SISTEM BASIS DATA

Putri Ariatna Alia S.ST., M.T., Johan Suryo Prayogo, S.Kom., M.T., Erik Yohan Kartiko, S.Pd., M.Kom.,
Dr. Dwi Prasetyo, Dipl.Inf, S.Kom, M.Si., Yuyun Khairunisa, M.Kom.,
Prof. Dr. H. Jufriadif Na'am, S.Kom., M.Kom., CIRRAndi Wijaya, M.Kom.,
Agung Teguh Setyadi, S.Kom., M.Kom., Dwi Remawati, S.Kom.,M.Kom.,
Dr. (H.C.) Zaid Romegar Mair, S.T., M.Cs., Rusina Widha Febriana, S.Kom., M.Kom.,
Radinal Setyadinsa, S.Pd., M.T.I., Asri Maspupah, S.S.T., M.T., Warna Agung Cahyono, S.Kom., MT.

Editor: **Sri Nadriati, S. Kom., M. Kom**

SISTEM BASIS DATA

Putri Ariatna Alia, S.ST., M.T., Johan Suryo Prayogo, S.Kom., M.T., Erik Yohan Kartiko, S.Pd., M.Kom., Dr. Dwi Prasetyo, Dipl.Inf, S.Kom, M.Si., Yuyun Khairunusi, M.Kom., Prof. Dr. H. Jufriadif Na'am, S.Kom., M.Kom., CIRR., Andi Wijaya, M.Kom., Agung Teguh Setyadi, S.Kom., M.Kom., Dwi Remawati, S.Kom., M.Kom., Dr. (H.C) Zaid Romegar Mair, S.T., M.Cs., Rusina Widha Febriana, S.Kom., M.Kom. Radinal Setyadinsa, S.Pd., M.T.I., Asri Maspupah, S.S.T., M.T., Warna Agung Cahyono, S.Kom., MT.

PENAMUDA MEDIA

Sistem Basis Data

Penulis:
Putri Ariatna Alia, S.ST., M.T., Johan Suryo Prayogo, S.Kom., M.T., Erik Yohan Kartiko, S.Pd., M.Kom., Dr. Dwi Prasetyo, Dipl.Inf, S.Kom, M.Si., Yuyun Khairunusi, M.Kom., Prof. Dr. H. Jufriadif Na'am, S.Kom., M.Kom., CIRR., Andi Wijaya, M.Kom., Agung Teguh Setyadi, S.Kom., M.Kom., Dwi Remawati, S.Kom., M.Kom., Dr. (H.C) Zaid Romegar Mair, S.T., M.Cs., Rusina Widha Febriana, S.Kom., M.Kom. Radinal Setyadinsa, S.Pd., M.T.I., Asri Maspupah, S.S.T., M.T., Warna Agung Cahyono, S.Kom., MT.

Editor:
Sri Nadriati, S.Kom., M.Kom

ISBN:
978-623-09-6245-5

Desain Sampul:
Tim PT Penamuda Media

Tata Letak:
Enbookdesign

Diterbitkan Oleh

PT Penamuda Media

Casa Sidoarium RT 03 Ngentak, Sidoarium Dodeam Sleman Yogyakarta

HP/Whatsapp : +6285700592256
Email : penamudamedia@gmail.com
Web : www.penamuda.com
Instagram : @penamudamedia

Cetakan Pertama, Oktober 2023

x + 159, 15x23 cm

KATA PENGANTAR

ALHAMDULILLAH dan Puji syukur dihadiahkan kepada kehadirat Allah Subhanahu wa ta'ala atas rahmat-Nya dikarenakan Buku Sistem Basis Data ini dapat diselesaikan dengan baik. Buku ini disusun sedemikian rupa agar bisa membantu dan menjadi pedoman/panduan pembelajaran untuk mahasiswa/i atau siapa pun yang ingin memahami sistem basis data. Buku ini akan membahas jenis basis data, basis data relational,sistem dan analisis kebutuhan serta banyak lagi tentang pemodelan dan Relasi entitasnya, Serta dilengkapi dengan teori konsep dan contoh kasus penyelesaian masalah menggunakan database MySql.

Terima kasih yang sebesar-besarnya kami ucapkan kepada pihak-pihak yang telah membantu dan mendukung pembuatan buku Sistem Basis Data ini. Besar harapan kami semoga buku ini dapat memberikan manfaat bagi para pembacanya. Buku ini masih jauh dari kesempurnaan. Oleh sebab itu kami membuka kritik dan saran guna untuk menyempurnakan buku ini dalam edisi selanjutnya.

