oleh *data engineer* sebagai dasar untuk menetapkan struktur masing – masing tabel. Contoh data sebagai berikut :

Tabel 8.1 *Database* Karyawan

nid	nama_d	tempat_lhr	tgl_lhr	jkelamin	alamat	kota	kodepos	gaji
10234	Doni Kurniawan	Jakarta	14/03/1988	pria	Jl. Mawar 3 No.2	Jakarta Selatan	43445	1300000
12990	Pratama Arya	Semarang	23/05/1980	pria	Jl. Melati 4 No.6	Jakarta Barat	43447	1200000
19202	Setiawan	Lampung	22/08/1990	pria	Jl. Flamboyan 7 No.45	Bekasi Barat	64432	1100000
18990	Tamara Leoni	Jakarta	07/06/1992	wanita	Jl. Duren Tiga 45 No 30	Jakarta Selatan	43445	1000000
12498	Wira Wardhana	Tegal	12/25/1989	pria	Jl. Peninggaran No 1	Jakarta Selatan	43445	1000000
34320	Puja Pratama	Semarang	11/22/1993	wanita	Jl. Menoreh no.2	Semarang	12432	1000000
23123	Rani Hidayanti	Semarang	22/09/1992	wanita	Jl. Banteng no.56	Semarang	12432	1100000
23435	Rahmat	Bekasi	23/11/1991	pria	Jl. Bunga 6 No.33	Bekasi Barat	64432	1100000
12341	Hidayatullah	Kendal	18/06/1993	pria	Jl. Arjuna No 55	Kendal	51387	3000000
23454	Wahyu Setyani	Blora	11/05/1989	wanita	Jl. Angkasa 6 No.12	Bekasi Barat	64432	1100000

Karakteristik Data :

1. Terdiri dari 9 kolom
2. Pada kolom pertama menunjukkan isi mengenai angka akan tetapi tidak merupakan jumlah, sering juga disebut dengan alfa numerik denga lebar tidak berubah (5 karakter / digit).
3. Kolom kedua diisi dengan karakter atau string yang memiliki panjang maksimum sebesar 30 karakter.
4. Kolom ketiga diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 10 karakter.
5. Kolom keempat diisi dengan data penanggalan / *date* sesuai dengan pengaturan.

6. Kolom kelima diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 6 karakter.
7. Kolom keenam diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 50 karakter.
8. Kolom ketujuh diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 20 karakter.
9. Kolom kedelapan diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 5 karakter.
10. Kolom kesembilan diisi dengan karakter atau string yang memiliki Panjang maksimum sebesar 8 karakter.

Karakteristik data karyawan ditunjukkan pada Tabel 8.1, tabel tersebut merupakan hasil dari pembuatan *database* sampai dengan membuat tabel, sehingga dapat dibuat database secara fisik menggunakan *tools* yang telah dicontohkan menggunakan *phpMyAdmin*. Berdasarkan tabel tesebut dapat dirumuskan bahwa kebutuhan minimal dalam penyusunannya adalah sebagai berikut :

1. Nama Kolom (*field* / atribut).

2. Tipe data (*data type*).

3. Lebar data (jumlah karakter).

4. Pendefinisian kolom.

BAB
IX

Data Definition Languange

Pendahuluan

Standard Query Language (SQL) merupakan bahasa pemrograman yang berperan dalam mengolah data dengan basis relasional. Istilah SQL berawal dari sebuah artikel yang ditulis oleh para peneliti IBM yang mengatakan pada masa yang akan datang terdapat suatu bahasa standar untuk mengakases data. Bahasa mempunyai istilah *Structured English Query Language* (SEQUEL). Dikarenakan adanya masalah dalam hak dagang maka istilah SEQUEL menjadi SQL sampai dengan saat ini. Perkembangan SQL ini dari waktu ke waktu mengalami perubahan dari SQL86, SQL89, SQL92, SQL99, dan MySQL (Brooks, 2022).

Dalam melakukan perannya mengolah basis data, terdapat beberapa perintah dalam SQL. Tiga diantaranya adalah *Data Definition Language* (DDL), *Data Manipulation Language* (DML), dan *Data Control Language* (DCL). Untuk DDL sendiri, istilah muncul pertama kali pada *Conference of Data System Languages* (CODASYL) yang merupakan sebuah organisasi yang bertugas untuk membuat aturan baku untuk COBOL dan DBMS (Phillips, 1985).

DDL ini berfungsi untuk membuat tabel dan memodifikasi struktur obyek dalam suatu *database*. Perintah dalam DDL adalah create, alter, drop, rename, comment, dan truncate. Create adalah perintah yang digunakan untuk membuat tabel, sedangkan perintah lainnya berfungsi untuk memodifikasi struktur obyek tabel.

Bab Isi

9.1 Structure Query Language

Relasi yang terdapat dalam suatu *database* ditentukan dengan *Data Definition Language* (DDL). Penggunaan DDL ini dapat memberikan informasi dari setiap relasi yang terjadi. Informasi tersebut meliputi (Java T Point, 2022):

1. Skema dari setiap relasi.
2. Jenis nilai yang terkait dengan setiap atribut.
3. Batasan integritas.
4. Himpunan indeks untuk setiap relasi.
5. Informasi keamanan dan otorisasi untuk setiap relasi.
6. Struktur penyimpanan fisik dari setiap relasi pada *harddisk.*

Tipe-tipe data standar yang berlaku pada SQL termasuk (Hakim, 2019)

1. **char**(*n*). Tipe data ini adalah karakter yang mempunyai panjang tetap, dimana *n* adalah panjang yang ditentukan oleh pengguna.

   Contoh

   NIM char(12) → '0325058001'

0	3	2	5	0	5	8	0	0	1		

2. **varchar**(*n*). Perbedaan dengan tipe data char, pada tipe data ini panjangnya berubah-ubah sesuai dengan yang diisikan oleh pengguna. Nilai n menunjukkan panjang data maksimum yang dapat disimpan.

   Contoh

   Nama varchar(10) → 'Elza','Anna'

E	l	z	a	A	n	n	a

3. **int**. Berupa bilangan bulat tanpa nilai decimal

   Contoh

   Nilai int(4) → nilai dapat berisi bilangan bulat dengan maksimal 4 digit. Misalnya Nilai = 1000

4. **double**. Bilangan dengan angka decimal

   Contoh

   IPK double(3,2) → IPK = 3.25. Angka 3 untuk menunjukkan bahwa total digit yang disimpan dalam variabel IPK adalah 3. Sedangkan angka 2 untuk menunjukkan jumlah dibelakang koma adalah dua digit. Penggunaan double ini mempunyai akurasi sampai dengan 15 angka decimal.

5. **float**(*n*). Sama halnya dengan double, tipe data float digunakan untuk menyimpan bilangan dengan angka decimal yang mempunyai akurasi 7 angka decimal

   Contoh

   Satuan float(3,2) → Satuan = 1.30.

6. **date**. Tipe data ini digunakan untuk menunjukkan tanggal dan waktu.

   Contoh

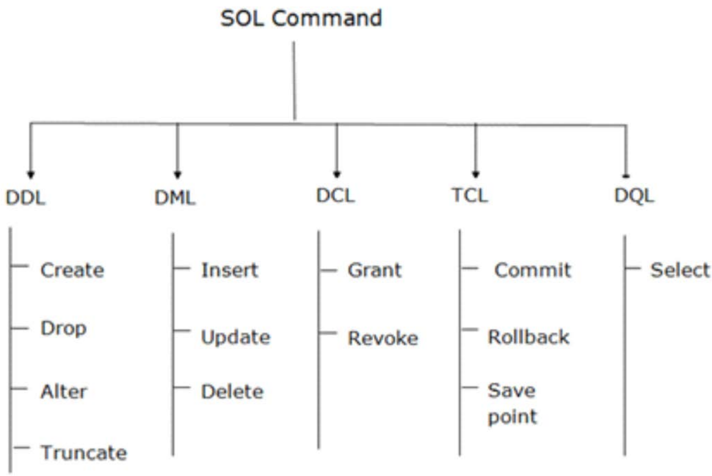
   dt date → 2022-10-20

   tm time → 20:40:40

Terdapat lima jenis perintah dalam SQL yaitu: DDL, DML, DCL, TCL, dan DQL.

9.2 Data Definition Language

Data Definition Language (DDL) adalah sekumpulan instruksi dalam Structure Query Language (SQL) yang berfungsi untuk melakukan manipulasi pada struktur basis data seperti membuat, menghapus, ataupun mengubah tabel (Amornchewin, 2018).



Gambar 9.1. Lima jenis perintah SQL

9.2.1 Create

Untuk membuat tabel baru dalam suatu basis data perintah yang digunakan adalah *create table*.

Sintaks *CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);*

Berikut adalah perintah untuk membuat tabel *Pasien*

<u>Contoh 1</u>

create table *Pasien*

(ID_Pasien **int**(5),

Nama_Pasien **varchar**(50),

Jenis_*Kelamin* **varchar**(10),

Umur **int**(2),

Alamat **varchar**(50),

primary key(*ID_Pasien*));

Tabel *Pasien* yang sudah didefinisikan diatas, mempunyai lima buah kolom yaitu ID_Pasien dengan tipe data integer; Nama_Pasien; Jenis_Kelamin; Umur; dan Alamat masing-masing mempunyai tipe data integer yang masing-masing mempunyai panjang 2 dan 5 digit serta tipe data varchar yang panjang maksimalnya adalah 10 dan 50 karakter. Pada tabel *Pasien* atribut ID_Pasien ditentukan sebagai *primary key*. Tampilan dari tabel *Pasien* yang masih kosong akan terlihat seperti dibawah ini.

ID_Pasien	Nama_Pasien	Jenis_Kelamin	Umur	Alamat

<u>Contoh 2</u>

create table *Dokter*

(*ID_Dokter* **int** (5) not null,

Nama_Dokter **varchar** (50) not null,

primary key (*ID_Dokter*));

Pada tabel *Dokter* muncul keterangan not null yang artinya kedua kolom tersebut tidak boleh kosong sehingga keduanya harus berisi data. Tampilan tabel *Dokter* sebagai berikut

ID_Dokter	Nama_Dokter

<u>Contoh 3</u>

create table *Obat*

(*ID_Obat* **int**(7),

Nama_Obat **varchar** (50),

Jenis_Obat **varchar** (10),

primary key (*ID_Obat*));

Tampilan tabel *Obat* sebagai berikut

ID_Obat	Nama_Obat	Jenis_Obat

<u>Contoh 4</u>

create table *Rawat_Jalan*

(*ID_RM* **int**(5),

ID_Pasien **int**(5),

ID_Dokter **int**(5),

ID_Obat **int**(7),

Tanggal_Berobat **date**,

Keluhan_Pasien **varchar**(100),

Diagnosis **varchar**(100),

Terapi **varchar**(100),

primary key (*ID_RM*), **foreign key** (*ID_Pasien*) **references** (*Pasien*), **foreign key** (*ID_Dokter*) **references** (*Dokter*), **foreign key** (*ID_Obat*) **references** (*Obat*));

Tampilan dari tabel *Rawat_Jalan*

ID_RM	ID_Pasien	ID_Dokter	ID_Obat
Tanggal_Berobat	**Keluhan_Pasien**	**Diagnosis**	**Terapi**

Pada keempat contoh perintah DDL terdapat beberapa batasan yang diterapkan yang bertujuan untuk menjaga konsistensi relasi basis data.

Tabel 9.1 Batasan dalam DDL

Batasan	Keterangan
not null	not null dapat diterapkan pada semua tipe data. Penggunaan batasan ini mempunyai arti bahwa nilai pada variabelnya tidak boleh kosong.
primary key	batasan ini digunakan untuk mengidentifikasi baris dalam suatu tabel dengan nilai unik. Variabel yang menjadi primary key tidak boleh bernilai kosong.
foreign key	batasan ini sama dengan primary key namun digunakan untuk menghubungkan antar tabel. Nilai dari foreign key juga harus unik dan tidak boleh kosong serta harus terdapat rujukan tabel yang terhubung.

9.2.2 Drop

Perintah drop mempunyai arti untuk menghapus struktur tabel dan semua obyek yang terdapat dalam basis data.

Sintaks

DROP TABLE table_name

Contoh 4

drop table *Obat*

dengan perintah ini maka tabel obat dan semua obyek yang terdapat didalamnya akan terhapus

9.2.3 Alter

Alter adalah perintah yang digunakan untuk melakukan perubahan struktur tabel. Perubahan tersebut dapat berupa modifikasi, menghapus, dan menambah kolom baru.

Sintak untuk menambah kolom

ALTER TABLE *table_name* ADD *column_name datatype*;

Contoh 5

ALTER TABLE *Pasien* ADD *Jenis_Asuransi* **varchar**(20);

Tabel Pasien akan berubah menjadi

ID_Pasien	Nama_Pasien	Jenis_ Kelamin	Umur	Alamat	Jenis_ Asuransi

Contoh 6

ALTER TABLE *Dokter* ADD *Tipe_Dokter* **varchar**(10);

Tabel Dokter akan berubah menjadi

ID_Dokter	Nama_Dokter	Tipe_Dokter

Sintak untuk menghapus kolom

ALTER TABLE *table_name* DROP COLUMN *column_name*;

<u>Contoh 7</u>

ALTER TABLE *Pasien* DROP COLUMN *Jenis_Asuransi*;

Maka tampilan tabel Pasien akan kembali seperti dibawah ini

ID_Pasien	Nama_Pasien	Jenis_Kelamin	Umur	Alamat

Sintak untuk modifikasi kolom

ALTER TABLE *table_name* RENAME COLUMN *old_name* to *new_name*;

<u>Contoh 8</u>

ALTER TABLE *Obat* RENAME COLUMN *Jenis_Obat* to *Kategori_Obat*;

Tampilan tabel Obat berubah menjadi

ID_Obat	Nama_Obat	Kategori_Obat

Sintak untuk modifikasi tipe data

ALTER TABLE *table_name* MODIFY COLUMN *column_name datatype*;

Contoh 9

ALTER TABLE *Pasien* MODIFY COLUMN *Jenis_Kelamin* ***char****(2)*;

Perintah ini, akan mengubah kolom Jenis_ Kelamin yang awalanya bertipe data varchar dengan panjang maksimal 10 kararter, tipe datanya diubah menjadi char dengan panjang maksimal 2 karakter.

9.2.4 Truncate

Perintah truncate digunakan untuk menghapus isi dari tabel saja bukan struktur tabel secara keseluruhan.

Sintaks

TRUNCATE TABLE table_name

Contoh 10

TRUNCATE TABLE Obat

Latihan Soal

A. Pilihan Ganda

1. Apa kepanjangan dari SQL
    a. Structure Question Language
    b. Structure Query Language

c. Strong Question Language

2. Perintah DDL untuk membuat tabel baru

a. create table

b. create database

c. create column

3. Nilai unik yang digunakan sebagai identifikasi tabel adalah

a. Not null

b. Foreign key

c. Primary key

B. Jawab Singkat

1. Tulis perintah DDL untuk membuat tabel baru dengan nama Alamat_Pasien.

....

(ID_Pasien int(5),

Alamat varchar(255),

Kota varchar(255),

Kecamatan varchar(255),

Kelurahan varchar(255),

Kode_Pos int(5));

2. Tulis perintah untuk menghapus tabel Alamat_Pasien............................... Alamat_

Pasien

BAB X

Data Manipulation Language

Pendahuluan

Database Management System (DBMS) adalah *software* yang digunakan untuk menjelaskan, menyimpan, serta kueri data secara independen. Semua *database management system* berisi penyimpanan dan komponen manajemen. Komponen penyimpanan mencakup semua data yang disimpan dalam bentuk terorganisir ditambah deskripsinya. Komponen manajemen berisi kueri dan *data manipulation language* untuk mengevaluasi dan mengedit data dan informasi. Komponen ini tidak hanya berfungsi sebagai antarmuka pengguna, tetapi juga mengelola akses dan izin pengeditan untuk pengguna (Meier & Kaufmann, 2019).

Setelah skema *database* dikompilasi dan database diisi dengan data, pengguna harus memiliki beberapa cara untuk memanipulasi *database*. Manipulasi yang umum termasuk pengambilan, penyisipan, penghapusan, dan modifikasi data. DBMS menyediakan satu set operasi atau bahasa yang disebut *data manipulation language* (DML) untuk tujuan ini (Shamkant & Ramez, 2017).

"Data manipulation language (DML)" artinya instruksi sesudah *database* dibuat dan tabel dibuat memakai perintah DDL, perintah DML dapat dipergunakan untuk memanipulasi data pada tabel serta *database*. Kenyamanan memakai perintah DML ialah bahwa perintah tersebut dapat dengan mudah diganti dan dibatalkan bila terdapat modifikasi yang keliru pada data atau nilainya yang telah dibuat. (Base, 2020). Perintah DML yang digunakan untuk melakukan tugas tertentu adalah:

1. "*Insert*" – Untuk penyisipan baris baru dalam tabel.
2. "*Update*" – Untuk modifikasi nilai data yang terdapat dalam baris tabel.
3. "*Delete*" – Untuk menghapus baris yang dipilih atau tabel lengkap dalam *database*.
4. "*Lock*" – Untuk menentukan akses pengguna ke "hanya baca" atau "baca dan menulis" hak istimewa.
5. "*Merge*" – Untuk menggabungkan beberapa baris dalam sebuah tabel.

Jenis instruksi *Data manipulation language* (DML) ini yang berkaitan dengan informasi yang terdapat di dalam tabel, tentang seperti apa menginput, menghapus, memperbaharui dan membaca informasi yang tersimpan di dalam *database*. DML dapat berupa level tinggi (berorientasi set, nonprosedural) atau level rendah (berorientasi *record*, prosedural). DML tingkat tinggi dapat disematkan dalam bahasa pemrograman *host*, atau dapat digunakan sebagai bahasa yang berdiri sendiri; dalam kasus terakhir ini sering disebut *query language* (Connolly & Begg, 2019).