Pekanbaru, Oktober 2023

Sri Nadriati, S. Kom., M. Kom

Editor

DAFTAR ISI

1
MENGENAL BASIS DATA

BASIS data adalah pusat pengelolaan informasi dalam bentuk terstruktur di dalam suatu sistem komputer. Dalam dunia digital yang terus berkembang, basis data adalah fondasi dari hampir semua aplikasi dan sistem informasi.

Basis data (database) adalah kumpulan data yang terorganisir dan disimpan secara sistematis di dalam suatu komputer. Data-data ini disimpan menggunakan struktur khusus agar mudah diakses, dimodifikasi, dan dikelola. Basis data memiliki peran penting dalam menyimpan informasi yang diperlukan oleh aplikasi perangkat lunak, sistem bisnis, dan organisasi.

Peran basis data dalam bisnis yaitu memainkan peran kunci dalam mendukung pengambilan keputusan yang cerdas. Dari pelacakan inventaris hingga analisis pasar, basis data membantu perusahaan menyusun strategi berbasis data yang efektif

Model data adalah desain konseptual dari suatu basis data. Model-model seperti relasional, hierarkis, dan jaringan memiliki karakteristik dan kegunaan masing-masing. Misalnya, model

relasional menggunakan tabel untuk menyimpan data, sementara model hierarkis mengorganisir data dalam struktur pohon.

Sistem basis data harus mendukung transaksi, termasuk penanganan transaksi konkuren, pemulihan setelah kegagalan, dan memastikan konsistensi data meskipun terjadi kesalahan atau kegagalan sistem.

Dalam era Big Data, basis data menjadi pondasi untuk penyimpanan dan analisis data dalam jumlah besar. Dengan teknologi seperti Hadoop dan sistem basis data NoSQL, organisasi dapat mengolah dan mendapatkan wawasan dari volume data yang sangat besar dan beragam.

Tantangan masa depan basis data termasuk mengatasi kompleksitas data, meningkatkan kecepatan akses, dan memanfaatkan kecerdasan buatan untuk analisis data yang lebih mendalam. Integrasi basis data dengan teknologi seperti blockchain juga menjadi fokus penting untuk meningkatkan keamanan dan integritas data.

Berikut adalah elemen-elemen kunci dari sebuah basis data :

1. **Data:**

   Data adalah fakta-fakta yang direkam dan disimpan. Data dalam basis data merujuk pada kumpulan informasi yang terorganisir dengan baik dan disimpan dalam format yang dapat diakses, dikelola, dan diperbarui. Data ini bisa berupa berbagai tipe informasi, termasuk teks, angka, gambar, suara, dan banyak lagi. Dalam konteks basis data, data diorganisir dalam berbagai cara untuk memungkinkan manajemen yang efisien. Contohnya adalah nama, alamat, nomor telepon, dan sebagainya.

2. **Sistem Manajemen Basis Data (DBMS):**

Sistem Manajemen Basis Data adalah perangkat lunak yang memungkinkan pengguna untuk mengakses, menambahkan, mengubah, dan menghapus data dalam basis data. DBMS juga bertanggung jawab atas keamanan data, integritas, dan pemulihan data. Berikut adalah beberapa karakteristik utama dari Sistem Manajemen Basis Data:

a. **Pendefinisian Data:**

DBMS memungkinkan pengguna mendefinisikan struktur data, termasuk jenis data, relasi antara data, dan batasan integritas data. Ini memungkinkan pengguna membuat tabel, indeks, dan konstrain agar data dapat disimpan dengan cara yang terorganisir.

b. **Manipulasi Data:**

DBMS memungkinkan pengguna untuk memasukkan, mengubah, menghapus, dan mencari data dalam basis data. SQL (Structured Query Language) adalah contoh bahasa query yang digunakan untuk manipulasi data dalam sistem basis data relasional.

c. **Kontrol Transaksi:**

DBMS memastikan konsistensi data selama proses transaksi. Sistem ini mendukung konsep transaksi, yang berarti beberapa operasi dapat digabungkan menjadi satu transaksi yang entah berhasil dilakukan sepenuhnya atau gagal dilakukan sepenuhnya.

d. **Optimasi Query:**

DBMS memiliki optimizer query yang memilih jalur eksekusi yang paling efisien untuk query yang diberikan.

Optimizer ini mempertimbangkan berbagai faktor seperti indeks, statistik, dan struktur data untuk meningkatkan kinerja query.

e. Keamanan Data:

DBMS memiliki mekanisme keamanan yang memungkinkan pengguna mengatur siapa yang dapat mengakses basis data dan apa yang dapat mereka lakukan dengan data tersebut. Ini melibatkan kontrol akses, otorisasi, dan autentikasi pengguna.

f. Backup dan Pemulihan:

DBMS menyediakan fasilitas untuk mencadangkan data secara teratur dan memulihkannya jika terjadi kegagalan sistem atau data rusak.

g. Pengelolaan Transaksi:

DBMS mendukung konsep transaksi, yang memungkinkan beberapa operasi dijalankan sebagai satu kesatuan yang tidak dapat dibagi. Ini memastikan konsistensi dan integritas data dalam basis data.

h. Optimasi Kinerja:

DBMS menggunakan berbagai teknik seperti indeks, caching, dan pengaturan konfigurasi untuk meningkatkan kinerja basis data, terutama ketika menangani jumlah data yang besar.

i. Replikasi dan Terdistribusi:

DBMS dapat mendukung replikasi data di beberapa lokasi dan manajemen basis data terdistribusi, memungkinkan pengguna mengakses data dari lokasi yang berbeda secara bersamaan.

j. **Dukungan untuk Bahasa Query:**

DBMS mendukung bahasa query yang memungkinkan pengguna untuk melakukan operasi seperti pengambilan, pembaruan, dan penghapusan data dari basis data.

DBMS adalah komponen kunci dalam infrastruktur teknologi informasi modern dan memainkan peran penting dalam menyimpan, mengelola, dan menyediakan data untuk aplikasi dan pengguna.

3. **Struktur Data:**

Data dalam basis data diorganisir dalam struktur yang terstruktur, seperti tabel dalam basis data relasional. Struktur ini memungkinkan pengguna untuk mengatur data ke dalam baris dan kolom, membuatnya mudah untuk dicari dan dimanipulasi. Struktur ini ditentukan oleh tipe basis data yang digunakan (seperti basis data relasional, basis data NoSQL, basis data grafik, dll).

4. **Kunci dan Relasi:**

Basis data dapat menggunakan konsep kunci (keys) dan relasi untuk mengaitkan data di antara tabel. Kunci adalah cara untuk mengidentifikasi secara unik sebuah catatan dalam tabel, sedangkan relasi menggambarkan cara tabel-tabel berhubungan satu sama lain.

5. **Bahasa Query:**

Bahasa query, seperti SQL (Structured Query Language), digunakan untuk mengakses dan mengelola

data dalam basis data. Dengan SQL, pengguna dapat membuat pertanyaan kompleks untuk mengambil data yang spesifik dari basis data SQL adalah bahasa deklaratif yang memungkinkan pengguna untuk melakukan berbagai operasi pada data, termasuk pengambilan (query), penyisipan, pembaruan, penghapusan, serta pengelolaan struktur basis data seperti tabel dan indeks .

6. **Integritas Data:**

   Integritas data adalah keadaan di mana data dalam basis data akurat, konsisten, dan sesuai dengan aturan yang telah ditetapkan. DBMS memastikan bahwa integritas data dijaga melalui pembatasan akses, validasi, dan konstrain.

7. **Keamanan Data:**

   Keamanan basis data melibatkan pengaturan akses yang tepat untuk pengguna yang berbeda. Ini termasuk pemberian izin kepada pengguna, enkripsi data, serta audit log untuk melacak aktivitas pengguna.

Basis data memiliki berbagai model seperti model relasional, hierarkis, jaringan, dan NoSQL, yang masing-masing memiliki kegunaan dan karakteristiknya sendiri. Basis data memainkan peran vital dalam dunia komputasi modern dengan memungkinkan organisasi menyimpan, mengakses, dan menganalisis data untuk membuat keputusan yang cerdas.

2
JENIS-JENIS BASIS DATA

DALAM dunia teknologi dan informasi, basis data merupakan komponen penting yang digunakan untuk menyimpan, mengelola, dan mengakses informasi. Basis data memungkinkan individu dan organisasi untuk menyimpan data secara efisien, menjaga konsistensi data, dan menggunakan atau mengambil informasi yang relevan sesuai kebutuhan. Terdapat beberapa jenis basis data yang berbeda, yang masing-masing memiliki karakteristik dan penggunaan yang unik. Dalam bab ini akan menjelaskan beberapa jenis basis data yang paling umum digunakan.