Pada bab sebelumnya, kami sudah membahas bagaimana membuat tabel dan struktur tabel. Bab ini cukup mudah dan akan mengenalkan anda serta membahas berbagai pernyataan kueri SQL tingkat lanjut menggunakan *data manipulation languages*. Untuk saat ini, mari kita pelajari semua kemampuan *data manipulation languages*.

Bab Isi

10.1 *Insert*

10.1.1 Pernyataan SQL INSERT INTO.

Pernyataan INSERT INTO digunakan dalam memasukkan *new record* pada tabel. Pernyataan INSERT INTO bisa ditulis menggunakan 2 metode (Python, 2020). Metode awal memastikan nama kolom serta nilai yang hendak disisipkan:

INSERT INTO nama_tabel (kolom1, kolom2, kolom3, ...)

VALUES (value1, value2, value3, ...);

Bila anda memberikan tambahan nilai buat seluruh kolom tabel, kita tidak butuh memastikan nama kolom dalam *query* SQL. Tetapi, kita harus memastikan urutan nilai nya harus berada dalam urutan yang sama dengan kolom yang berada pada tabel. Sintaks INSERT INTO dapat kita lihat dibawah ini:

INSERT INTO nama_tabel

VALUES (value1, value2, value3, ...);

Query untuk **INSERT INTO Example** adalah sebagai berikut :

INSERT INTO Pelanggan (NamaPelanggan, NamaKontak, Alamat, Kota, KodePos, Negara)

VALUES ('Rachmalia', 'Rachmalia Feta', 'Puri Nirwana 3', 'Cibinong', ' 16913', 'Indonesia');

Insert Data Hanya di Kolom Tertentu

Dimungkinkan juga untuk hanya memasukkan data pada kolom tertentu saja sesuai yang kita pilih. *Statement* SQL dibawah ini akan memasukkan *new record*, namun hanya insert data pada kolom "NamaPelanggan", "Kota", serta "Negara" (IDPelanggan nantinya diperbarui secara otomatis karena auto increment):

INSERT INTO Pelanggan (NamaPelanggan, Kota, Negara)

VALUES ('Rachmalia', 'Cibinong', 'Indonesia');

10.1.2 *SQL NULL Values*

Field yang isi nilainya NULL merupakan field tanpa nilai. Bila field pada tabel berbentuk opsional, diperbolehkan untuk *insert new record* ataupun *update record* tanpa melakukan penambahan nilai ke field yang dimaksud. Setelah itu, field akan disimpan menggunakan nilai NULL. How to Test for NULL Values? Kita tidak bisa melakukan pengujian terhadap nilai NULL menggunakan operator perbandingan, semacam =, <, ataupun <>.

Kita perlu memakai operator IS NULL serta IS NOT NULL selaku gantinya. Contoh *query* nya adalah selaku berikut :

Sintaks IS NULL

```
SELECT nama_kolom
FROM nama_tabel
WHERE nama_kolom IS NULL;
```

Sintaks IS NOT NULL

```
SELECT nama_kolom
FROM nama_tabel
WHERE nama_kolom IS NOT NULL;
```

Operator IS NULL

Operator IS NULL digunakan ketika menguji nilai kosong (nilai NULL).
SQL berikut mencantumkan semua Pelanggan dengan nilai NULL di field "Alamat":

```
SELECT NamaPelanggan, Nama-
Kontak, Alamat
FROM Pelanggan
WHERE Alamat IS NULL;
```

Tip: Selalu gunakan IS NULL untuk mencari nilai NULL.

Operator IS NOT NULL

Operator IS NOT NULL dapat kita gunakan ketika menguji nilai yang bukan kosong (nilai NOT NULL).
SQL berikut mencantumkan semua Pelanggan dengan nilai di field "Alamat":

```
SELECT NamaPelanggan, NamaKon-
tak, Alamat
FROM Pelanggan
WHERE Alamat IS NOT NULL;
```

10.2 *Update*

10.2.1 Pernyataan SQL UPDATE

Pernyataan UPDATE bisa kita gunakan dalam mengubah *record* yang terdapat pada tabel (Vega, 2020). *Query* umum nya dapat kita lihat di bawah ini :

```
UPDATE nama_tabel
SET kolom1 = value1, kolom2 = value2, ...
WHERE kondisi;
```

Statment SQL dibawah ini menampilkan pembaharuan pelanggan yang pertama (IDPelanggan = 1) dengan contact person baru serta kota baru.

UPDATE Pelanggan

SET NamaKontak = 'Asep Rahmat Ginanjar', City= 'Cianjur'

WHERE IDPelanggan = 1;

UPDATE Multiple Records

Klausa ini merupakan klausa WHERE yang mana dapat menentukan berapa jumlah *record* yang dapat kita perbaharui. *Statement* SQL dibawah ini menampilkan pembaharuan NamaKontak menjadi "Anjar" untuk seluruh *record* yang negaranya "Malaysia":

UPDATE Pelanggan

SET NamaKontak ='Anjar'

WHERE Negara= ' Malaysia';

Update Warning!

Hati-hati ketika anda ingin memperbarui *record*. Bila anda tidak menuliskan klausa WHERE seperti sintaks dibawah ini, maka SELURUH *record* nantinya diperbarui! Misalnya seperti *query* dibawah ini :

UPDATE Pelanggan

SET NamaKontak ='Anjar';

10.3 Delete

Sintaks DELETE dapat kita gunakan dalam menghapus *record* yang ada pada tabel (Foster & Godbole, 2022). *Query* umum nya dapat kita lihat di bawah ini :

DELETE FROM nama_tabel WHERE kondisi;

Pernyataan SQL berikut menghapus pelanggan "Asep Rahmat Ginanjar" dari tabel "Pelanggan", contoh *query* nya sebagai berikut :

DELETE FROM Pelanggan

WHERE NamaPelanggan = 'Asep Rahmat Ginanjar';

Delete Semua *Records*

Kita dapat menghapus seluruh baris yang ada pada tabel tanpa kita harus menghapus tabelnya juga. Artinya adalah struktur yang ada pada tabel, atribut yang ada pada tabel, serta indeks yang ada pada tabel akan tetap utuh, *query* umum nya dapat kita lihat di bawah ini :

DELETE FROM nama_tabel;

Contoh *query* menghapus seluruh baris yang ada pada tabel Pelanggan adalah:

DELETE FROM Pelanggan;

10.4 *Select*

10.4.1 Pernyataan SQL SELECT

Pernyataan SELECT dapat kita gunakan dalam memilih atau menentukan data dari *database*. Data yang di pilih akan ditampilkan serta disimpan pada tabel hasil, yang dapat kita sebut sebagai kumpulan atau set hasil (w3schools, 2022). *Query* umumnya adalah sebagai berikut :

SELECT kolom1, kolom2, ...

FROM nama_tabel;

Sintaks diatas, kolom1, kolom2, ... merupakan nama-nama field pada tabel yang akan kita pilih atau kita tampilkan datanya. Bila kita akan memilih dan ingin menampilkan seluruh field yang ada pada tabel, kita dapat menggunakan *query* dibawah ini :

SELECT * FROM nama_tabel;

Tabel 10.1 Tabel Pelanggan

IDPelanggan	Nama Pelanggan	Nama Kontak	Alamat	Kota	Kode Pos	Negara
1	Asep Rahmat Ginanjar	Anjar	Shaffa Residence B8	Cibinong	16913	Indonesia
2	Aini Fazriani	Aini	Garden City	Bandung	19541	Indonesia
3	Aisyah Irani Davian	Aisyah Irani	Calle Rio Sonora 1000 Colonia	Cuernavaca	54981	Mexico
4	Nuning Widya Rachmanita	Nuning	Ophir c10	Padang	14593	Indonesia
5	Neneng Rachmalia Feta	Feta	Puri nirwana 3	Cibinong	16913	Indonesia

SQL dibawah ini menampilkan kolom "NamaPelanggan" dan juga kolom "Kota" yang ada pada tabel 10.1 tabel "Pelanggan", *query* nya sebagai berikut :

SELECT NamaPelanggan, Kota FROM Pelanggan;

10.4.2 Pernyataan SQL SELECT DISTINCT

Untuk mengembalikan atau menampilkan nilai yang berbeda *(different)*, kita dapat menggunakan *query* SELECT DISTINCT. *Query* ini dapat kita gunakan didalam tabel dan kolom yang berisi data redundan atau bernilai sama atau data yang terduplikasi. Akan tetapi, terkadang kita hanya ingin menampilkan nilai atau data yang berbeda. Query umum nya dapat kita lihat dibawah ini :

SELECT DISTINCT kolom1, kolom2, ...

FROM nama_tabel;

Contoh query yang kita implementasikan pada data seperti berikut:

Pernyataan SQL berikut mencantumkan jumlah negara pelanggan yang berbeda:

SELECT COUNT(DISTINCT Negara) FROM Pelanggan;

Number of Records: 91

Country
Germany
Mexico
Mexico
UK
Sweden
Germany
France
Spain
France

Statement SQL dibawah ini menampilkan data berbeda yang terdapat pada kolom "Negara" di tabel "Pelanggan":

SELECT DISTINCT Negara FROM Pelanggan;

Data yang dihasilkan seperti dibawah ini:

Number of Records: 21

Country
Germany
Mexico
UK
Sweden
France
Spain
Canada
Argentina

Pernyataan SQL berikut mencantumkan jumlah negara pelanggan yang berbeda:

SELECT COUNT(DISTINCT Negara) FROM Pelanggan;

Number of Records: 1

COUNT(DISTINCT Country)
21

10.4.3 SQL WHERE Clause

Sintaks WHERE membantu kita dalam memfilter *records*. Sintaks WHERE juga dapat membantu kita dalam mengekstrak khusus *records* yang memenuhi syarat tertentu saja. Kueri secara umumnya adalah sebagai berikut :

SELECT kolom1, kolom2, ...

FROM nama_tabel

WHERE kondisi;

Catatan : Sintaks WHERE bisa kita gunakan dalam *statement* SELECT, namun tidak hanya *statement* SELECT saja kita juga bisa mengganakan WHERE dalam *statement* UPDATE, DELETE, dan lain-lain.

Statement SQL dibawah ini menampilkan seluruh Pelanggan dari negara "Mexico", di tabel "Pelanggan":

SELECT * FROM Pelanggan

WHERE Negara='Mexico';

Data yang dihasilkan seperti dibawah ini:

2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico
80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123	México D.F.	05033	Mexico

Sintaks SQL WHERE (Text Fields vs. Numeric Fields). Saat kita menuliskan SQL, jangan lupa menambahkan tanda kutip satu/tunggal di dekat nilai teks (sebagian besar sistem basis data mengizinkan tanda kutip double). Akan tetapi, field numerik tidak diperbolehkan diapit oleh tanda kutip, *query* nya seperti berikut :

SELECT * FROM Pelanggan

WHERE IDPelanggan=1;

Sintaks SQL WHERE untuk Operator dapat kita implementasikan juga dalam klausa WHERE, operator-operator berikut dapat kita gunakan di klausa WHERE:

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Contoh *query* sesuai dengan operator-operator diatas diurutkan dari operator paling atas sampai operator paling bawah :

1. SELECT * FROM Produk WHERE Harga = 18;
2. SELECT * FROM Produk WHERE Harga > 30;
3. SELECT * FROM Produk WHERE Harga < 30;
4. SELECT * FROM Produk WHERE Harga >= 30;
5. SELECT * FROM Produk WHERE Harga <= 30;
6. SELECT * FROM Produk WHERE Harga <> 18;
7. SELECT * FROM Produk WHERE Harga BETWEEN 40 AND 50;
8. SELECT * FROM Pelanggan WHERE Kota LIKE ‘c%’;
9. SELECT * FROM Pelanggan WHERE Kota IN (‘Jakarta’,’Indonesia’);

10.4.4 Operator SQL And, Or dan Not

Sintaks WHERE bisa kita implementasikan juga menggunakan operator And, Or, serta Not. Berdasarkan pada lebih dari satu keadaan, kita bisa menggunakan operator And serta Or untuk memfilter *record* :

1. Operator And menunjukkan *record* bila seluruh keadaan yang dibedakan oleh And merupakan True.
2. Operator Or menunjukkan *record* bila hanya salah satu keadaan yang dibedakan oleh Or merupakan True.
3. Operator Not menunjukkan *record* bila keadaan Not True/Tidak Benar

***QUERY* And secara umum**

```
SELECT kolom1, kolom2, ...
FROM nama_tabel
WHERE kondisi1 AND kondisi2 AND kondisi3 ...;
```

***QUERY* Or secara umum**

```
SELECT kolom1, kolom2, ...
FROM nama_tabel
WHERE kondisi1 OR kondisi2 OR kondisi3 ...;
```

***QUERY* Not secara umum**

```
SELECT kolom1, kolom2, ...
FROM nama_tabel
WHERE NOT kondisi;
```

Query dibawah ini merupakan contoh kasus tabel Pelanggan adalah sebagai berikut:

Pernyataan SQL berikut, select seluruh fields dari tabel “Pelanggan” dengan negara “Indonesia” AND kota “Bandung”:

SELECT * FROM Pelanggan

WHERE Negara=’Indonesia’ AND Kota=’Bandung’;

Pernyataan SQL berikut, select seluruh fields dari tabel “Pelanggan” dengan kota “Bandung” OR “Malang”:

SELECT * FROM Pelanggan

WHERE Kota=‘Bandung’ OR Kota=‘Malang’;

SELECT * FROM Pelanggan

WHERE Negara=’Indonesia’ OR Negara=’Malaysia’;

Pernyataan SQL berikut, Select seluruh fields dari tabel “Pelanggan” yang negaranya BUKAN “Indonesia”:

SELECT * FROM Pelanggan

WHERE NOT Negara=’Indonesia’;

Pernyataan SQL berikut, select seluruh fields dari tabel "Pelanggan" dengan negara "Indonesia" AND kota "Bandung":
SELECT * FROM Pelanggan
WHERE Negara='Indonesia' AND Kota='Bandung';

Pernyataan SQL berikut, select seluruh fields dari tabel "Pelanggan" dengan kota "Bandung" OR "Malang":
SELECT * FROM Pelanggan
WHERE Kota='Bandung' OR Kota='Malang';

SELECT * FROM Pelanggan
WHERE Negara='Indonesia' OR Negara='Malaysia';

Pernyataan SQL berikut, Select seluruh fields dari tabel "Pelanggan" yang negaranya BUKAN "Indonesia":
SELECT * FROM Pelanggan
WHERE NOT Negara='Indonesia';

10.4.5 Kombinasi antara And, Or dan Not

Kita dapat juga mengkombinasikan antara operator And, Or serta operator Not. *Statement* SQL dibawah ini, select seluruh fields yang ada pada tabel "Pelanggan" dengan kolom negara harus "Indonesia" And kolom kota harus "Bandung" OR "Malang" (kita perlu menggunakan tanda kurung guna membuat ekspresi yang kompleks):

SELECT * FROM Pelanggan

WHERE Negara='Indonesia' And (Kota='Bandung' Or Kota='Malang');

Pernyataan SQL berikut select seluruh fields dari “Pelanggan” yang negaranya Not “Indonesia” And Not “Malaysia”:

SELECT * FROM Pelanggan

WHERE Not Negara='Indonesia' And Not Negara='Malaysia';

10.4.6 Sintaks SQL ORDER BY

Dalam basis data kita dapat mengurutkan data berdasarkan urutan naik maupun urutan turun, untuk mendapatkan hasil yang ingin di urutkan kita bisa menggunakan SQL atau sintaks ORDER BY. SQL Order By membantu kita untuk mengurutkan *record* pada urutan menaik secara defaultnya. Sedangkan untuk sebaliknya, kita dapat menggunakan kata kunci DESC untuk mengurutkan *record* pada urutan menurun.

Pernyataan SQL dibawah ini, select seluruh Pelanggan pada tabel “Pelanggan”, dan kita ingin mengurutkan data yang ada pada kolom “Negara”, hasilnya nanti akan menampilkan data-data pada kolom Negara yang di urutkan menaik/Ascending, Negara yang akan tampilkan diurutkan dari huruf depan A-Z:

SELECT * FROM Pelanggan

ORDER BY Negara;

Contoh sintaks ORDER BY DESC

Pernyataan SQL dibawah ini, select seluruh pelanggan yang ada pada tabel “Pelanggan”, kemudian data yang ada pada kolom “Negara” diurutkan secara DESCENDING, hasilnya nanti akan menampilkan data-data pada kolom Negara yang di urutkan menurun/Descending, Negara yang akan tampilkan diurutkan dari huruf depan Z-A:

SELECT * FROM Pelanggan ORDER BY Negara DESC;

Contoh Beberapa Kolom ORDER BY

Pernyataan SQL berikut, select seluruh Pelanggan yang ada pada tabel "Pelanggan", kemudian diurutkan berdasarkan kolom "Negara" dan "NamaPelanggan". Ini berarti bahwa data akan terurut berdasarkan Negara, akan tetapi jika beberapa baris memiliki Negara yang sama, data akan diurutkan berdasarkan NamaPelanggan:

SELECT * FROM Pelanggan ORDER BY Negara, NamaPelanggan;

Pernyataan SQL berikut, select seluruh Pelanggan yang ada pada tabel "Pelanggan", kemudian data diurutkan secara menaik berdasarkan kolom "Negara" dan data diurutkan secara menurun berdasarkan kolom "NamaPelanggan":

SELECT * FROM Pelanggan ORDER BY Negara ASC, NamaPelanggan DESC;

10.4.7 SQL SELECT TOP Clause (SQL TOP, LIMIT)

Dalam menentukan jumlah *record* yang akan dikembalikan, kita dapat menggunakan sintaks SELECT TOP. Pada tabel yang cukup besar dengan banyaknya *record* bahkan ribuan *record*, sintak SELECT TOP akan sangat berguna. Pada saat kita mengembalikan beberapa *record* bisa mempengaruhi performa. *Query* dasar nya adalah sebagai berikut :

SELECT nama_kolom

FROM nama_tabel

WHERE kondisi

LIMIT nomor;

Pernyataan SQL berikut menunjukkan contoh yang setara menggunakan klausa LIMIT (untuk MySQL), *query* nya sebagai berikut:

SELECT * FROM Pelanggan

LIMIT 4;

Contoh SQL Top Percent

Pernyataan SQL berikut menunjukkan contoh yang setara menggunakan klausa LIMIT (untuk MySQL), *query* nya sebagai berikut:

SELECT * FROM Pelanggan

WHERE Negara='Indonesia'

LIMIT 4;

10.4.8 Fungsi SQL Min() dan Max()

Untuk mengembalikan nilai terkecil ketika nilai tersebut berasal dari kolom yang kita pilih, dapat menggunakan Fungsi Min().