A. Basis Data Relasional

Basis data relasional adalah jenis basis data yang paling umum digunakan di seluruh dunia. Hal ini berdasarkan pada model relasional yang diperkenalkan oleh Edgar Codd pada tahun 1970. Dalam basis data relasional, data disimpan dalam bentuk tabel yang terdiri dari baris dan kolom. Kunci primer

(*primary key*) adalah atribut unik yang digunakan untuk mengidentifikasi setiap baris dalam tabel. Kunci asing (*foreign keyi*) adalah atribut dalam satu tabel yang mengacu pada kunci primer tabel lainnya. Kunci primer dan kunci asing digunakan untuk menghubungkan tabel dalam basis data relasional. Operasi dasar pada basis data relasional adalah SELECT, INSERT, UPDATE, dan DELETE.

Adapun kelebihan dari basis data relasional adalah :

1. Operasi SELECT digunakan untuk mengambil data dari tabel, dan dapat menambahkan operasi WHERE untuk menentukan kriteria seleksi tertentu.
2. Operasi INSERT digunakan untuk menambahkan data baru ke dalam tabel.
3. Operasi UPDATE digunakan untuk memperbarui data yang sudah ada.
4. Operasi DELETE digunakan untuk menghapus data dari tabel.

B. Basis Data NoSQL

Basis data NoSQL adalah jenis basis data yang tidak mengikuti model relasional tradisional. Seiring dengan perkembangan teknologi informasi dan pertumbuhan data yang cepat, muncul kebutuhan akan jenis basis data yang lebih fleksibel dan dapat menangani berbagai jenis data yang berbeda. Karena kebutuhan inilah mengapa Basis Data NoSQL menjadi semakin penting dalam dunia komputasi modern. NoSQL singkatan dari “Not Only SQL” adalah kategori basis data yang berbeda dari basis data relasional tradisional. Basis data ini dirancang untuk mengatasi

kekurangan yang mungkin muncul dalam basis data relasional saat mengelola data dengan struktur yang kompleks atau volume yang sangat besar. Beberapa karakteristik utama basis data NoSQL adalah :

1. **Model Data Fleksibel**

   Basis data NoSQL mendukung berbagai model data, seperti dokumen, grafik, kolom, dan lainnya. Basis data NoSQL ini memungkinkan pengembang untuk memilih model yang sesuai dengan kebutuhan aplikasi mereka.

2. **Skalabilitas Horizontal**

   Basis data NoSQL dirancang untuk skalabilitas horizontal, yang memungkinkan untuk menambahkan lebih banyak server atau node untuk mengatasi beban kerja yang lebih besar. Basis data NoSQL ini sangat berguna dalam lingkungan dimana pertumbuhan data sangat cepat.

3. **Konsistensi Fleksibel**

   Sebagian besar basis data NoSQL mendukung model konsistensi yang fleksibel, yang memungkinkan untuk memilih antara konsistensi yang kuat atau konsistensi yang lebih lemah tergantung pada kebutuhan aplikasi.

C. Basis Data Hierarki

Basis data hierarki adalah basis data yang dirancang khusus untuk menyimpan, mengelola, dan memanipulasi data yang memiliki struktur hierarki atau tata susunan berjenjang. Jenis basis data ini sangat relevan dalam konteks pengorganisasian data yang memiliki hubungan induk-anak

atau tingkat keanggotaan. Basis data hierarki mempunyai dua model yaitu :

1. **Model Adjacency List**

   Adalah salah satu pendekatan yang paling sederhana dalam mengimplementasikan hierarki. Setiap node memiliki kolom yang mengacu pada node induknya dengan cara yang mirip dengan relasi foreign key dalam basis data relasional. Model ini cocok untuk hierarki yang tidak terlalu dalam atau tidak terlalu kompleks.

2. **Model Nested Set**

   Model Nested Set menggunakan Teknik penandaan untuk mengatur node dalam hierarki. Setiap node memiliki dua kolom, yaitu “batas kiri” (left boundary) dan “batas kanan” (right boundary). Model ini memungkinkan penanganan hierarki yang sangat dalam dan kompleks, tetapi memerlukan perhatian khusu dalam pemeliharaan dan pemutakhiran.

Beberapa operasi dalam basis data hierarki adalah

1. **Penambahan dan Penghapusan Node**

   Operasi paling dasar dalam basis data hierarki adalah penambahan dan penghapusan node. Ketika menambahkan node, perlu menentukan node induknya. Penghapusan node juga harus mempertimbangkan anak-anak dari node yang akan dihapus.

2. **Pencarian dan Penghapusan Node**

   Pencarian dan navigasi dalam basis data hierarki sangat penting. Kita dapat mencari node berdasarkan

kriteria tertentu atau melakukan navigasi dari tingkat tertentu ke tingkat lain dalam hierarki.

D. Basis Data Objek

Basis data objek adalah jenis basis data yang dirancang khusus untuk menyimpan dan mengelola data yang memiliki struktur objek atau tipe data yang kompleks. Jenis basis data ini sangat relevan dalam konteks penyimpanan data yang tidak hanya terbatas pada data sederhana seperti angka dan teks, tetapi juga mencakup objek seperti gambar, suara, dadn tipe data yang lebih kompleks. Beberapa operasi dalam basis data objek adalah

1. Penyimpanan dan Pemanggilan Objek

   Operasi paling dasar dalam basis data objek adalah penyimpanan dan pemanggilan objek. Kita dapat menyimpan objek dalam basis data dan mengambilnya Kembali berdasarkan kriteria tertentu.

2. Pembaruan dan Penghapusan Objek

   Pembaruan dan penghapusan objek juga penting dalam basis data objek. Kita dapat memperbarui atribut objek atau menghapus objek dari basis data.

E. Basis Data Time-Series

Basis data time-series adalah jenis basis data yang dirancang khusus untuk menyimpan dan mengelola data yang berkaitan dengan waktu atau urutan waktu. Jenis basis data ini sangat relevan dalam konteks analisis dan pemantauan data yang berubah seiring waktu, seperti data

cuaca, data sensor dan data keuangan. Beberapa model basis data time-series adalah :

1. Tabel Time-Series

   Model basis data time-series sering menggunakan tabel untuk menyimpan data time-series. Setiap baris dalam tabel mewakili satu timestamp (penanda waktu yang menunjukkan kapan data dalam time-series dicatat), dan kolom-kolomdalam tabel mewakili berbagai atribut atau dimensi data yang diamati pada waktu tertentu.

2. Struktur Data Khusus

   Dalam beberapa kasus yang ada, basis data time-series dapat menggunakan struktur data khusus yang dioptimalkan untuk penyimpanan dan pengambilan data berdasarkan waktu. Contoh struktur ini adalah rangkaian waktu (real series arrays) dan basis data berbasis kolom yang diorganisasikan berdasarkan timestamp.

F. Basis Data Geografis (GIS)

Basis data geografis atau *Geographical Information System* (GIS), adalah jenis basis data yang dirancang khusus untuk menyimpan, mengelola, dan menganalisis data yang berkaitan dengan lokasi geografis. Jenis basis data ini sangat penting dalam berbagai aplikasi, seperti pemetaan, pemantauan lingkungan, perencanaan kota dan lainnya.

Beberapa model basis data geografis adalah

1. **Model Vektor**

   Model vector menggambarkan data geografis dengan menggunakan titik, garis, dan polygon. Titik digunakan untuk merepresentasikan titik-titik lokasi, garis untuk merepresentasikan fitur linear, dan poligon untuk merepresentasikan area tertentu. Model ini sangat cocok untuk pemodelan data geografis dengan detail tinggi dan objek yang kompleks.

2. **Model Raster**

   Model raster merepresentasikan data geografis dalam bentuk grid atau matriks piksel. Setiap piksel memiliki nilai yang mewakili informasi geografis pada lokasi tersebut. Model ini cocok untuk pemetaan data geografis yang memiliki keterkaitan tinggi dengan data spasial.

Operasi dalam basis data geografis adalah sebagai berikut:

1. **Pemetaan dan Visualisasi**

   Pemetaan dan visualisasi adalah operasi dasar dalam basis data geografis. Kita dapat membuat peta interaktif dan menggambarkan data geografis dala berbagai bentuk, seperti pada peta tematik, peta kontur, atau peta panas (heatmap).

2. **Analisis Geospasial**

   Analisis geospasial memungkinkan untuk melakukan berbagai analisis berdasarkan lokasi geografis. Analisis ini mencakup analisis jarak, analisis pola spasial,

dan pemodelan geostatistik yang dapat membatu dadlam mengambil keputusan dengan didukung data geografis.