Sintaks Min()

SELECT Min(nama_kolom)

FROM nama_tabel

WHERE kondisi;

Untuk mengembalikan nilai terbesar ketika nilai tersebut berasal dari kolom yang kita pilih, dapat menggunakan Fungsi Max().

Sintaks Max()

SELECT Max(nama_kolom)

FROM nama_tabel

WHERE kondisi;

Contoh Min(). Sintaks SQL dibawah ini menampilkan harga dari produk yang paling murah:

SELECT Min(Harga) AS HargaPalingMurah

FROM Produk;

Contoh Max(). Sintaks SQL dibawah ini menampilkan harga dari produk yang paling mahal:

SELECT Max(Harga) AS HargaPalingMahal

FROM Produk;

10.4.9 Fungsi SQL Count(), Avg() dan Sum()

Ketika kita ingin mengembalikan jumlah pada baris yang sesuai dengan kondisi yang telah ditentukan, kita dapat menggunakan fungsi count().

Sintaks Count()

SELECT Count(nama_kolom)

FROM nama_tabel WHERE kondisi;

Ketika kita ingin mengembalikan nilai average atau nilai rata-rata pada kolom yang bersifat numerik, kita dapat menggunakan fungsi avg().

Sintaks Avg()

SELECT Avg(nama_kolom)

FROM nama_tabel

WHERE kondisi;

Ketika kita ingin mengembalikan jumlah total pada kolom yang bersifat numerik, kita dapat menggunakan fungsi sum().

Sintaks Sum()

SELECT Sum(nama_kolom)

FROM nama_tabel

WHERE kondisi;

Contoh implementasi *query* yang dapat kita terapkan pada setiap fungsi yang sudah kita bahas sebelumnya:

Sintaks SQL dibawah ini menampilkan jumlah pada kolom produk: **Count() Example** SELECT count(IDProduk) FROM Produk;	Note: Nilai null tidak perlu dihitung
Sintaks SQL dibawah ini menampilkan harga average seluruh produk: **Avg() Example** SELECT Avg(Harga) FROM Produk;	Note : Nilai null dapat kita abaikan.

Sintaks SQL dibawah ini menampilkan jumlah pada kolom "Kuantitas" yang ada di tabel "DetailPesanan": **Sum() Example** SELECT Sum(Kuantitas) FROM DetailPesanan;	Note : Nilai null dapat kita abaikan.

10.4.10 SQL LIKE Operator

Ketika ingin mencari suatu pola yang bisa ditentukan dalam kolom, kita dapat menggunakan operator Like dengan sintaks Where. Terdapat 2 karakter pengganti yang biasa digunakan bersamaan dengan operator Like, diantaranya adalah :

1. (%) karakter persen menunjukkan nol, satu, atau terdiri dari beberapa karakter.
2. (_) karakter garis bawah menunjukkan hanya satu karakter saja.

Karakter (%) persen dan karakter (_) garis bawah dapat kita gunakan secara bersamaan.

Tabel 10.2 terdiri dari contoh-contoh yang menampilkan operator Like yang tidak sama dengan karakter pengganti '%' serta '_':

Tabel 10.2 Tabel Operator Like

LIKE Operator	**Description**
WHERE NamaPelanggan LIKE 'i%'	Menemukan nilai atau data apa pun yang dimulai dengan "i"
WHERE NamaPelanggan LIKE '%i'	Menemukan nilai atau data apa pun yang diakhiri dengan "i"
WHERE NamaPelanggan LIKE '%ah%'	Menemukan nilai atau data apa pun yang memiliki "ah" di posisi apa pun

WHERE NamaPelanggan LIKE '_d%'	Menemukan nilai atau data apa pun yang memiliki "d" di posisi kedua
WHERE NamaPelanggan LIKE 'i_%'	Menemukan nilai atau data apa pun yang dimulai dengan "i" dan panjangnya minimal 2 karakter.
WHERE NamaPelanggan LIKE 'i__%'	Menemukan nilai atau data apa pun yang dimulai dengan "i" dan panjangnya minimal 3 karakter.
WHERE NamaPelanggan LIKE 'i%e'	Menemukan nilai atau data apa pun yang dimulai dengan "i" dan diakhiri dengan "e".

Untuk contoh *query* dari masing-masing operator LIKE adalah sebagai berikut: *Statement* SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya berawal dengan huruf "i":

SELECT * FROM Pelanggan

WHERE NamaPelanggan LIKE 'i%';

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya berakhir dengan huruf "i":

SELECT * FROM Pelanggan

WHERE NamaPelanggan LIKE '%i';

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya terdapat huruf "ah" di posisi mana pun, baik posisi awal, tengah, maupun akhir:

SELECT * FROM Pelanggan

WHERE NamaPelanggan LIKE '%ah%';

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya terdapat huruf "d" di posisi nomor dua:

```
SELECT * FROM Pelanggan
WHERE NamaPelanggan LIKE '_d%';
```

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya berawal dengan huruf "i" serta panjang karakternya minimal 3:

```
SELECT * FROM Pelanggan
WHERE NamaPelanggan LIKE 'i__%';
```

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya berawal dengan huruf "i" serta berakhir dengan huruf "e":

```
SELECT * FROM Pelanggan
WHERE NamaPelanggan LIKE 'i%e';
```

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang NamaPelanggan nya **tidak** berawal dengan huruf "i"

```
SELECT * FROM Pelanggan
WHERE NamaPelanggan NOT LIKE 'i%';
```

10.4.11 Operator SQL IN

Operator IN memungkinkan kita dalam memilih sebagian nilai pada sintaks WHERE. Operator IN merupakan singkatan dari sebagian keadaan OR. *Query* secara umumnya merupakan sebagai berikut:

SELECT nama_kolom

FROM nama_tabel

WHERE nama_kolom IN (value1, value2, ...);

Sintak yang lainnya adalah :

SELECT nama_kolom

FROM nama_tabel

WHERE nama_kolom IN (*Select Statement*);

Untuk implementasi *query* nya adalah sebagai berikut:

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang sedang menetap di “Jerman”, “Francis” or “United Kingdom”:

SELECT * FROM Pelanggan

WHERE Negara IN (‘Jerman’, ‘Francis’, ‘United Kingdom’);

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang sedang TIDAK menetap di “Jerman”, “Francis” or “United Kingdom”:

SELECT * FROM Pelanggan

WHERE Negara NOT IN (‘Jerman’, ‘Francis’, ‘United Kingdom’);

Statement SQL dibawah ini menampilkan seluruh Pelanggan yang memiliki asal negara yang sama dengan Kasir:

SELECT * FROM Pelanggan

WHERE Negara IN (SELECT Negara FROM Kasir);

10.4.12 Operator SQL Between

Untuk dapat menentukan nilai sesuai rentang nilai yang kita inginkan, kita bisa menggunakan operator BETWEEN. Rentang nilai yang kita ingin tampilkan dapat berupa teks, angka atau dapat juga berupa tanggal. Operator BETWEEN ini memiliki sifat inklusif, yang artinya nilai awal dan juga nilai akhir harus kita tuliskan:

SELECT nama_kolom

FROM nama_tabel

WHERE nama_kolom BETWEEN value1 AND value2;

Untuk implementasi *query* yang kita terapkan didalam tabel adalah sebagai berikut:

BETWEEN Example	**NOT BETWEEN Example**
Pernyataan SQL berikut menampilkan semua produk den-gan harga BETWEEN 20 dan 30: SELECT * FROM Produk WHERE Harga BETWEEN 20 AND 30;	Untuk menampilkan produk di luar rentang contoh ditabel kiri, gunakan NOT BETWEEN: SELECT * FROM Produk WHERE Harga NOT BETWEEN 20 AND 30;

Contoh-contoh *query* Between lainnya sebagai berikut:

Contoh BETWEEN dengan IN

Statement SQL dibawah ini menampilkan seluruh produk

yang harganya antara/BETWEEN 20 dan 30. Setelah itu, data produk yang IDKategori nya 1, 2, or 3 jangan ditampilkan:

SELECT * FROM Produk

WHERE Harga BETWEEN 20 AND 30

AND IDKategori NOT IN (1,2,3);

Contoh BETWEEN Text Values

Statement SQL dibawah ini menampilkan seluruh produk yang NamaProduk nya antara/ BETWEEN Shampo Sunsilk dan Scarlett Whitening:

SELECT * FROM Produk

WHERE NamaProduk BETWEEN 'Shampo Sunsilk' AND 'Scarlett Whitening'

ORDER BY NamaProduk;

Statement SQL dibawah ini menampilkan seluruh produk yang NamaProduk nya antara/ BETWEEN Shampo Sunsilk dan Lux Sakura Bloom:

SELECT * FROM Produk

WHERE NamaProduk BETWEEN "Shampo Sunsilk" AND "LUX Sakura Bloom"

ORDER BY NamaProduk;

Contoh NOT BETWEEN Text Values

Statement SQL dibawah ini menampilkan seluruh produk yang NamaProduk nya bukan diantara/NOT BETWEEN Shampo Sunsilk dan Scarlett Whitening:

SELECT * FROM Produk

WHERE NamaProduk NOT BETWEEN 'Shampo Sunsilk' AND 'Scarlett Whitening'

ORDER BY NamaProduk;

Contoh BETWEEN Dates

Statement SQL dibawah ini menampilkan seluruh order yang TanggalPemesanan nya diantara/ BETWEEN '01-Februari-2022' and '18-Oktober-2022':

SELECT * FROM Order

WHERE TanggalPemesanan BETWEEN #01/02/2022# AND #18/10/2022#;

ATAU

SELECT * FROM Order
WHERE TanggalPemesanan BETWEEN '2022-02-01'
AND '2022-10-18';

10.4.13 Operator SQL EXISTS

Ketika kita ingin menguji eksistensi suatu *record* apapun pada sub *query*, kita dapat menggunakan operator Exists. Operator Exists berfungsi mengembalikan nilai true/benar ketika sub *query* mengembalikan satu atau lebih *record*. *Query* umumnya adalah sebagai berikut:

SELECT nama_kolom

FROM nama_tabel

WHERE EXISTS

(SELECT nama_kolom FROM nama_tabel WHERE kondisi);

Contoh SQL EXISTS

Statement SQL dibawah ini mengembalikan nilai **true** dan mencantumkan Pemasok dengan harga produk kurang dari 30:

SELECT NamaPemasok

FROM Pemasok

WHERE EXISTS (SELECT NamaProduk FROM Produk

WHERE Produk.IDPemasok = Pemasok. IDPemasok AND Harga < 30);

Statement SQL dibawah ini mengembalikan nilai **true** dan menampilkan Pemasok serta harga produk yang sama dengan 28:

SELECT NamaPemasok

FROM Pemasok

WHERE EXISTS (SELECT NamaProduk FROM Produk WHERE Produk.IDPemasok = Pemasok. IDPemasok AND Harga = 28);

10.4.14 Operator SQL ANY and ALL

Ketika kita ingin menggunakan operator Any dan juga operator All harus bersamaan dengan sintaks Where ataupun Having. Operator Any berfungsi mengembalikan nilai true bila salah **satu** nilai sub queri memenuhi syarat/kondisi. Kemudian untuk operator All berfungsi mengembalikan nilai true bila **seluruh** nilai sub queri memenuhi syarat/ kondisi. Untuk *query* secara umum nya sebagai berikut :

Sintaks ANY	**Sintaks ALL**
SELECT nama_kolom FROM nama_tabel WHERE nama_kolom **operator ANY** (SELECT nama_kolom FROM nama_tabel WHERE kondisi);	SELECT nama_kolom FROM nama_tabel WHERE nama_kolom **operator ALL** (SELECT nama_kolom FROM nama_tabel WHERE kondisi);

Contoh SQL ANY

Operator Any berfungsi mengembalikan nilai true bila salah satu nilai sub queri memenuhi syarat/ kondisi. *Statement* SQL dibawah ini mengembalikan nilai **true** dan menampilkan NamaProduk bila mendeteksi *record* apapun yang berada pada tabel DetailPesanan yang mana jumlah = 20:

```
SELECT NamaProduk
FROM Produk
WHERE IDProduk = ANY (SELECT IDProduk
FROM DetailPesanan WHERE Kuantitas = 20);
```

Statement SQL dibawah ini mengembalikan nilai **true** dan menampilkan NamaProduk bila mendeteksi *record* apapun yang berada pada tabel DetailPesanan yang mana kuantitas > 89:

```
SELECT NamaProduk
FROM Produk
```

```
WHERE IDProduk = ANY (SELECT IDProduk
FROM DetailPesanan WHERE Kuantitas > 89);
```

Contoh SQL ALL

Operator All berfungsi mengembalikan nilai true bila seluruh nilai sub queri memenuhi syarat/ kondisi.

Statement SQL dibawah ini mengembalikan nilai **true** dan menampilkan NamaProduk bila **seluruh** *record* yang berada pada tabel DetailPesanan memiliki kuantitas = 20 (contoh sintaks dibawah ini nantinya akan mengembalikan nilai False, karena tidak seluruh *record* yang berada pada tabel DetailPesanan memiliki kuantitas = 20):

```
SELECT NamaProduk
FROM Produk
WHERE IDProduk = ALL (SELECT IDProduk
FROM DetailPesanan WHERE Kuantitas = 20);
```

10.4.15 SQL Aliases

Ketika kita ingin memberikan nama sementara untuk tabel atau kolom yang ada di dalam tabel, kita dapat menggunakan SQL Aliases. Kita perlu menggunakan SQL Aliases untuk membentuk nama kolom agar lebih mudah dibaca. Alias hanya ada selama adanya kueri. *Query* secara umumnya adalah sebagai berikut:

Sintaks Alias Column	*Sintaks Alias Table*
SELECT nama_kolom AS nama_ alias FROM nama_tabel;	SELECT nama_kolom FROM nama_tabel AS nama_alias;

Contoh *Query* Alias untuk Kolom

Statement SQL dibawah ini menampilkan seluruh pesanan Pelanggan dengan IDPelanggan nya = 3 (Nuning Widya). Untuk contoh ini, kita memakai tabel "Pelanggan" dan "Pesanan", dan memberikan nama alias pada masing-masing tabel. Tabel Pelanggan dengan nama alias "p" dan tabel Pesanan dengan nama alias "pe" (Pada contoh kali ini, kita membentuk alias agar SQL yang kita buat jadi lebih ringkas):

SELECT pe.IDPesanan, pe.TanggalPemesanan, p.NamaPelanggan
FROM Pelanggan AS p, Pesanan AS pe
WHERE p.NamaPelanggan='Nuning Widya' AND p.IDPelanggan=pe.IDPelanggan;

Sintaks *query* SQL dibawah ini sama seperti sintaks SQL sebelumnya, namun tanpa menggunakan alias:

SELECT Pesanan.IDPesanan, Pesanan. TanggalPemesanan, Pelanggan.NamaPelanggan
FROM Pelanggan, Pesanan
WHERE Pelanggan.NamaPelanggan ='Nuning Widya' AND Pelanggan.IDPelanggan = Pesanan. IDPelanggan;

BAB XI

MySQL

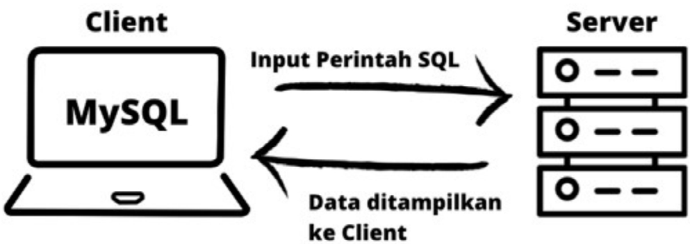
Pendahuluan

Gambar 11.1 Logo MySQL

MySQL sebagai salah satu sistem mengelola basis data relasional yang berbasis SQL (RDBMS) bersifat *open source* yang beroperasi dalam model sisi klien (Pengguna). MySQL bersifat *server* basis data gratis di bawah Lisensi Publik Umum GNU (GPL) yang dapat digunakan untuk keperluan pribadi atau komersial tanpa membayar lisensi.

Pengembang pertama MySQL adalah MySQL AB Company of Sweden yang memulai perjalanannya pada tahun 1994. Padan tahun 2008 perusahaan Sun Microsystem yang berasal dari Amerika telah mengakuisisi MySQL AB secara penuh. Kemudian pada tahun 2010 Oracle mengakuisisi Sun Microsystems perusahaan teknologi terbesar di Amerika. Sejak saat itu MySQL dimiliki sepenuhnya oleh Oracle.