G. Basis Data Berorientasi Kolom

Basis data berorientasi kolom adalah jenis basis data yang dirancang khusus untuk penyimpanan dan pengelolaan data dalam bentuk kolom daripada baris seperti pada basis data relasional tradisional. Model basis data berorientasi kolom adalah sebagai berikut :

1. Tabel kolom

   Model basis data berorientasi kolom menggunakan tabel kolom yang mewakili data dalam bentuk kolom, bukan baris. Setiap kolom memiliki tipe data yang spesifik dan data diisi dalam kolom tersebut. Model ini sangat efisien dalam analisi data agregat dan operasi agregasu seperti SUM, AVG, dan COUNT.

2. Kompresi data

   Salah satu keunggulan utama basis data berorientasi kolom adalah kemampuannya untuk melakukan kompresi data. Data dalam kolom dengan tipe data yang sama dapat dikompresi lebih baik daripada data dalam baris yang sering kali berbeda-beda. Model ini menghasilkan pemghematan ruang penyimpanan yang signifikan.

Operasi dalam basis data berorientasi kolom adalah :

1. Query dan Analisis

   Basis data berorientasi kolom sangat cocok untuk operasi query dan analisis yang memerlukan akses

efisien ke data dalam satu atau beberapa kolom. Kita dapat dengan mudah melakukan agregasi, filter, dan analisis statistic terhadap data.

2. **Penambahan dan Penghapusan Data**

Operasi penambahan dan penghapusan data dapat dilakukan dalam bisnis data berorientasi kolom. Data dapat ditambahkan atau dihapus dalam kolom yang sudah ada atau dalam kolom baru sesuai kebutuhan aplikasi

3
BASIS DATA RELASIONAL

A. Pengertian

Basis data relasional adalah sebuah sistem penyimpanan data yang didasarkan pada model data relasional. Model data relasional memanfaatkan tabel sebagai struktur utama dalam melakukan penyimpanan informasi. Model ini menunjukkan cara mengelola/mengorganisasikan data secara fisik dalam memory sekunder, yang akan berdampak pula pada bagaimana kita mengelompokkan data dan membentuk keseluruhan data yang terkait dalam sistem yang kita buat.

Data relasional terdiri dari beberapa tabel yang saling terkait dan memiliki hubungan satu sama lain. Setiap tabel dalam data relasional memiliki baris dan kolom, dengan setiap baris mewakili entitas atau objek tertentu, sedangkan setiap kolom merepresentasikan atribut atau ciri dari entitas tersebut.

B. Keuntungan

Basis data relasional memiliki sejumlah keuntungan yang membuatnya menjadi pilihan yang populer dalam penyimpanan dan pengelolaan data terstruktur. Berikut adalah beberapa keuntungan utama dari basis data relasional:

1. Struktur Terstruktur: Basis data relasional menggunakan tabel dengan baris dan kolom, sehingga data disimpan dalam struktur yang terstruktur dengan baik. Ini memudahkan pemahaman dan pengelolaan data.
2. Konsistensi Data: Model data relasional menerapkan aturan integritas referensial, yang memastikan bahwa data yang ada dalam basis data tetap konsisten dan valid. Hal ini menjaga integritas data selama siklus hidup aplikasi.
3. Normalisasi: Basis data relasional mendukung normalisasi, yang adalah proses desain yang membantu mengurangi redundansi data. Dengan normalisasi yang baik, data disimpan secara efisien, mengurangi risiko inkonsistensi.
4. Hubungan Antar Tabel: Basis data relasional memungkinkan pembuatan hubungan antara tabel melalui kunci asing. Ini memungkinkan data yang terkait untuk dihubungkan dan digunakan bersama, menjadikan desain basis data lebih fleksibel.
5. Pengoptimalan Kinerja: memiliki mekanisme untuk mengoptimalkan kueri, seperti indeks, yang dapat meningkatkan kinerja akses data, terutama pada basis data yang besar.