11.1 Cara Kerja MySQL



Gambar 11.2 Cara Kerja MySQL

Gambar di atas mengilustrasikan bahwa *client* (pengguna) berinteraksi dengan antarmuka/ *Graphic User Interface* (GUI) dam melakukan perintah/permintaan SQL ke *server* melalui jaringan internet maupun intranet. Kemudian server akan menampilkan data ke client sesuai dengan permintaan. Hasil output akan benar jika client dan server memahami perintah dengan baik. Berikut beberapa Proses MySQL :

1. MySQL dapat menyimpan serta memanipulasi data kemudian dapat mendefinisikan hubungan setiap tabel.
2. *Client* memberiukan perintah kueri (*queries*)

ke MySQL dengan menulis pernyataan SQL tertentu.

3. Server akan menjawab request dari client dengan memberikan informasi dan menampilkannya ke client.

Tentukan GUI mana yang akan digunakan di sisi klien. GUI yang sederhana dan intuitif membuat tugas manajemen data cepat dan mudah. Beberapa GUI MySQL yang paling populer adalah MySQL WorkBench, PHPMyAdmin, dbForge dan dBeaver

Terdapat beberapa GUI MySQL dapat beroperasi di berbagai macam operating system baik yang gratis maupun berbayar. Memilih GUI biasanya berdasarkan kebutuhan dan preferensi masing-masing pengguna. GUI terbaik dan yang banyak dipakai untuk mengelola database adalah phpMyAdmin.

11.2 Kelebihan MySQL

399 systems in ranking, December 2022

Rank			DBMS	Database Model	Score		
Dec 2022	Nov 2022	Dec 2021			Dec 2022	Nov 2022	Dec 2021
1.	1.	1.	Oracle	Relational, Multi-model	1250.31	+8.62	-31.43
2.	2.	2.	MySQL	Relational, Multi-model	1199.40	-6.14	-6.64
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	924.35	+11.84	-29.67
4.	4.	4.	PostgreSQL	Relational, Multi-model	617.97	-5.18	+9.76
5.	5.	5.	MongoDB	Document, Multi-model	469.33	-8.57	-15.34
6.	6.	6.	Redis	Key-value, Multi-model	182.57	+0.52	+9.03
7.	↑ 8.	7.	IBM Db2	Relational, Multi-model	146.61	-2.95	-20.56
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model	144.93	-5.40	-12.80
9.	9.	↑ 10.	Microsoft Access	Relational	133.83	-1.20	+7.84
10.	10.	↓ 9.	SQLite	Relational	132.44	-2.19	+3.76

Gambar 11.3 Ranking MySQL Per Desember 2022

(Sumber: https://db-engines.com/)

MySQL tidak hanya dikenal sebagai (R)

DBMS di pasar tetapi juga merupakan salah satu database paling populer setelah Oracle Database ketika dievaluasi berdasarkan kriteria utama seperti menggunakan pencarian profil di LinkedIn. Seberapa sering muncul dalam hasil dan jumlah teknik yang dibahas. Banyak perusahaan teknologi menggunakan perangkat lunak untuk semakin memperkuat kehadiran mereka di forum online. Berikut adalah beberapa alasan mengapa MySQL banyak digunakan:

1. *Easy to Use*

Source Code (Kode Sumber) dapat diubah sesuai kebutuhan tanpa keterbatasan untuk *upgrade* ke versi premium berbayar. Proses instalasi relatif mudah dan singkat.

2. *Good Performance*

Terdapat banyak pilihan server yang dapat mendukung MySQL. Dengan kinerja yang baik MySQL mampu membantu menyimpan data dalam jumlah besar dan melakukan tugas lainnya.

3. *Lightweight*

MySQL dapat diinstal pada *server* dengan spesifikasi lebih rendah. Jadi jika *server* hanya memiliki *storage* 1GB, masih bisa menggunakan MySQL sebagai *database* dan tidak perlu khawatir.

4. *Multiple User Support*

Dapat mendukung banyak pengguna secara bersamaan menggunakan MySQL tanpa masalah serta dapat digunakan saat mengerjakan proyek secara tim sehingga seluruh tim dapat bekerja sama tanpa menunggu orang lain selesai.

5. *Integration*

Situs web dan program dapat dikembangkan menggunakan bahasa pemrograman yang berbeda sehingga tidak perlu khawatir menggunakan MySQL. Dengan begitu MySQL membantu pengembangan perangkat lunak yang lebih efisien dan benar-benar sederhana melalui integrasi antar bahasa pemrograman.

6. *Industry Standard*

Telah banyak digunakan oleh industri menggunakan sistem manajemen *database* selama bertahun-tahun dan banyak sumber daya telah ditangani oleh *developer* ahli dan memastikan bahwa perangkat lunak MySQL yang diperbarui secara berkala.

7. *Security*

Keamanan data adalah prioritas utama untuk perangkat lunak RDBMS. MySQL menetapkan tingkat keamanan yang tinggi dengan sistem hak akses dan manajemen pengguna. Otentikasi berbasis *host* dan enkripsi kata sandi juga dimungkinkan.

Fitur-fitur MySQL Meliputi

1. Sistem *database* relasional RDBMS
2. berarsitektur *client-server* yang mana sisi *server* dapat diinstal pada komputer dan *client* bisa berada pada komputer yang sama atau di komputer lainnya yang terhubung dengan jaringan internet maupun intranet.
3. Berbasis SQL (*Structured Query Language*)
4. Dukungan untuk subpilihan.
5. Dukungan pilihan (subpilihan) di dalam opsi dan mendukung tampilan.
6. Dukungan replikasi.
7. Dukungan transaksi.
8. Mendukung kunci asing (*Foreign Key)*.
9. *Function* dari *Geographic Information System* disediakan
10. Gratis di instal dan digunakan
11. Berjalan baik serta dapat digunakan di berbagai *platform*
12. Tingkat keamanan yang mumpuni
13. Pengembangan perangkat lunak cepat.

11.3 Jenis tabel MySQL

MySQL dapat mendukung berbagai jenis tabel yang bergantung pada pengaturan pada saat instalasi MySQL. MySQL memiliki 3 (tiga) jenis tipe data yaitu: MyISAM InnoDB dan HEAP.

apabila tidak ada jenis tabel yang ditentukan pada saat membuat tabel, maka jenis tabel dibuat secara otomatis sesuai dengan pengaturan *default* server MySQL. Pengaturan otomatis ini ditentukan oleh variabel default-table-type di file *config* MySQL.

MyISAM

Merupakan Jenis tabel statis sederhana yang mudah digunakan. menggunakan tabel jenis ini untuk menyimpan data yang sederhana. Kecepatan menjadi kelebihan jenis tabel MyISAM ini. Pada saat memilih jenis tabel ini maka secara otomatis MySQL memilih salah satu tabel MyISAM:

a. ***Static***. Gunakan saat semua kolom dalam tabel berukuran tetap. Pastikan tidak ada tipe data *Varchar*, *Text*, dan *Blob*. Tipe ini lebih cepat dan stabil karena karakternya statis.

b. ***Dynamic***. Gunakan tipe data *Varchar* yang dinamis saat memiliki kolom. Efisiensi ukuran data (*file*) dari ruang isi kolom yang dinamis merupakan ciri khas dari tipe ini.

c. ***Compressed***. Perpaduan statis dan dinamis menjadi satu tipe sehingga ukuran menjadi lebih kecil. Tipe ini tidak bisa dilakukan

INSERT, UPDATE dan *DELETE karena* terkompresinya tabel.

InnoDB

Keunggulan dari InnoDB yang dapat mendukung proses transaksi. yaitu:

a. *Table Transaction Support*.
b. *Row Level Locking Support*.
c. *Foreign Key Constrains Support*.
d. *Crash Recovery*.

Heap

Tabel Heap memiiki karakter tidak menyimpan data pada *hard disk* melainkan pada memori (RAM). Tabel Heap biasanya digunakan untuk jadwal. jika koneksi *server* terputus atau mati maka file-file dihapus dari MySQL.

Jenis Tabel Yang Lain

Selain MyISAM, InnoDB dan HEAP yang disebutkan diatas, terdapat jenis tabel lainnya dibawah ini:

1. **BDB**. Karakternya mirip InnoDB, namun belum maksimal.
2. **Archieve**. Digunakan untuk proses backup dengan menyimpan tabel yang terkompres.

3. **CSV**. Diperuntukan menyimpan dalam format data yang dipisahkan dengan tanda/ titik koma (,).
4. **NDBTable** (MySQLCluster)
5. **Federated** (ExternalTables)

Tipe-tipe *Field* (Kolom) MySQL

Ada beberapa tipe data untuk bidang tabel MySQL (kolom). Tipe kolom ini menjadi tolak ukur pada ukuran tabel basis data. Tipe kolom pada MySQL dibagi kedalam grup-grup seperti tanggal dan waktu string numerik dan kumpulan grup (set dan hitungan). Setiap tipe kolom tentunya memiliki batas *range* dan *size*.

Tipe Angka atau *Numeric*

Menyimpan data angka atau numerik. Ciri khas dari data numerik adalah bahwa data tersebut memungkinkannya untuk melakukan operasi perhitungan seperti penjumlahan, pengurangan, perkalian dan pembagian. Berikut Tipe *field* (kolom) di MySQL yang termasuk tipe numerik:

1. ***TINYINT***

Function	:	Simpan data bilangan bulat *positive* dan *negative*.
Range	:	-128 hingga 127
Size	:	1 *byte* (8 bit).

2. *SMALLINT*

Function : Simpan data bilangan bulat *positive* dan *negative.*

Range : -32768 hingga 32767

Size : 2 *bytes* (16 bit).

3. *MEDIUMINT*

Function : Simpan data bilangan bulat *positive* dan *negative*

Range : -8388608 hingga 8388607

Size : *3 bytes* (24 bit)

4. *INT*

Function : Simpan data bilangan bulat *positive* dan *negative*

Range : -2.147483648 hingga 2147483647

Size : *4 bytes* (32 bit)

5. *BIGINT*

Function : Simpan data bilangan bulat *positive* dan *negative*

Range : $\pm 9{,}22 \times 10^{18}$

Size : 8 *bytes* (64 bit)

6. *FLOAT*

Function : Simpan data bilangan pecahan *positive* dan *negative* yang presisi tunggal.

Range : -3.402823466E+38 hingga -1.175494351E-38, 0, dan 1.175494351E-38 hingga 3.402823466E+38.

Size : 4 *bytes* (32 bit)

7. *DOUBLE*

Function : Simpan data bilangan pecahan *positive* dan *negative* yang presisi ketelitian ganda.

Range : -1.79 . . . E+308 hingga -2.22 . . . E-308, 0, dan 2.22 . . . E-308 hingga 1.79 . . . E+308.

Size : 8 *bytes* (64 bit)

8. *REAL*

Memiliki fungsi yang sama dengan tipe data DOUBLE

9. *DECIMAL*

Function : menyimpan data bilangan pecahan *positive* dan *negative*.

Range : -1.79 . . . E+308 hingga -2.22 . . . E-308, 0, dan 2.22 . . . E-308 hingga 1.79 . . . E+308.

Size : **8** *bytes* (64 bit)

10. NUMERIC

Memiliki fungsi yang sama dengan tipe data DECIMAL

Tipe *Date* dan *Time*

Dapat menyimpan data tanggal dan waktu. Berikut dibawah ini yang termasuk tipe data *date* dan *time*:

11. ***DATE***

Function : Simpan data tanggal

Range : 1000-01-01 hingga 9999-12-31 (YYYY-MM-DD)

Size : *3 bytes*

12. ***TIME***

Function : Simpan data waktu

Range : -838:59:59 hingga +838:59:59 (HH:MM:SS)

Size : *3 bytes*

13. *DATETIME*

Function : Simpan tanggal dan waktu

Range : 1000-01-01 00:00:00 hingga

9999-12-31 23:59:59

Size : 8 *bytes*

14. *YEAR*

Function : Simpan data tahun dari tanggal

Range : 1900 hingga 2155

Size : 1 *byte*

Tipe *String*

Dapat menyimpan data teks dan mampu menampung banyak data gabungan huruf, angka dan karakter. Berikut dibawah ini yang termasuk tipe string:

15. ***CHAR***

Function : Simpan data *string* tetap

Range : 0 hingga 255 karakter

16. ***VARCHAR***

Function : Simpan data *string* dinamis

Range : 0 hingga 65535 karakter

17. ***TINYTEXT***

Function : Simpan data *text*

Range : 0 hingga 65535 karakter

18. ***TEXT***

Function : Simpan data *text*

Range : 0 hingga 65535 (2^{16} - 1) karakter

19. ***MEDIUMTEXT***

Function : Simpan data *text*

Range : 0 hingga 2^{24} - 1 karakter

20. LONGTEXT

Function : Simpan data *text*

Range : 0 hingga 2^{32} - 1 karakter

Tipe *BLOB*

Menyimpan kode binary dari suatu file atau objek. Berikut dibawah ini kelompok tipe blob:

21. *BIT*

Function : Simpan data *binary*

Range : 64 angka *binary*

22. *TINYBLOB*

Function : Simpan data *binary*

Range : 255 *bytes*

23. *BLOB*

Function : Simpan data *binary*

Range : 2^{16} hingga 1 byte

24. *MEDIUMBLOB*

Function : Simpan data *binary*

Range : 2^{24} hingga 1 *byte*

25. *LONGBLOB*

Function : Simpan data *binary*

Range : 2^{32} hingga 1 *byte*

Lainnya

26. *ENUM*

Function : Simpan kumpulan data (enumerasi)

Range : Hingga 65535 *string*

27. *SET*

Function : Simpan himpunan data (kombinasi)

Range : Hingga 255 *string* anggota

11.4 Merancang *Database*

Desain *database* menentukan apakah suatu aplikasi efektif atau tidak. Dengan beberapa aturan merancang *database* tentunya suatu aplikasi menjadi lebih optimal dan efektif.

Aturan merancang *database*:

1. Hindari data *redundancy* (ganda/duplikat) pada setiap table *database*
2. *Field* (kolom) harus bersifat unik pada setiap tabel database. *Field* ini yang akan menjadi *primary key*
3. Hindari tabel tidak normal (*unnormalized*).
4. Gunakan tipe data yang tepat agar ukuran *database* sekecil mungkin.
5. Pastikan rancangan database mampu me-*record* data dan sesuai kebutuhan aplikasi.

Cara penamaan database:

1. Konsistensi penamaan dan perhatian dalam merancang *database* karena memiliki sifat *case-sensitive*.
2. Maksimal 64 karakter dalam penamaan *database*, tabel dan kolom.
3. Tidak disarankan menggunakan karakter atau simbol khusus.

Penamaan *field* (kolom) harus sesuai dengan isi data yang akan disimpan.

BAB

XII

Fungsi di MySQL

Pendahuluan

Fungsi dapat diartikan sebagai kumpulan beberapa instruksi yang digunakan untuk melakukan suatu proses tertentu dan kemudian mengembalikan hasilnya. Fungsi pada umumnya memerlukan parameter, yang nilainya (argumen) menjadi input untuk diproses ketika fungsi tersebut dipanggil, tetapi ada juga fungsi yang tidak memerlukan parameter. Fungsi dapat digunakan untuk menggantikan instruksi-instruksi di tingkat aplikasi atau digunakan secara bersama dengan instruksi-instruksi lainnya di dalam aplikasi.

MySQL tidak hanya dapat menyimpan dan mengambil data, tetapi MySQL dapat melakukan manipulasi data sebelum menyimpan atau mengambilnya. MySQL dapat melakukan manipulasi data dengan menggunakan fungsi. Penggunaan fungsi pada MySQL akan memberikan kemudahan bagi *user* dalam mengolah data.

Jenis-jenis fungsi pada MySQL dapat berupa :

1. *Built in Function* yaitu fungsi bawaan yang disertakan dalam MySQL, atau fungsi yang

sudah disediakan oleh MySQL.

2. *User-Defined Function* atau *Stored* Procedure / *Stored Function* yaitu fungsi yang dibuat oleh user di dalam server MySQL atau menggunakan bahasa pemrograman, seperti C, C++, dan kemudian ditambahkan ke server MySQL, yang kemudian dapat digunakan dalam pernyataan SQL

Dalam buku ini akan dibahas beberapa fungsi yang disediakan MySQL (*built in function*) yang umumnya digunakan dalam pengolahan basis data.

Fungsi-fungsi bawaan yang disertakan (*built-in function*) pada MySQL dapat dikelompokkan menjadi :

1. Fungsi numerik yaitu fungsi yang dapat digunakan untuk mengolah tipe data numerik.
2. Fungsi *string*/teks yaitu fungsi yang digunakan untuk memanipulasi tipe data *string*/teks.
3. Fungsi waktu dan tanggal yaitu fungsi yang dapat digunakan untuk mengolah tipe data waktu (*time*) dan tanggal (*date*)
4. Fungsi tambahan lainnya

12.1 Fungsi Numerik

Fungsi numerik dapat dioperasikan pada data-data yang bertipe numerik. Fungsi ini memerlukan argumen berupa data numerik, dan akan mengembalikan nilai yang berupa data numerik.

Fungsi-fungsi yang termasuk dalam fungsi numerik, antara lain :

1. **ABS (x)**

   Fungsi ini akan menghasilkan nilai absolut dari nilai x.

   Contoh :

```
MariaDB [(none)]> select abs (32), abs (-32);
+----------+-----------+
| abs (32) | abs (-32) |
+----------+-----------+
|       32 |        32 |
+----------+-----------+
```

2. **CEILING (x) atau CEIL (x)**

   Fungsi ini akan menghasilkan nilai bilangan bulat terkecil yang tidak kurang (lebih kecil) dari nilai x.

   Contoh :

```
MariaDB [(none)]> select ceil (1.34), ceil (-1.34);
+-------------+--------------+
| ceil (1.34) | ceil (-1.34) |
+-------------+--------------+
|           2 |           -1 |
+-------------+--------------+
```

3. **FLOOR (x)**

   Fungsi ini akan menghasilkan nilai bilangan bulat terbesar yang tidak melebihi (lebih besar) dari nilai x.

   Contoh :

```
MariaDB [(none)]> select floor (1.34), floor (-1.34);
+--------------+---------------+
| floor (1.34) | floor (-1.34) |
+--------------+---------------+
|            1 |            -2 |
+--------------+---------------+
```

4. ROUND (x) atau ROUND (x , d)

Fungsi ini akan menghasilkan nilai pembulatan dari nilai x dengan presisi sebanyak d digit tempat desimal. Jika d bernilai negatif, maka d digit nilai x disebelah kiri koma desimal, akan menjadi nol (0). Besarnya nilai d antara -30 sampai dengan 30.

Contoh :

```
MariaDB [(none)]> select round (125.345,1) , round (125.345,2);
+-------------------+-------------------+
| round (125.345,1) | round (125.345,2) |
+-------------------+-------------------+
|             125.3 |            125.35 |
+-------------------+-------------------+
```

```
MariaDB [(none)]> select round (125.345,-1) , round (125.345,-2);
+--------------------+--------------------+
| round (125.345,-1) | round (125.345,-2) |
+--------------------+--------------------+
|                130 |                100 |
+--------------------+--------------------+
```

5. TRUNCATE (x , d)

Fungsi ini akan menghasilkan nilai x yang terpotong dengan d digit tempat desimal. Jika nilai d = 0, maka nilai desimal (bagian pecahan) dari nilai x akan dihilangkan. Jika d bernilai negatif, maka d digit nilai x disebelah kiri koma desimal, akan menjadi nol (0)

Contoh :

```
MariaDB [(none)]> select truncate (125.345 , 1) ,
    -> truncate (125.345 , 2);
+-----------------------+-----------------------+
| truncate (125.345 , 1) | truncate (125.345 , 2) |
+-----------------------+-----------------------+
|                 125.3 |                125.34 |
+-----------------------+-----------------------+
```

```
MariaDB [(none)]> select truncate (125.345 , -1) ,
    -> truncate (125.345 , -2);
+------------------------+------------------------+
| truncate (125.345 , -1) | truncate (125.345 , -2) |
+------------------------+------------------------+
|                    120 |                    100 |
+------------------------+------------------------+
```

Perbedaan hasil dari fungsi Round dan Truncate

```
MariaDB [(none)]> select round (125 , -1) , truncate (125 , -1);
+-----------------+--------------------+
| round (125 , -1) | truncate (125 , -1) |
+-----------------+--------------------+
|             130 |                120 |
+-----------------+--------------------+
```

6. POWER (x , n) atau POW (x , n)

Fungsi ini akan menghasilkan nilai x pangkat n (x^n).

Contoh :

```
MariaDB [(none)]> select pow (2,3), pow (2,-3);
+-----------+------------+
| pow (2,3) | pow (2,-3) |
+-----------+------------+
|         8 |      0.125 |
+-----------+------------+
```

7. SQRT (x)

Fungsi ini akan menghasilkan nilai akar pangkat dua dari nilai x (jika nilai x >= 0). Jika x bernilai negatif, maka fungsi akan menghasilkan NULL.

Contoh :

```
MariaDB [(none)]> select sqrt(9) , sqrt(-9);
+---------+----------+
| sqrt(9) | sqrt(-9) |
+---------+----------+
|       3 |     NULL |
+---------+----------+
```

8. RAND () dan RAND (x)

Fungsi ini akan menghasilkan nilai pecahan (*floating point*) acak antara 0 sampai kurang dari dari 1. Jika fungsi diberikan parameter x, maka fungsi tersebut akan menghasilkan nilai pecahan yang sama apabila fungsi dipanggil kembali.

Contoh :

```
MariaDB [(none)]> select rand() , rand(3);
+--------------------+--------------------+
| rand()             | rand(3)            |
+--------------------+--------------------+
| 0.7682530607732507 | 0.9057697559760601 |
+--------------------+--------------------+
1 row in set (0.000 sec)

MariaDB [(none)]> select rand() , rand(3);
+---------------------+--------------------+
| rand()              | rand(3)            |
+---------------------+--------------------+
| 0.07105648151697264 | 0.9057697559760601 |
+---------------------+--------------------+
```

Fungsi numerik lain yang biasanya digunakan untuk mengolah data numerik, adalah fungsi agregasi. Fungsi ini memiliki parameter dengan argumen berupa kumpulan (range) data numerik yang berasal dari suatu field / atribut dari sebuah tabel.

Untuk memberikan contoh fungsi-fungsi yang merupakan fungsi agregasi, digunakan tabel mahasiswa, sbb:

```
MariaDB [test]> select * from mahasiswa;
+------+-----------------+-------+
| nim  | nama            | nilai |
+------+-----------------+-------+
| 1234 | Alif Perdana    |    95 |
| 1235 | Budi Pratama    |    65 |
| 1236 | Citra Utami     |    85 |
| 1237 | Desy Fridasari  |    70 |
| 1238 | Eka Suryana     |    75 |
| 1239 | Fitria Harumsari|    90 |
| 1240 | Gita Swara      |  NULL |
+------+-----------------+-------+
7 rows in set (0.000 sec)
```

9. MAX (range)

Fungsi ini menghasilkan nilai terbesar dari kumpulan nilai (range). Parameter yang digunakan adalah nama field yang berisi kumpulan nilai (range) yang akan diproses.

Contoh :

Nilai terbesar dari field nilai pada tabel Mahasiswa

```
MariaDB [test]> select max(nilai) from mahasiswa;
+------------+
| max(nilai) |
+------------+
|         95 |
+------------+
```

10. MIN (range)

Fungsi ini menghasilkan nilai terkecil dari kumpulan nilai (range). Parameter yang digunakan adalah nama field yang berisi kumpulan nilai (range) yang akan diproses.

Contoh :

Nilai terkecil dari field nilai pada tabel Mahasiswa

```
MariaDB [test]> select min(nilai) from mahasiswa;
+------------+
| min(nilai) |
+------------+
|         65 |
+------------+
```

11. SUM (range)

Fungsi ini menghasilkan nilai total atau jumlah dari kumpulan nilai (range). Parameter yang digunakan adalah nama field yang berisi kumpulan nilai (range) yang akan diproses.

Contoh :

Nilai total dari field nilai pada tabel Mahasiswa

```
MariaDB [test]> select sum(nilai) from mahasiswa;
+------------+
| sum(nilai) |
+------------+
|        480 |
+------------+
```

12. AVG (range)

Fungsi ini menghasilkan nilai rata-rata dari kumpulan nilai (range). Parameter yang digunakan adalah nama field yang berisi kumpulan nilai (range) yang akan diproses.

Contoh :

Nilai rata-rata dari field nilai pada tabel Mahasiswa

```
MariaDB [test]> select avg(nilai) from mahasiswa;
+------------+
| avg(nilai) |
+------------+
|    80.0000 |
+------------+
```

13. COUNT (*) atau COUNT (range)

Fungsi ini menghasilkan banyaknya record pada sebuah tabel. Parameter yang digunakan adalah nama field yang berisi kumpulan nilai (range) yang akan diproses. Jika parameter nama field disertakan dalam fungsi, maka fungsi ini akan menghasilkan banyaknya record pada field tersebut, yang tidak bernilai NULL

Contoh :

Banyaknya record pada tabel Mahasiswa berdasarkan seluruh field, field nama, dan field nilai

```
MariaDB [test]> select count(*), count(nama), count(nilai)
    -> from mahasiswa;
+----------+-------------+--------------+
| count(*) | count(nama) | count(nilai) |
+----------+-------------+--------------+
|        7 |           7 |            6 |
+----------+-------------+--------------+
```

Hasil dari fungsi COUNT(nilai) = 6, karena ada sebuah data pada field nilai yang bernilai NULL.

12.2 Fungsi *String*/Teks

Fungsi string/teks dapat dioperasikan pada data-data yang bertipe string/teks. Fungsi ini menggunakan argumen berupa data string/teks, dan akan mengembalikan nilai yang berupa data string/teks. Fungsi-fungsi yang termasuk dalam fungsi string/teks, antara lain :

- **CONCAT (str1, str2, ...)**

  Fungsi ini menghasilkan string yang merupakan penggabungan dari argumen parameter str1, str2, ...

  Contoh :

  Menggabungkan kata 'SAYA' , 'BELAJAR', dan 'MySQL'

```
MariaDB [test]> select concat('SAYA' , 'BELAJAR', 'MySQL');
+-----------------------------------+
| concat('SAYA' , 'BELAJAR', 'MySQL') |
+-----------------------------------+
| SAYABELAJARMySQL                  |
+-----------------------------------+
```

```
MariaDB [test]> select concat('SAYA' , ' BELAJAR ', 'MySQL ');
+-------------------------------------+
| concat('SAYA' , ' BELAJAR ', 'MySQL ') |
+-------------------------------------+
| SAYA BELAJAR MySQL                  |
+-------------------------------------+
```

Jika salah satu string argumen pada parameternya bernilai NULL, maka hasil fungsi adalah NULL.

```
MariaDB [(none)]> select lcase ('Saya Belajar MySQL');
+------------------------------+
| lcase ('Saya Belajar MySQL') |
+------------------------------+
| saya belajar mysql           |
+------------------------------+
```

Jika argumen pada parameternya bertipe numerik, maka argumen tersebut akan diubah menjadi string dengan nilai yang sama.

```
MariaDB [test]> select concat('SAYA' , ' BELAJAR ', 123.45);
+-----------------------------------+
| concat('SAYA' , ' BELAJAR ', 123.45) |
+-----------------------------------+
| SAYA BELAJAR 123.45               |
+-----------------------------------+
```

- **CONCAT_WS (separator, str1, str2, ….)**

Fungsi ini akan menghasilkan string yang merupakan penggabungan argumen str1, str2, …. dengan argumen separator sebagai pemisah diantara str1, str2, …..

Contoh :

Menggabungkan kata 'SAYA', 'BELAJAR', dan 'MySQL' dengan menggunakan pemisah '*'

```
MariaDB [test]> select concat_ws('*' , 'SAYA' , 'BELAJAR' , 'MySQL');
+-----------------------------------------+
| concat_ws('*' , 'SAYA' , 'BELAJAR' , 'MySQL') |
+-----------------------------------------+
| SAYA*BELAJAR*MySQL                      |
+-----------------------------------------+
```

```
MariaDB [test]> select concat_ws('*' , 'SAYA' , NULL , 'MySQL');
+-------------------------------------+
| concat_ws('*' , 'SAYA' , NULL , 'MySQL') |
+-------------------------------------+
| SAYA*MySQL                          |
+-------------------------------------+
```

- **LCASE (str)** dan **LOWER (str)**

  Fungsi ini akan mengubah seluruh karakter argumen str, menjadi huruf kecil.

  Contoh :

  Mengubah kata 'Saya Belajar MySQL'

```
MariaDB [(none)]> select lcase ('Saya Belajar MySQL');
+------------------------------+
| lcase ('Saya Belajar MySQL') |
+------------------------------+
| saya belajar mysql           |
+------------------------------+
```

- **UCASE (str)** dan **UPPER (str)**

  Fungsi ini akan mengubah seluruh karakter argumen str, menjadi huruf besar.

  Contoh :

Mengubah kata 'Saya Belajar MySQL'

```
MariaDB [(none)]> select ucase ('Saya Belajar MySQL');
+-----------------------------+
| ucase ('Saya Belajar MySQL') |
+-----------------------------+
| SAYA BELAJAR MYSQL          |
+-----------------------------+
```

- **LEFT (str , len)** dan **RIGHT (str , len)**

  Fungsi ini memiliki 2 parameter, yaitu str bertipe string dan len bertipe bilangan bulat. Fungsi LEFT (str,len) ini akan menghasilkan string sebanyak len karakter yang diambil mulai dari karakter paling kiri string str. Fungsi RIGHT (str,len) ini akan menghasilkan string sebanyak len karakter yang diambil sampai karakter paling kanan string str.

  Contoh :

  Mengambil 4 karakter dari kiri pada kata 'JAKARTA'

  Mengambil 4 karakter dari kanan pada kata 'JAKARTA'

```
MariaDB [(none)]> select left('JAKARTA',4) , right('JAKARTA',4);
+-------------------+--------------------+
| left('JAKARTA',4) | right('JAKARTA',4) |
+-------------------+--------------------+
| JAKA              | ARTA               |
+-------------------+--------------------+
```

- **MID (str , pos , len)**

  Fungsi ini memiliki 3 parameter, str bertipe string, pos dan len, bertipe bilangan bulat. Fungsi ini menghasilkan string sebanyak len karakter yang dimulai dari karakter pada posisi ke pos dari string str. Posisi ke-1 dimulai dari posisi karakter paling kiri dari string str.

Contoh :

Mengambil 4 karakter mulai dari karakter ke 2 dari kata 'JAKARTA'

Mengambil 2 karakter mulai dari karakter ke 4 dari kata 'JAKARTA'

```
MariaDB [(none)]> select mid('JAKARTA',2,4) , mid('JAKARTA',4,2);
+-------------------+-------------------+
| mid('JAKARTA',2,4) | mid('JAKARTA',4,2) |
+-------------------+-------------------+
| AKAR              | AR                |
+-------------------+-------------------+
```

- **SUBSTRING (str , pos , len)** dan **SUBSTR (str , pos , len)**

Fungsi ini memiliki kegunaan yang sama dengan fungsi MID.

Contoh :

Mengambil 4 karakter mulai dari karakter ke 2 dari kata 'JAKARTA'

Mengambil 2 karakter mulai dari karakter ke 4 dari kata 'JAKARTA'

```
+-------------------------+-----------------------+
| substring('JAKARTA',2,4) | substr('JAKARTA',4,2) |
+-------------------------+-----------------------+
| AKAR                    | AR                    |
+-------------------------+-----------------------+
```

- **TRIM (str)** dan **LTRIM (str)** dan **RTRIM (str)**

  Fungsi ini akan menghasilkan string dengan menghilangkan karakter spasi yang terdapat pada argumen parameter str. LTRIM (str) akan menghilangkan karakter spasi yang berada di sebelah kiri string str. RTRIM (str) akan menghilangkan karakter spasi yang berada di sebelah kanan string str. TRIM (str) akan menghilangkan karakter spasi yang berada di sebelah kiri dan kanan string str.

  Contoh :

```
MariaDB [(none)]> select ltrim ('   SAYA BELAJAR   '),
    -> rtrim ('   SAYA BELAJAR   ');
+-----------------------------+-----------------------------+
| ltrim ('   SAYA BELAJAR   ') | rtrim ('   SAYA BELAJAR   ') |
+-----------------------------+-----------------------------+
| SAYA BELAJAR                |    SAYA BELAJAR             |
+-----------------------------+-----------------------------+
```

  LTRIM akan menghilangkan spasi yang ada di kiri string

  ‘   SAYA BELAJAR   ‘ sehingga hasilnya adalah ‘SAYA BELAJAR   ‘;

  RTRIM akan menghilangkan spasi yang ada di kanan string ‘   SAYA BELAJAR ‘ sehingga hasilnya adalah

  ‘   SAYA BELAJAR‘;

- **LENGTH (str)**

  Fungsi ini akan menghasilkan nilai panjang string str.

Contoh :

Menghitung panjang kata ‘JAKARTA’ dan ‘KOTA TUA’.

```
MariaDB [(none)]> select length ('JAKARTA'), length ('KOTA TUA');
+-------------------+--------------------+
| length ('JAKARTA') | length ('KOTA TUA') |
+-------------------+--------------------+
|                 7 |                  8 |
+-------------------+--------------------+
```

Panjang kata ‘JAKARTA’ adalah 7, sedangkan kata ‘KOTA TUA’ adalah 8, karena spasi antara karakter ‘A’ pada kata KOTA’ dan karakter ‘T’ pada kata ‘TUA’ terdapat spasi, yang juga diperhitungkan sebagai sebuah karakter.

12.3 Fungsi Tanggal dan Waktu

Fungsi tanggal dan waktu dapat dioperasikan pada data-data yang bertipe date atau time atau datetime, atau mengubah data bertipe string menjadi data bertipe tanggal dan atau waktu. Fungsi-fungsi yang termasuk dalam fungsi tanggal dan waktu, antara lain :

- **NOW ()** dan **SYSDATE()**

  Fungsi ini akan menghasilkan tanggal dan jam dari sistem ketika fungsi tersebut dipanggil.

  Contoh :

  Menampilkan tanggal dan waktu untuk saat ini.

```
MariaDB [(none)]> select now();
+---------------------+
| now()               |
+---------------------+
| 2023-02-04 21:48:58 |
+---------------------+
```

- **YEAR (date) , MONTH (date) , WEEK (date)**

  Fungsi ini memiliki satu parameter dengan argumen bertipe date (tanggal). YEAR (date) akan menghasilkan tahun dari argumen date. MONTH (date) akan menghasilkan nilai urutan bulan dalam setahun dari argumen date. WEEK (date) akan menghasilkan urutan minggu dalam setahun dari argumen date.