C. Contoh Tabel

Berikut adalah contoh tabel data relasional yang dapat digunakan untuk menyimpan informasi tentang pelanggan dan pesanan:

Tabel Pelanggan (Customer):

ID Pelanggan	Nama Pelanggan	Alamat	Nomor Telepon
1	John Doe	Jalan ABC No. 1	123-456-7890
2	Jane Smith	Jalan XYZ No. 2	987-654-3210
3	Bob Johnson	Jalan 123 No. 3	555-123-4567

Tabel Pesanan (Order):

ID Pesanan	Tanggal Pesanan	ID Pelanggan	Jumlah Total
101	2023-09-15	1	500.00
102	2023-09-16	2	350.00
103	2023-09-17	1	200.00
104	2023-09-18	3	750.00

Dalam contoh ini, terdapat dua tabel terkait: Tabel Pelanggan dan Tabel Pesanan. Hubungan antara keduanya dapat dilihat melalui kolom "ID Pelanggan" dalam Tabel Pesanan, yang adalah kunci asing yang merujuk ke kolom "ID Pelanggan" dalam Tabel Pelanggan.

Misalnya, pesanan dengan ID 101 dibuat oleh pelanggan dengan ID 1 (John Doe), pesanan dengan ID 102 dibuat oleh pelanggan dengan ID 2 (Jane Smith), dan seterusnya.

Data ini disusun dengan baik dalam bentuk tabel dan memungkinkan Anda untuk menjalankan kueri SQL untuk mengambil informasi yang relevan, seperti daftar pesanan yang dibuat oleh pelanggan tertentu atau total belanjaan dari pesanan-pesanan tersebut.

D. Istilah dalam Model Data Relasional

Dalam model data relasional, terdapat sejumlah istilah dan konsep yang penting untuk dipahami. Berikut adalah beberapa istilah yang umum dalam model data relasional:

1. Tabel (Table): Tabel adalah entitas utama dalam model data relasional yang digunakan untuk menyimpan data. Tabel terdiri dari baris dan kolom, dengan setiap baris mewakili satu entitas atau objek, dan setiap kolom mewakili atribut atau ciri dari entitas tersebut.
2. Kolom (Column): Kolom adalah bagian dari tabel yang merepresentasikan atribut atau ciri-ciri dari entitas. Kolom juga disebut sebagai "field" atau "atribut."
3. Baris (Row): Baris adalah entitas individual dalam tabel. Setiap baris mewakili satu entitas atau objek tertentu dalam basis data.
4. Kunci Primer (Primary Key): Kunci primer adalah satu atau lebih kolom dalam sebuah tabel yang digunakan untuk mengidentifikasi secara unik setiap baris dalam tabel tersebut. Kunci primer memastikan bahwa tidak ada duplikat data dalam tabel.
5. Kunci Asing (Foreign Key): Kunci asing adalah kolom dalam sebuah tabel yang merujuk ke kunci primer dalam tabel lain. Kunci asing digunakan untuk membangun

hubungan antara tabel dan memungkinkan penggabungan data dari tabel yang berbeda.

6. Integritas Referensial (Referential Integrity): Integritas referensial adalah prinsip yang memastikan bahwa kunci asing dalam tabel mengacu pada nilai yang valid dalam tabel referensi. Ini memelihara konsistensi data dalam basis data.
7. Indeks (Index): Indeks adalah struktur yang digunakan untuk mempercepat pencarian dan pengambilan data dalam tabel. Indeks memungkinkan akses cepat ke baris-baris yang memenuhi kriteria tertentu.
8. Normalisasi: Normalisasi adalah proses desain basis data relasional yang bertujuan untuk mengurangi redundansi data dan meningkatkan integritas data. Ini melibatkan pemisahan data yang berulang-ulang ke dalam tabel terpisah untuk menghindari duplikasi informasi.
9. SQL (Structured Query Language): SQL adalah bahasa yang digunakan untuk mengakses, mengelola, dan mengambil data dari basis data relasional. SQL mencakup perintah-perintah seperti SELECT (untuk mengambil data), INSERT (untuk memasukkan data baru), UPDATE (untuk memperbarui data), dan DELETE (untuk menghapus data).
10. Transaksi (Transaction): Transaksi adalah rangkaian operasi yang dijalankan dalam basis data relasional. Transaksi harus memenuhi properti ACID (Atomicity, Consistency, Isolation, Durability), yang memastikan keamanan dan konsistensi data.
11. Skema (Schema): Skema adalah struktur yang mendefinisikan bagaimana tabel dan hubungan antara tabel dalam basis data relasional didefinisikan. Ini

mencakup definisi tabel, kolom, kunci asing, dan kunci primer.