  Contoh :

  Menampilkan tahun, bulan, dan minggu untuk tanggal 23 Januari 2023

```
MariaDB [(none)]> select year('2023-01-23'),
    -> month('2023-01-23'), week('2023-01-23');
+--------------------+---------------------+--------------------+
| year('2023-01-23') | month('2023-01-23') | week('2023-01-23') |
+--------------------+---------------------+--------------------+
|               2023 |                   1 |                  4 |
+--------------------+---------------------+--------------------+
```

- **HOUR (time) , MINUTE (time) , SECOND (time)**

  Fungsi ini memiliki satu parameter dengan argumen yang bertipe time (jam). HOUR (time) akan menghasilkan nilai jam dari argumen yang diberikan. MINUTE (time) akan menghasilkan nilai menit dari argumen yang diberikan. SECOND (time) akan menghasilkan nilai detik dari argumen yang diberikan.

Contoh :

Menampilkan jam, menit, dan detik untuk waktu 21:48:59

```
MariaDB [(none)]> select hour('21:48:59'),
    -> minute('21:48:59'), second('21:48:59');
+------------------+--------------------+--------------------+
| hour('21:48:59') | minute('21:48:59') | second('21:48:59') |
+------------------+--------------------+--------------------+
|               21 |                 48 |                 59 |
+------------------+--------------------+--------------------+
```

- **ADDDATE (date , days)**
  SUBDATE (date , days)

Fungsi ini memiliki dua parameter, yaitu date dan days. Date memiliki argumen bertipe date (tanggal) yang menentukan tanggal dan days memiliki argumen bertipe bilangan bulat yang menentukan banyaknya hari. ADDDATE (date , days) akan menghasilkan tanggal setelah argumen tanggal di parameter date ditambahkan dengan argumen jumlah hari pada parameter days. SUBDATE (date , days) akan menghasilkan tanggal setelah argumen tanggal di parameter date dikurangkan dengan argumen jumlah hari pada parameter days.

Contoh :

Menampilkan tanggal 14 hari setelah tanggal 23 Januari 2023, dan menampilkan tanggal 14 hari sebelum tanggal 23 Januari 2023.

```
MariaDB [(none)]> select adddate('2023-01-23',14) ,
    -> subdate('2023-01-23',14);
+--------------------------+--------------------------+
| adddate('2023-01-23',14) | subdate('2023-01-23',14) |
+--------------------------+--------------------------+
| 2023-02-06               | 2023-01-09               |
+--------------------------+--------------------------+
```

- **ADDTIME (time1 , time2)**

  SUBTIME (time1 , time2)

  Fungsi ini memiliki dua parameter dengan argumen yang bertipe waktu (time) dengan format jam:menit:detik, contoh 3:10:40 yang berarti 3 jam 10 menit dan 40 detik. Parameter time1 dapat juga berisi argumen dengan tipe tanggal dan waktu (datetime) dengan format (tahun-bulan-tanggal jam:menit:detik), contoh '2023-01-23 22:50:10', yang berarti tanggal 23 Januari 2023 jam 22:50:10.

  ADDTIME (time1 , time2) akan menghasilkan lamanya waktu yang merupakan penjumlahan dari argumen time1 dan time2. SUBTIME (time1 , time2) akan menghasilkan lamanya waktu yang merupakan selisih dari argumen time1 dan time2.

  Jika argumen time1 bertipe datetime, maka hasil dari fungsi juga akan bertipe datetime. Jika kedua argumen (time1 dan time2) bertipe datetime maka fungsi akan menghasilkan nilai NULL.

Contoh :

```
MariaDB [(none)]> select addtime('22:50:10','3:20:30') ,
    -> subtime('22:50:10','3:20:30');
+-------------------------------+-------------------------------+
| addtime('22:50:10','3:20:30') | subtime('22:50:10','3:20:30') |
+-------------------------------+-------------------------------+
| 26:10:40                      | 19:29:40                      |
+-------------------------------+-------------------------------+
```

Menampilkan tanggal dan waktu setelah tanggal 23 Januari 2023 jam 22:50:10 ditambahkan dengan 3 hari 3 jam 20 menit 30 detik

```
MariaDB [(none)]> select addtime('2023-01-23 22:50:10','3 3:20:30');
+------------------------------------------+
| addtime('2023-01-23 22:50:10','3 3:20:30') |
+------------------------------------------+
| 2023-01-27 02:10:40                      |
+------------------------------------------+
```

Hasil jika kedua argumen bertipe datetime

```
MariaDB [(none)]> select addtime('2023-01-23 22:50:10','2023-01-23 3:20:30');
+---------------------------------------------------+
| addtime('2023-01-23 22:50:10','2023-01-23 3:20:30') |
+---------------------------------------------------+
| NULL                                              |
+---------------------------------------------------+
```

- **DATE_FORMAT (date , format) TIME_FORMAT (time , format)**

Fungsi ini menghasilkan format string dari argumen parameter date atau time, sesuai argumen parameter format yang ditentukan. Argumen format dapat dilihat pada tabel berikut :

Tabel 12.1 Format Date / Time

Bentuk	Keterangan
%a	Singkatan nama hari (Sun, Mon, …)
%b	Singkatan nama bulan (Jan, Feb, …)
%c	Nilai bulan dalam setahun (0...12)
%D	Hari dalam sebulan dengan akhiran dalam Bahasa Inggris (1^{st}, 2^{nd},3^{rd},…)
%d	Nilai hari dalam sebulan (00…31)
%e	Nilai hari dalam sebulan (0…31)
%f	Nilai mikrodetik (000000…999999)
%H	Nilai jam (00…23)
%h	Nilai jam (01…12)
%I	Nilai jam (01…12)
%i	Nilai menit (00…59)
%j	Nilai hari dalam setahun (001…366)
%k	Nilai jam (0…23)
%l	Nilai jam (1…12)
%M	Nama bulan (January…December)
%m	Nilai bulan dalam setahun (00…12)
%p	AM atau PM
%r	Waktu, 12 jam (hh:mm:ss diikuti AM atau PM)
%S	Nilai detik (00…59)
%s	Nilai detik (00…59)
%T	Waktu, 12 jam (hh:mm:ss)
%U	Nilai mingguan dalam setahun (00…53), dimana hari Minggu merupakan hari pertama setiap minggunya. Fungsi WEEK() mode 0
%u	Nilai mingguan dalam setahun (00…53), dimana hari Senin merupakan hari pertama setiap minggunya. Fungsi WEEK() mode 1
%W	Nama hari dalam seminggu (Sunday…Saturday)
%w	Nilai hari dalam seminggu (0=Sunday… 6=Saturday)
%Y	Nilai tahun dalam 4 digit

Bentuk	Keterangan
%y	Nilai tahun dalam 2 digit

Contoh :

Mengubah format data tanggal menjadi singkatan nama hari diikuti 2 digit tanggal - singkatan nama bulan – 4 digit tahun

```
MariaDB [(none)]> select date_format(now(),'%a %d-%b-%Y');
+----------------------------------+
| date_format(now(),'%a %d-%b-%Y') |
+----------------------------------+
| Sun 05-Feb-2023                  |
+----------------------------------+
```

Mengubah format data tanggal dan waktu menjadi nama bulan hari dengan akhiran Bahasa Inggris 4 digit tahun dan waktu dalam 12 jam diikuti AM/PM

```
MariaDB [(none)]> select date_format(now(),'%M %D %Y %r');
+----------------------------------+
| date_format(now(),'%M %D %Y %r') |
+----------------------------------+
| February 5th 2023 01:59:16 AM    |
+----------------------------------+
```

- **STR_TO_DATE (str , format)**

Fungsi ini akan mengubah argumen string pada parameter str menjadi nilai bertipe datetime, jika parameter str mengandung bagian date dan time. Fungsi ini akan mengubah argumen string pada parameter str menjadi nilai bertipe date atau time, jika parameter str hanya mengandung bagian date atau time saja. Bentuk format dapat dilihat pada tabel 12.1. Literal pada argumen di parameter format harus bersesuaian dengan bagian date atau time atau datetime pada argumen di parameter str. Jika literal antara str dan format tidak sesuai, maka fungsi akan menghasilkan nilai NULL.

Contoh :

String 01-05-20 akan diubah menjadi tipe date. 01 menjadi tanggal (format yang sesuai %d), 05 menjadi bulan (format yang sesuai %m), dan 20 menjadi tahun (format yang sesuai %Y).

```
iaDB [(none)]> select str_to_date ('01-05-20','%d-%m-%Y
-----------------------------------+
tr_to_date ('01-05-20','%d-%m-%Y') |
-----------------------------------+
020-05-01                          |
-----------------------------------+
```

Apabila literal argumen str tidak sesuai dengan literal argumen format maka fungsi akan menghasilkan nilai NULL.

```
aDB [(none)]> select str_to_date ('01-05-19','%d-%M-%Y
----------------------------------+
r_to_date ('01-05-19','%d-%M-%Y') |
----------------------------------+
LL                                |
```

Pada contoh di atas, literal 05 tidak sesuai dengan literal format yang menggunakan %M, karena format ini digunakan untuk nama bulan January…December.

12.4 Fungsi Tambahan Lainnya

Fungsi-fungsi yang termasuk dalam fungsi tambahan yang disediakan oleh MySQL, diantaranya adalah :

- **DATABASE ()**

Fungsi ini tidak memiliki parameter. Fungsi ini akan menghasilkan nama database yang sedang aktif atau digunakan.

Contoh :

```
MariaDB [test]> select database();
+------------+
| database() |
+------------+
| test       |
+------------+
```

- **USER ()**

Fungsi ini tidak memiliki parameter. Fungsi ini akan menghasilkan nama user atau nama host yang saat ini menggunakan MySQL.

Contoh :

```
MariaDB [test]> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
```

- **VERSION ()**

Fungsi ini tidak memiliki parameter. Fungsi ini akan menghasilkan versi dari MySQL yang digunakan.

Contoh :

```
MariaDB [test]> select version();
+-----------------+
| version()       |
+-----------------+
| 10.4.11-MariaDB |
+-----------------+
```

- **IF (kondisi , nilai_if_true , nilai_if_false)**

  Fungsi ini memiliki 3 parameter, yaitu kondisi yang akan dilakukan pengujian, nilai jika kondisi terpenuhi (true), dan nilai jika kondisi tidak terpenuhi (false).

  Contoh :

  Memberikan keterangan untuk nilai yang diperoleh mahasiswa pada tabel mahasiswa, dengan ketentuan jika nilai lebih dari 75 akan dinyatakan lulus, jika tidak maka dinyatakan gagal.

```
ariaDB [test]> select nim, nama, nilai ,
    -> if (nilai > 75,'LULUS','GAGAL') as keterangan
    -> from mahasiswa;
-----+-------------------+-------+------------+
nim  | nama              | nilai | keterangan |
-----+-------------------+-------+------------+
1234 | Alif Perdana      |    95 | LULUS      |
1235 | Budi Pratama      |    65 | GAGAL      |
1236 | Citra Utami       |    85 | LULUS      |
1237 | Desy Fridasari    |    70 | GAGAL      |
1238 | Eka Suryana       |    75 | GAGAL      |
1239 | Fitria Harumsari  |    90 | LULUS      |
1240 | Gita Swara        |  NULL | GAGAL      |
-----+-------------------+-------+------------+
```

BAB
XIII

VQuery dan View

Pendahuluan

Dalam setiap Sistem Manajemen Basis Data dikenal istilah Query dan View. Dalam Bab ini akan dibahas Query dan View pada Sistem Manajemen Basis Data dengan menggunakan MySQL. Query adalah setiap perintah yang digunakan untuk mengambil data dari sebuah tabel. MySQL dapat digunakan untuk query data, memfilter data, menyortir data, menggabungkan tabel, mengelompokkan data, memodifikasi data. Tampilan adalah kueri tersimpan yang dijalankan MySQL saat tampilan dipanggil. Query biasanya berupa pernyataan SELECT yang mengambil data dari satu atau beberapa tabel. SELECT sebagai DML telah dibahas pada bab sebelumnya, oleh karena itu dalam bab ini akan difokuskan pada query yang melibatkan paling tidak dua tabel.

13.1 Query

Pembahasan mengenai query dimulai dari yang paling sederhana yaitu Simple Join dan menggunakan table-table dari bukunya Thomas Connolly (2014) yang telah diolah dengan menggunakan DBMS MySQL. Berikut adalah table-table yang digunakan.

```
SELECT * FROM `branch`
```

branchNo	street	city	postcode
B002	56 Clover Dr	London	NW10 6EU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU

```
SELECT * FROM `staff`
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
SL21	John	White	Manager	M	1965-10-01	30000	B005
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

```
SELECT * FROM `propertyforrent`
```

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	*NULL*	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

13.2 Simple Join

Simple join sering disebut juga INNER JOIN digunakan untuk menampilkan record yang cocok dari kedua tabel. Ini adalah tipe gabungan yang akan dapatkan secara default jika kata kunci INNER (atau kata kunci lainnya, seperti LEFT , RIGHT , atau FULL) dihilangkan dan cukup gunakan JOIN. Biasanya ada dua (atau lebih) tabel dalam pernyataan gabungan.

Simple join menghasilkan query hanya tuple-tuple dari kedua tabel yang memiliki nilai key yang sama (contoh : c.clientNo = v.clientNo). Simple join ini sama dengan equi-join pada aljabar relasional.

Berikut ini contoh kasus query simple join dengan melibatkan dua table dan mengurutkan hasilnya.

- *Untuk setiap kantor cabang, tampilkan nomor staf dan nama staf yang mengelola properti beserta nomor propertinya.*

```
SELECTst.branchNo, st.
staffNo, fName, lName,propertyNo
FROM Staff st, PropertyForRent pro
WHERE st.staffNo = pro.staffNo
ORDER BY st.branchNo, st.staffNo, propertyNo
```

Hasilnya :

branchNo ▲ 1	staffNo	fName	lName	propertyNo ▲ 3
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

Berikut ini contoh kasus simple join dengan melibatkan tiga table dengan mengurutkan hasilnya.

- *Untuk setiap kantor cabang, tampilkan staff yang mengelola properti termasuk kota dimana kantor cabang tersebut berada beserta nomor properti yang dikelola mereka.*

SELECT br.branchNo, br.city,st.staffNo,fName, lName, propertyNo

, Staff st, PropertyForRent proWHERE br. branchNo =st.branchNo

AND st.staffNo = pro.staffNo

ORDERBYbr.branchNo,st.staffNo,propertyNo

Hasilnya :

branchNo ▲ 1	city	staffNo	fName	lName	propertyNo ▲ 3
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

Berikut ini contoh join dengan Multiple Grouping Columns.

- *Temukan jumlah properti yang dikelola oleh setiap staff.*

```
SELECT st.branchNo, st.staffNo,
 COUNT(*) AS myCount
FROM Staff st, PropertyForRent pro
WHERE st.staffNo = pro.staffNo
GROUP BY st.branchNo, st.staffNo
ORDER BY st.branchNo, st.staffNo
```

Hasilnya :

branchNo ▲ 1	staffNo	myCount
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

13.3 Union

Union adalah operator untuk menggabungkan hasil dari dua Query atau lebih ke dalam kumpulan hasil tunggal berbeda yang mencakup semua baris milik semua Query di Union. Dengan kata lain, UNION digunakan untuk menggabungkan hasil dari dua atau lebih pernyataan SELECT, dengan ketentuan :

- Pernyataan SELECT dalam UNION memiliki jumlah kolom yang sama,
- Setiap kolom memiliki tipe data yang sejenis,
- Setiap kolom dalam pernyataan SELECT memiliki urutan yang sama.

Berikut ini contoh kasus penggunaan UNION.

- *Tampilkan semua kota dimana kantor cabang atau properti berada.*

```
(SELECT city, branchNo
FROM Branch
WHERE city IS NOT NULL)
UNION
(SELECT city, branchNo
FROM PropertyForRent
WHERE city IS NOT NULL)
```

Hasilnya :

city	branchno
London	B002
Glasgow	B003
Bristol	B004
London	B005
Aberdeen	B007

Intersect

Operator INTERSECT menggabungkan dua pernyataan SELECT dan hanya mengembalikan kumpulan data yang ada di kedua pernyataan tersebut. Sederhananya, ini seperti irisan pada dua himpunan dalam matematika.

INTERSECT berbeda dengan INNER JOIN, INNER JOIN adalah operator yang umumnya cocok dengan sekumpulan kolom terbatas dan dapat mengembalikan nol baris atau lebih banyak baris dari salah satu tabel. INTERSECT adalah operator berbasis set yang membandingkan baris lengkap antara dua set dan tidak pernah dapat mengembalikan lebih banyak baris daripada di tabel yang lebih kecil.

Berikut ini adalah contoh kasus dari query yang menggunakan INTERSECT.

- *Tampilkan semua kota dan nomor kantor cabang dimana kantor cabang dan properti berada.*

(SELECT city, BranchNo FROM Branch)

13.4 INTERSECT

(SELECT city, BranchNo FROM PropertyForRent);

Karena dalam MySQL tidak ada operator INTERSECT, maka penulisannya dapat digantikan dengan operator IN untuk mensimulasikan kasus Intersect ini.

```
SELECT city FROM Branch
WHERE city
IN (SELECT city FROM PropertyForRent)
```

Hasilnya :

city
London
Glasgow
London
Aberdeen

13.5 Difference (Except)

Pernyataan SQL EXCEPT digunakan untuk memfilter record berdasarkan INTERSECTION record yang dihasilkan melalui dua pernyataan SELECT record yang ada di antara dua tabel difilter dari tabel di sisi kiri pernyataan EXCEPT dan record yang tersisa dikembalikan.

Pernyataan EXCEPT tidak berbeda dengan

pernyataan MINUS. Keduanya memiliki tujuan yang sama dan keduanya hanyalah dua cara berbeda untuk mencapai fungsi yang sama. Perbedaannya adalah EXCEPT tersedia di database PostgreSQL sedangkan MINUS tersedia di MySQL dan Oracle.

Berikut ini adalah contoh kasus bagaimana menggunakan EXCEPT dalam sebuah query.

- *Tampilkan semua kota dimana ada kantor cabang tanpa ada propertinya.*

(SELECT city FROM Branch)

EXCEPT

(SELECT city FROM PropertyForRent);

Dalam MySQL operator EXCEPT dapat ditulis seperti dibawah ini:

```
SELECT DISTINCT city FROM Branch WHERE city
NOT IN
(SELECT city FROM PropertyForRent)
```

Hasilnya :

city
Bristol

13.6 VIEW

Dalam bahasa SQL, sebuah View adalah tabel virtual yang dihasilkan dari pernyataan SQL. VIEW seperti halnya sebuah tabel berisi baris dan kolom. Kolom atau Field pada VIEW adalah kolom atau field dari satu atau lebih tabel nyata dalam database.

Pada View dapat ditambahkan pernyataan dan fungsi SQL dan menyajikan data seolah-olah data berasal dari sebuah tabel.

13.7 Membuat View

Sebuah View dapat dibuat dengan pernyataan CREATE VIEW, berikut ini adalah sintaksnya.

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

View selalu menampilkan data terbaru. Mesin basis data membuat ulang view, setiap kali pengguna menjalankan query. Berikut ini adalah contoh pernyataan SQL untuk membuat sebuah view yang berisi staff yang berasal dari cabang dan kota, kemudian menampilkan staff yang berasal dari kota 'Glasgow'.

```
CREATE VIEW Staff_City AS
SELECT br.branchNo, br.city, st.staffNo,
    fName, lName, propertyNo
FROM Branch br,Staff st,PropertyForRent pro
WHERE br.branchNo = st.branchNo
    AND st.staffNo = pro.staffNo;
```

Dari View di atas dapat dibuat sebuah seperti di bawah ini:

SELECT * FROM Staff_City Where City = 'Glasgow';

Hasilnya :

```
SELECT * FROM `Staff_City` WHERE City = 'Glasgow'
```

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36

Pernyataan SQL berikut adalah contoh membuat View yang memilih setiap salary dalam tabel "Staff" dengan salary-nya lebih tinggi dari salary rata-rata:

```
CREATE VIEW Salary_Staff_diatas_Rata2 AS
SELECT staffNo, fName, lName, Salary
```

```
FROM Staff
WHERE Salary > (SELECT AVG(Salary)
    FROM Staff);
```

Sebuah query dapat dibuat dari view di atas seperti di bawah ini:

SELECT * FROM Salary_Staff_diatas_Rata2;

Hasilnya :

staffNo	fName	lName	Salary
SG14	David	Ford	18000
SG5	Susan	Brand	24000
SL21	John	White	30000

13.8 Mengubah View

Sebuah VIEW dapat diubah dengan pernyataan CREATE OR REPLACE VIEW, berikut ini adalah sintaksnya.

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Pernyataan SQL berikut menambahkan kolom "Position" pada View "staff city" :

CREATE VIEW Staff_City AS

SELECT br.branchNo, br.city, st.staffNo,

fName,lName, propertyNo, Position br,Staff st,PropertyForRent proWHERE br. branchNo = st.branchNo

AND st.staffNo = pro.staffNo

Kemudian dapat dilakukan query terhadap view yang sudah diubah, maka hasilnya :

```
SELECT * FROM staff_city
```

branchNo	city	staffNo	fName	lName	propertyNo	Position
B003	Glasgow	SG14	David	Ford	PG16	Supervisor
B003	Glasgow	SG37	Ann	Beech	PG21	Assistant
B003	Glasgow	SG37	Ann	Beech	PG36	Assistant
B005	London	SL41	Julie	Lee	PL94	Assistant
B007	Aberdeen	SA9	Mary	Howe	PA14	Assistant

13.9 Menghapus View

Sebuah VIEW dapat dihapus dengan pernyataan DROP VIEW, berikut ini adalah sintaksnya.

DROP VIEW view_name;

Pernyataan SQL berikut menghapus View "Staff_City" :

```
DROP VIEW Staff_City;
```

View sering digunakan untuk membantu programmer dalam pembuatan laporan pada pengembangan aplikasi. Dengan View juga akses terhadap database dapat dibatasi tidak langsung pada table utama.

BAB
XIV

Hak Akses User

Pendahuluan

Hak akses user merupakan salah satu faktor penting dalam basis data. Untuk menghindari hal-hal yang tidak diinginkan, user harus diberikan batasan-batasan akses yang berbeda. Hal ini diperlukan agar tidak semua user mendapatkan kontrol penuh terhadap seluruh basis data termasuk tabel user. Pada bab ini akan membahas tentang bagaimana membuat akun user baru dan memberikan hak akses yang sesuai kepada user. Selain itu bab ini juga akan membahas tentang tipe hak akses apa saja yang ada pada MySQL, bagaimana memberikan hak akses sesuai kebutuhan, serta bagaimana menghapus hak akses user, termasuk mengganti password dan menghapus user di MySQL.

14.1 User Default MySQL

Secara default, MySQL telah memberikan username dan password dengan level akses root yang memiliki kendali penuh terhadap seluruh database dan tabel. Disamping itu, user tersebut juga dapat digunakan untuk mengelola akun user.

Pada sistem operasi Microsoft Windows, username dan password default adalah root dengan password *null*. Sedangkan pada sistem operasi Linux, username dan password secara default adalah root dengan password yang sama dengan username, yaitu root.

14.2 Membuat Akun User Baru di MySQL

Terdapat 2 (dua) bagian pada sebuah akun user di MySQL, yaitu nama user dan nama host. Untuk membuat akun user pada MySQL dapat dilakukan dengan perintah berikut pada terminal MySQL:

CREATE USER ‘username_baru’@’localhost’ IDENTIFIED BY ‘password_baru’;

Pada perintah di atas, nama host adalah localhost, dimana user hanya dapat mengakses MySQL dari komputer yang terdapat MySQL di dalamnya.

Namun jika user ingin dapat mengakses dari komputer lain, maka akun user dapat dibuat dengan perintah berikut:

CREATE USER ‘username_baru’@’10.0.0.1’ IDENTIFIED BY ‘password_baru’;

Pada perintah di atas, nama host diganti dengan alamat IP 10.0.0.1, sehingga jika komputer lain di dalam jaringan ingin mengakses server MySQL cukup dengan menuliskan alamat host tersebut.

Sedangkan jika server MySQL ingin dapat diakses dari komputer manapun maka dapat menggunakan perintah berikut:

CREATE USER ‘username_baru’@’%’

IDENTIFIED BY ‘password_baru’;

Namun sebaiknya hal ini dihindari agar keamanan basis data dapat lebih terjaga.

Setelah menambahkan akun user baru, perlu untuk memberikan hak akses kepada user tersebut agar dapat mengakses database MySQL. Jika tidak, user tersebut tidak akan dapat digunakan untuk mengakses maupun memanipulasi database MySQL.

14.3 Memberikan Hak Akses User di MySQL

Pada MySQL terdapat beberapa hak akses yang dapat diberikan pada user, diantaranya adalah sebagai berikut:

a. ALL PRIVILEGES: Memberikan hak akses penuh kepada user.
b. CREATE: User diperbolehkan untuk membuat database dan tabel.
c. DROP: User diperbolehkan untuk menghapus database dan tabel.
d. SELECT: User diperbolehkan untuk membaca data pada tabel.
e. INSERT: User diperbolehkan untuk memasukkan data ke dalam tabel.

f. UPDATE : User diperbolehkan untuk mengubah data pada tabel.

g. DELETE: User diperbolehkan untuk menghapus data pada tabel.

Untuk memberikan hak akses secara spesifik terhadap user, maka dapat digunakan perintah sebagai berikut:

GRANT hak_akses1, hak_akses2 ON nama_ database.nama_tabel

TO ‘username’@’localhost’;

Berikut adalah contoh perintah untuk memberikan hak akses penuh kepada user pada database yang ditentukan:

GRANT ALL PRIVILEGES ON nama_database.*

TO ‘username’@’localhost’;

Berikut adalah contoh perintah untuk memberikan hak akses penuh kepada user pada seluruh database:

GRANT ALL PRIVILEGES ON *.*

TO ‘username’@’localhost’;

Pada perintah di atas, user akan diberikan hak akses yang sama dengan user root.

Berikut adalah contoh perintah untuk memberikan hak akses penuh kepada user pada tabel tertentu di dalam database yang ditentukan:

GRANT ALL PRIVILEGES ON nama_database. nama_tabel

TO ‘username’@’localhost’;

Berikut adalah contoh perintah untuk memberikan beberapa hak akses sekaligus kepada user pada database yang ditentukan:

GRANT SELECT, INSERT, UPDATE ON nama_ database.* TO username@’localhost’;

14.4 Menampilkan Hak Akses User di MySQL

Untuk menampilkan hak akses user dapat menggunakan perintah SHOW GRANTS.

SHOW GRANTS FOR ‘username’@’localhost’;

Pada perintah di atas akan menghasilkan tampilan seperti berikut:

```
+--------------------------------------------------------+
| Grants for username@localhost                          |
+--------------------------------------------------------+
|GRANT USAGE ON *.* TO 'username' @
  localhost'                                             |
| GRANT ALL PRIVILEGES ON `nama_database`.*
  TO 'username'@'localhost'                              |
+--------------------------------------------------------+
2 rows in set (0.00 sec)
```

14.5 Menghapus Hak Akses User di MySQL

Untuk menghapus hak akses user pada dasarnya hampir sama dengan perintah untuk memberikan hak akses pada user.

Untuk menghapus hak akses user pada database tertentu dapat menggunakan perintah sebagai berikut:

```
REVOKE ALL PRIVILEGES ON nama_database.*
FROM 'username'@'localhost';
```

Perintah di atas digunakan untuk menghapus seluruh hak akses user pada database yang ditentukan.

14.6 Mengganti Password User di MySQL

Untuk mengganti password user pada MySQL dapat menggunakan perintah berikut.

```
UPDATE user SET Password=PASSWORD("password_baru")
WHERE User='username';
```

14.7 Menghapus User di MySQL

Untuk menghapus user pada MySQL dapat menggunakan perintah DROP USER.

```
DROP USER 'username'@'localhost';
```

Perintah di atas digunakan untuk menghapus user dan seluruh hak akses user tersebut.

Sedangkan jika ingin menghapus beberapa user sekaligus dapat menggunakan perintah seperti berikut:

```
DROP USER
'username1'@'localhost',
'username2'@'localhost',
'username3'@'localhost';
```

Jika user yang ingin dihapus sedang aktif, maka harus di-non aktifkan terlebih dahulu dengan melihat nomor ID user yang akan dihapus. Untuk melihat nomor ID user yang sedang aktif dapat dilakukan dengan perintah berikut:

```
SHOW PROCESSLIST;
```

Setelah perintah tersebut dijalankan, akan terlihat daftar user yang sedang aktif beserta nomor ID pada user tersebut. Selanjutnya dapat menjalankan perintah berikut untuk menon-aktifkan user yang akan dihapus.

KILL Id_number;

Dimana Id_number adalah nomor ID user yang sedang aktif yang akan dihapus. Selanjutnya jalankan perintah untuk menghapus user seperti perintah sebelumnya.

DROP USER ‘username’@’localhost’; yang sedang aktif yang akan dihapus. Selanjutnya jalankan perintah untuk menghapus user seperti perintah sebelumnya.

DROP USER ‘username’@’localhost’;

Daftar Pustaka

Alvin Dwi Hardiansyah, & Dewi, C. N. P. (2020). Perancangan Basis Data Sistem Informasi Perwira Tugas Belajar (Sipatubel) Pada Kementerian Pertahanan. *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (SENAMIKA)*, *1*(2), 222–233.

Maulana, H. (2016). Analisis Dan Perancangan Sistem Replikasi Database Mysql Dengan Menggunakan Vmware Pada Sistem Operasi Open Source. *InfoTekJar (Jurnal Nasional Informatika Dan Teknologi Jaringan)*, *1*(1), 32–37. https://doi.org/10.30743/infotekjar.v1i1.37

Mulyana, E., & Wahana, A. (2017). Rancang Bangun Sistem Basis Data Penelitian Menggunakan Top Down Approach. *TELKA - Telekomunikasi, Elektronika, Komputasi Dan Kontrol*, *3*(2), 152–167. https://doi.org/10.15575/telka.v3n2.152-167

Munawaroh, S. (2005). Mengeksplorasi Database PostgreSQL dengan PgAdmin III. *Jurnal Teknologi Informasi DINAMIK*, *X*(2), 103–107.

Radiyyah, A. F., Lubis, M. P., Pasaribu, Y. O., & Nazara, B. S. (2022). Pengenalan Serta Penggunaan Microsoft Eccess pada Perusahaan. *Jurnal Ilmu Komputer, Ekonomi Dan Manajemen (JIKEM)*, *2*(2), 2763–2769.

Sudarso. (2021). Pemanfaatan Basis Data, Perangkat Lunak dan Mesin Industri Dalam Meningkatkan Produksi Perusahaan (Literature Review Executive Support Sistem (Ess) for Business). *Jurnal Manajemen Pendidikan Dan Ilmu Sosial*, *3*(1), 1–14. https://doi.org/10.38035/jmpis.v3i1

Dantes, Gede Rasben, dkk. 2019. "Pengantar Basis Data". Rajawali Pers. Depok. Edisi 1. ISBN 978-602-425-387-5. Hal. 14-15.

Suryanto. Y., 2015. "Lingkungan Basis Data", diakses tanggal 12 Januari 2023. https://surya-sisteminformasi.blogspot.com/2015/08/lingkungan-basis-data-1.html

ANSI/X3/SPARC Study Group on Data Base Management Systems, 1975, Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2, diakses 12 Januari 2023. https://dl.acm.org/action/showFmPdf?doi=10.1145%2F984332

Jardine, Donald A. 1977. The ANSI/SPARC DBMS Model. North-Holland Pub. Co. ISBN 0-7204-0719-2, diakses tanggal 12 Januari 2023. https://catalogue.nla.gov.au/Record/3562331

Jeffrey A. Hoffer, Mary B. Prescott, Fred R. McFadden. 2005. "The Database Environment, Modern Database Management, 7th Edition". Prentice Hall. Diakses tanggal 12 Januari 2023.

https://studylib.net/doc/10202680/here

Strickland, Myrtle. 2019. “Database Environment” tanggal 12 Januari 2023. https://slideplayer.com/slide/13084654/

Fikry, M., 2019. Basis Data. UNIMAL PRESS, 2019

Hoffer, B.L., 2002. Language Borrowing and Language Diffusion: An Overview. Intercultural Studies XI-4.

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

Turban, E., Kim, D., McKay, J. et al., 2015. Electronic Commerce: A Managerial Perspective (8th Edition). London: Springer

Considine, B., Parkes, A., Olese, K., Blount, Y., & Speer, D. (2012). *Accounting information systems: understanding business processes* (4th ed.). John Wiley & Sons.

Coronel, Carlos., Morris, S. (Steven A.), & Rob, Peter. (2011). *Database systems : design, implementation, and management*. Course Technology Cengage Learning.

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (M. Goldstein & K. Loanes, Ed.; Seventh). PEARSON.

Hoffer, J., Ramesh, V., & Topi, H. (2016). *Modern Database Management* (12th Edition). Pearson Education.

D. Soyusiawaty., S. Winiarti., M. Rosyda., J. Fahana. 2020. “Buku Ajar Mata Kuliah Basis Data”, Program Studi Teknik Informatika, Universitas Ahmad Dahlan. Yogyakarta : Indonesia.

M. Fikry., 2019. “Basis Data”, Unimal Press : Lhokseumawe. ISBN : 978-602-464-078-1.

P.S Chen., 1977. “The Entity-Relationship Model: Toward Unified View of Data”, Center for Information System Research, Massachusetts Institute of Technology : America.

R.A Putri,. 2022 “Buku Ajar : Basis Data”, Media Sains Indonesia, Bandung : Indonesia. ISBN : 978-623-362-547-0.

S.S Bagui, Richard W.E., 2022. “Database Design Using Entity-Relationship Diagrams”, CRC Press : Florida, America. DOI: 10.1201/9781003314455.

Date, C. J. (2004). *An Introduction to Database Systems 8e.* California: Pearson.

Harrington, J. L. (2016). *Relational Database Design and Implementation.* Cambrige: Morgan Kaufman.

Navathe, R. E. (2016). *FUNDAMENTALS OF Database Systems.* New Jersey: Pearson.

S. Sumathi, S. E. (2007). *Fundamentals of Relational Database Management Systems.* New York: Springer.

Date, C. (2001). The database relational model : a retrospective review and analysis : a historical account and assessment of E.F. Codd’s contribution to the field of database technology. Massachussets: Addison-Wesley Longman inc. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Edgar_F._Codd

Elmasri, R., & Navathe, S. (2016). Fundamental of Database System, 7th ed. New York: Pearson Education.

Rob, P., & Coronel, C. (2009). Database Systems : Design, Implementation and Managment, 8th ed. Boston: Thomson Course Technology.

Thakur, S. (2016, June 12). Database-normalization-explain-1nf-2nf-3nf-bcnf-with-examples. Retrieved from WhatIsDBMS.com: https://whatisdbms.com/database-normalization-explain-1nf-2nf-3nf-bcnf-with-examples/

Amornchewin, R. (2018). The Development of SQL Language Skills in Data Definition and Data Manipulation Languages Using Exercises with Quizizz for Students' Learning Engagement. *Indonesian Journal of Informatics Education*, 85-90.

Brooks, C. (2022, November 16). *Business News Daily*. Retrieved from What is SQL: https://www.businessnewsdaily.com/5804-what-is-sql.html

Hakim, R. F. (2019, July 16). *medium.com.* Retrieved from Tipe Data SQL: https://medium.com/@986110101/tipe-data-sql-1c47a91605c6

Java T Point. (2022, January 08). *Java T Point.* Retrieved from SQL Command: https://www.javatpoint.com/dbms-sql-command

Phillips, M. C. (1985). Early Meetings of the

Conference on Data Systems Languages. *Annals of the History of Computing 7*, 316-325.

Jubilee, Enterprise. 2017. Otodidak MySQL untuk Pemula.

Elex Media Komputindo: Jakarta.

Sianipar, R.H. 2017. Belajar Cepat Pemrograman Query

dengan MySQL. ANDI: Yogyakarta.

Solichin, Ahmad. 2010. MySQL 5 dari Pemula Hingga

Mahir. Universitas Budi Luhur: Jakarta.

MySQL Functions. (n.d.). Retrieved January 17, 2023, from https://www.w3schools.com/mysql/mysql_ref_functions.asp

MySQL: IF Function. (n.d.). Retrieved January 17, 2023, from https://www.techonthenet.com/mysql/functions/if.php

MySQL :: MySQL 8.0 Reference Manual: 12.1 Built-In Function and Operator Reference. (n.d.). Retrieved January 17, 2023, from https://dev.mysql.com/doc/refman/8.0/en/built-in-function-reference.html

Peterson, R. (2020, March 26). *MySQL Functions: String, Numeric, User-Defined, Stored*. https://www.guru99.com/functions.html

MySQL Tutorial. (2023, January 17). Retrieved from w3 schools: www.w3schools.com

Thomas Connolly, C. B. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management.* Pearson Education Limited.

Butcher, Tony, 2002. "Sams Teach Yourself MySQL in 21 Days, Second Edition", Sams Publishing.

Sverdlov, Etel, 2012. "How To Create a New User and Grant Permissions in MySQL", Digital Ocean.

Tentang Penulis

I Putu Dody Suarnatha, S.Kom., M.Kom, akrab dipanggil dody, Lahir di Mengwi, 6 Maret 1993. Penulis mengawali pendidikan pada SDN 01 Buduk tahun 1998. Merupakan alumni SMPN 1 Mengwi tahun 2004, dan SMKN 1 Denpasar pada tahun 2010. Setelah tamat SMK, penulis melanjutkan studi ke S1 mengambil jurusan Teknik Informatika di STMIK STIKOM INDONESIA,". Di tahun 2018 penulis melanjutkan studi pada jenjang Strata-2 dan mengambil jurusan Sistem Informasi di Universitas Pendidikan Ganesha (UNDIKSHA). Sejak kuliah penulis memiliki ketertarikan dengan bidang kajian yang berhubungan dengan dunia Pendidikan. Buku ini merupakan salah satu karya yang dihasilkan penulis.

Tentang Penulis

Michael Sitorus, S.Kom., M.Kom., C.HRA, buku ini adalah salah satu karya dan akan secara konsisten disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Saya bersuku Batak di Indonesia kelahiran kota Medan. Saya berlatarbelakang pendidikan Ilmu Komputer yakni Pendidikan S1 di STMIK Sisingamangaraja XII Medan dan S2 di Universitas Budi Luhur Jakarta. Saya punya pengalaman mengajar di *University of Indonesian*, Institut Teknologi dan Bisnis Bank Rakyat Indonesia/*Cyber University*, Politeknik Kesehatan 1 Jakarta, dan *University of Satya Negara Indonesia*. Saya berpengalaman di IT & Bisnis Digital aktif sebagai *IT Concultant* serta memegang *Best Moderator* KEMENKOPUKM RI tahun 2021 dan juga mendapatkan penghargaan Best Lecturer dengan Bahan Ajar Terbanyak dari BRInnovate Awards 2020. Saya juga pernah membawa mahasiswa menghasilkan Juara-Juara tingkat Internasional dan Nasional. Saya juga pernah membimbing mahasiswa mendapatkan HIBAH KEWIRAUSAHAAN (HOKI) yang diselenggarakan oleh PT. PENGADAIAN tahun 2022-2023. Saya juga mendirikan sebuah perusahaan Konsultan IT bernama PT. KRISYNDO, disana saya sebagai Founder dan sekaligus CEO nya. Selain itu saya aktif sebagai Master Trainer Digital Enterprneurship Academy (DEA) di KEMKOMINFO RI.

Penulis dapat dihubungi melalui kontak personal ke 085359463609 atau dapat kontak via email ke : michaelmangatursitorus@gmail.com

Tentang Penulis

Rifka Widyastuti, penulis kelahiran Baturaja, 19 September 1992. Penulis telah menyelesaikan pendidikan s-1 Teknik Informatika, Universitas Sriwijaya pada tahun 2014. Kemudian penulis melanjutkan pendidikan dan lulus tahun 2018 dari S-2 Magister Teknologi Informasi, Universitas Indonesia dan S-2 Departement Information Technology, National Taiwan University of Science and Technology (NTUST). Saat ini penulis menggeluti bidang computer science kurang lebih selama 8 tahun mulai tahun 2014 dan sangat tertarik pada bidang *Software Engineering*, *Data Science*, dan *Artificial Intelligence*. Buku ini adalah salah satu karya dan inshaa allah secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Penulis dapat dihubungi melalui kontak personal ke +62-851-3247-9030 atau via email ke: rifka.widyastuti@gmail.com

Tentang Penulis

Deddy Kurniawan, laki-laki kelahiran Samarinda, 08 Oktober 1996**.** Penulis menyelesaikan pendidikan S-1 di STMIK Widya Cipta Dharma Samarinda pada Program Studi Teknik Informatika pada tahun 2018, kemudian melanjutkan pendidikan S-2 di Bina Nusantara University Program Magister Jurusan Teknik Informatika dan berhasil menyelsaikan pendidikan pada tahun 2021. Pada tahun yang sama penulis mulai terjun menggeluti dunia pendidikan. Pada saat ini penulis mengajar aktif di salah satu universitas swasta di Kalimantan Timur Kota Samarinda. Beberapa artikel ilmiah telah dipublikasikan oleh penulis baik di Jurnal Nasional maupun Jurnal Internasional bereputasi. Penulis tertarik pada bidang riset khususnya Data Mining, Decision Support Model dan Fuzzy Logic. Buku ini adalah salah satu karya dan semoga kedepannya secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Penulis dapat dihubungi melalui kontak dan email: 085389375175 dan deddy.stmik14@gmail.com.

Tentang Penulis

Satya Arisena Hendrawan, pria kelahiran Jakarta, 24 Mei 1992. Penulis telah menyelesaikan pendidikan dan lulus tahun 2015 dari S-1 Sistem Komputer, Universitas Diponegoro. Kemudian, penulisan melanjutkan pendidikan dan lulus tahun 2018 dari S-2 Ilmu Komputer, Institut Pertanian Bogor. Penulis telah menggeluti bidang *software engineering* selama 3-5 tahun dan mulai tahun 2017 tertarik minat untuk belajar di bidang *data science*. Pada tahun 2019 hingga kini, penulis mulai aktif mengajar di salah satu perguran tingi swasta di Jakarta. Penulis sangat tertarik dan antusias pada bidang *Software Engineering*, *Data Science* dan *Artificial Intelligence*. Buku ini jadi salah satu karya dan semoga kedepannya secara konsisten untuk menulis pada buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Penulis dapat dihubungi melalui kontak personal ke +62-821-3843-2680 atau dapat kontak via email ke : arisenahendrawan@gmail.com

Tentang Penulis

Satya Arisena Hendrawan, pria kelahiran Jakarta, 24 Mei 1992. Penulis telah menyelesaikan pendidikan dan lulus tahun 2015 dari S-1 Sistem Komputer, Universitas Diponegoro. Kemudian, penulisan melanjutkan pendidikan dan lulus tahun 2018 dari S-2 Ilmu Komputer, Institut Pertanian Bogor. Penulis telah menggeluti bidang *software engineering* selama 3-5 tahun dan mulai tahun 2017 tertarik minat untuk belajar di bidang *data science*. Pada tahun 2019 hingga kini, penulis mulai aktif mengajar di salah satu perguran tingi swasta di Jakarta. Penulis sangat tertarik dan antusias pada bidang *Software Engineering*, *Data Science* dan *Artificial Intelligence*. Buku ini jadi salah satu karya dan semoga kedepannya secara konsisten untuk menulis pada buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Penulis dapat dihubungi melalui kontak personal ke +62-821-3843-2680 atau dapat kontak via email ke : arisenahendrawan@gmail.com

Tentang Penulis

Dwipo Setyantoro, adalah seorang ayah dari 3 anak yang dilahirkan di Jakarta pada pertengahan Mei 1969. Penulis menyelesaikan studi S-1 di Ilmu Komputer Universitas Indonesia pada tahun 1995 dan melanjutkan studi S-2 di program studi Magister Manajemen Sistem Informasi di Universitas Gunadarma. Penulis mengawali karirnya sebagai dosen pada tahun 2008 dan mengajar di beberapa kampus ternama di Jakarta. Mata kuliah yang kerap diampu adalah Algoritma dan Pemrograman, Basis Data, serta Analisa & Perancangan Sistem Informasi. Buku ini adalah salah satu karya penulis dan semoga secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Untuk kepentingan korespondensi, penulis dapat dihubungi di kontak e-mail dwipo.setyantoro@gmail.com

Tentang Penulis

Akhmad Pandhu Wijaya, S.Kom., M.Kom, penulis berkelahiran Semarang, 18 Juni 1993. Saat ini penulis merupakan dosen tetap pada Universitas Wahid Hasyim Semarang pada Program Studi Teknik Informatika. Penulis telah menyelesaikan Pendidikan S-1 Teknik Informatika Universitas Dian Nuswantoro Semarang pada tahun 2015, dan melanjutkan pendidikan S-2 Magister Teknik Informatika di Universitas Dian Nuswantoro dan tahun lulus 2018. Penulis memiliki minat pada bidang *data science* dengan mayoritas riset pada bidang *text mining*, *data mining*. Buku ini merupakan salah satu karya dan Inshaa Allah secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan.

Penulis dapat dihubungi melalui kontak email ke :

Email aktif : pandhudsn@unwahas.ac.id

Tentang Penulis

Immanuela Puspasari Saputro, perempuan kelahiran Kudus, 25 Mei 1978**.** Penulis menyelesaikan pendidikan S-1 di Universitas Sanata Dharma Yogyakarta Program Studi Ilmu Komputer kemudian melanjutkan pendidikan S-2 di Universitas Atma Jaya Yogyakarta Program Magister Jurusan Teknik Informatika. Sejak tahun 2004 sampai sekarang penulis menggeluti dunia pendidikan. Penulis pernah mengajar di salah satu universitas swasta besar di Manado, Sulawesi Utara. Saat ini penulis aktif mengajar di salah satu universitas besar di Jakarta Barat. Beberapa artikel ilmiah telah dipublikasikan oleh penulis baik di Jurnal Nasional maupun Jurnal Internasional bereputasi. Penulis tertarik pada bidang riset khususnya Jaringan Saraf Tiruan, Fuzzy Logic, Artificial Intelligence, Image Processing, dan Computer Vision. Buku ini adalah salah satu karya dan semoga kedepannya secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan. Penulis dapat dihubungi melalui email: immasaputro@gmail.com

Tentang Penulis

Neneng Rachmalia Feta, perempuan kelahiran Air Molek Provinsi Riau, 18 Januari 1993**.** Penulis menyelesaikan Pendidikan D-3 di Universitas Telkom Program Studi Manajemen Informatika lulus tahun 2013, kemudian melanjutkan pendidikan S-1 di Universitas Kristen Maranatha Program Studi Teknik Informatika lulus tahun 2015, yang kemudian melanjutkan pendidikan S-2 di Institut Pertanian Bogor Program Magister Jurusan Ilmu Komputer lulus tahun 2018. Penulis telah menggeluti bidang *Software Engineering* dan *Quality Assurance* dari tahun 2013 sampai sekarang. Pada awal tahun 2019 hingga saat ini, penulis mulai aktif mengajar di salah satu perguran tingi swasta di Jakarta. Penulis tertarik pada bidang *Software Engineering, Decision Support System, Fuzzy Logic dan Knowledge Management System*. Buku ini adalah salah satu karya dan semoga kedepannya secara konsisten akan disusul dengan buku-buku berikutnya. Pokok bahasan buku yang ditulis semata-mata untuk berbagi ilmu pengetahuan. Penulis dapat dihubungi melalui kontak personal ke +62-812-9549-9008 atau dapat kontak via email ke : nenengrachmaliafeta@gmail.com.

Tentang Penulis

Diky Wardhani, lahir dan tumbuh di Tangerang, 03 Juli 1991. Mengenyam Pendidikan S1 Sistem Informasi Universitas Islam Negeri Syarif Hidayatullah Jakarta dan S2 Ilmu Komputer Universitas Budi Luhur Jakarta. sejak 2015 penulis sebagai *Graphic/Web Designer* serta *Front-End Developer* dan mulai tahun 2017 sampai saat ini penulis sebagai *Facilitator* di salah satu *Education Technology* di Jakarta. Pada tahun 2018 memiliki ketertarikan dan antusias dalam bidang UI/UX *Research and Design* dan menjadi dosen di perguruan tinggi swasta Jakarta. Buku ini akan menjadi batu loncatan untuk terus berkarya dalam penulisan buku selanjutnya sehingga terus berbagi ilmu pengetahuan yang bermanfaat bagi insan manusia.

Kontak personal : 0856-9710-9666
Email : wardhanidiky@gmail.com

Tentang Penulis

I Gede Agus Suwartane, kelahiran Jakarta, 6 Agustus 1969**.** Penulis menyelesaikan pendidikan S-1 di Universitas Persada Indonesia Y.A.I, Program Studi Teknik Informatika, kemudian melanjutkan pendidikan S-2 di Universitas Budi Luhur Jakarta, Program Studi Magister Ilmu Komputer, dengan konsentrasi Teknologi Sistem Informasi. Sejak tahun 2002 sampai sekarang penulis menggeluti dunia pendidikan. Saat ini, penulis menjadi dosen di Program Studi Sistem Informasi dan Program Studi Informatika Fakultas Teknik Universitas Persada Indonesia Y.A.I. Selain mengajar, penulis juga aktif sebagai instruktur pelatihan di bidang IT dan terlibat dalam beberapa proyek pengembangan IT. Penulis tertarik pada riset, khususnya di bidang *Data Mining*, *Information System*, *Decision Support System*, dan *Business Intelligence*. Penulis telah mempublikasikan beberapa artikel ilmiah di Jurnal Nasional maupun Jurnal Internasional. Buku ini adalah salah satu karya penulis di bidang Sistem Manajemen Basis Data. Pokok bahasan buku yang ditulis, semata-mata untuk berbagi ilmu pengetahuan. Penulis dapat dihubungi melalui email: agus.suwartane@upi-yai.ac.id

Tentang Penulis

Ahmad Rosadi, adalah seorang dosen pada Program Studi Sistem Informasi, Fakultas Teknik, Universitas Persada Indonesia Y.A.I. Dosen kelahiran Indonesia, tanggal 12 Februari 1968, telah menyelesaikan pendidikan S1 pada Jurusan Matematika FMIPA Unpad Indonesia pada tahun 1992, dan S2 di Magister Ilmu Komputer Universitas Indonesia pada tahun 2000. Dosen mempunyai Sertifikat Profesional Pendidik dan juga memiliki sertifikat Profesioanl Certified International Business Intelligense Associate (CIBIA).

Memiliki pengalaman bekerja 12 tahun pada Pusat Litbang Sistem Informasi dan Otomasi Administrasi Negara Lembaga Administrasi Negara RI (tahun 1992 sampai dengan 2004), menjadi Konsultan pengembangan dan perencanaan SI/TI pada beberapa Pemerintah Daerah (*e-Governmen*). Sejak tahun 2006 sampai saat ini menjadi Dosen pada Prodi Sistem Informasi dan Informatika Fakultas Teknik UPI YAI. Mengampu mata kuliah Dasar-sar Pemrograman, Teknologi Web dan Multimedia. Dosen telah menghasilkan publikasi dengan fokus pada bidang Informatika dan Sistem Informasi. Buku ini adalah salah satu bentuk publikasi karya ilmiah dalam Sistem Manajemen Basis Data, disamping buku-buku bidang teknologi informasi lain yang penulis pernah berkontribusi di dalamnya.

Sistem Basis Data adalah buku yang ditujukan untuk memberikan pemahaman komprehensif tentang dunia database. Buku ini mencakup konsep dan teori dasar sistem basis data, termasuk relasi, normalisasi, dan model data. Pembahasan tentang desain database juga diberikan, mulai dari memahami kebutuhan data, menentukan entitas dan atribut, hingga melakukan proses normalisasi.

Buku ini juga membahas tentang implementasi database dengan menggunakan DBMS (Database Management System). Topik ini meliputi pemahaman tentang SQL (Structured Query Language), cara membuat dan mengelola tabel, memanipulasi data, dan membuat query.

Dengan membaca Sistem Basis Data, pembaca akan memperoleh pemahaman yang komprehensif tentang database dan bagaimana mengelola data secara efisien. Buku ini berguna bagi siapa saja yang ingin mempelajari sistem basis data, baik untuk tujuan akademis maupun profesional.

DITERBITKAN OLEH
PT. MIFANDI MANDIRI DIGITAL

Jln Payanibung Ujung D
Dalu Sepuluh-B, Tanjung Morawa
Kab. Deli Serdang Sumatera Utara

ISBN 978-623-09-2563-4
9 786230 925634