12. Perintah DDL (Data Definition Language): Perintah DDL digunakan untuk mendefinisikan struktur skema basis data, seperti membuat tabel, mengubah skema, atau menghapus tabel.
13. Perintah DML (Data Manipulation Language): Perintah DML digunakan untuk mengelola data dalam tabel, seperti menyisipkan, mengubah, atau menghapus data.
14. View: View adalah representasi virtual dari satu atau lebih tabel dalam basis data. View memungkinkan akses terbatas ke data dan dapat digunakan untuk menyembunyikan informasi yang tidak diperlukan dari pengguna akhir.

Istilah-istilah ini adalah dasar-dasar yang penting untuk memahami dan bekerja dengan basis data relasional dan model data relasional secara keseluruhan.

E. Relasional Key

Relational key adalah konsep penting dalam desain basis data relasional karena mereka membantu menjaga integritas data, mendefinisikan hubungan antar tabel, dan memastikan bahwa setiap baris dalam tabel memiliki identifikasi yang unik. Kunci ini memainkan peran penting dalam keamanan dan konsistensi data dalam basis data relasional. Beberapa jenis kunci yang umum digunakan dalam basis data relasional adalah:

1. **Primary Key (Kunci Primer):**

   Primary key adalah kolom atau kombinasi kolom dalam sebuah tabel yang digunakan untuk mengidentifikasi secara unik setiap baris dalam tabel tersebut.

   Setiap tabel dalam basis data relasional biasanya memiliki satu primary key.

   Primary key memastikan bahwa tidak ada duplikasi data dalam tabel dan menjaga integritas data.

2. **Foreign Key (Kunci Asing):**

   Foreign key adalah kolom dalam sebuah tabel yang merujuk ke primary key dalam tabel lain.

   Foreign key digunakan untuk membangun hubungan antar tabel dalam basis data relasional.

   Ini memungkinkan pembuatan relasi antara data dalam tabel yang berbeda dan memastikan konsistensi data.

3. **Unique Key (Kunci Unik):**

   Unique key adalah kolom atau kombinasi kolom yang memastikan bahwa semua nilai dalam kolom tersebut unik, tetapi tidak diharuskan seperti primary key.

   Tabel dapat memiliki lebih dari satu unique key.

   Ini digunakan untuk memastikan bahwa data tertentu dalam kolom memiliki nilai yang unik.

4. **Candidate Key (Kunci Kandidat):**

   Candidate key adalah kumpulan kolom yang dapat digunakan sebagai primary key karena mereka meme-

nuhi syarat unik dan dapat mengidentifikasi setiap baris secara unik.

Dalam beberapa kasus, Anda dapat memilih salah satu dari beberapa candidate key untuk menjadi primary key.

5. **Super Key (Kunci Super):**

   Super key adalah kombinasi kolom yang memiliki kemampuan untuk mengidentifikasi setiap baris dalam tabel secara unik.

   Ini bisa menjadi kombinasi dari beberapa kolom, termasuk primary key.

6. **Composite Key (Kunci Gabungan):**

   Composite key adalah primary key atau unique key yang terdiri dari lebih dari satu kolom.

   Kombinasi kolom dalam composite key digunakan bersama-sama untuk mengidentifikasi secara unik setiap baris dalam tabel.

F. Relasional Integrity Rules

Relational Integrity Rules (Aturan Integritas Relasional) adalah sekumpulan aturan atau kebijakan yang diterapkan dalam basis data relasional untuk memastikan kualitas, integritas, dan konsistensi data. Tujuan dari aturan integritas relasional adalah untuk memastikan bahwa data dalam basis data tetap akurat dan sesuai dengan struktur data yang telah ditentukan. Terdapat dua jenis aturan integritas relasional yang umumnya diterapkan :

1. **Aturan Integritas Referensial (Referential Integrity Rules):**

   Aturan integritas referensial (RI) digunakan untuk memastikan bahwa hubungan antara tabel dalam basis data relasional dipatuhi dengan benar.

   Aturan ini terkait dengan penggunaan kunci asing (foreign key) dalam tabel. Beberapa aspek utama dari aturan integritas referensial meliputi:

   Pastikan bahwa nilai-nilai dalam kolom kunci asing (foreign key) selalu merujuk ke nilai yang valid dalam kolom kunci primer (primary key) di tabel referensi.

   Melakukan tindakan tertentu ketika terjadi perubahan atau penghapusan data di tabel referensi yang dapat memengaruhi data yang terkait di tabel lain.

   Contoh: Jika Anda memiliki tabel "Orders" dengan kolom "CustomerID" sebagai kunci asing yang merujuk ke tabel "Customers," aturan integritas referensial akan memastikan bahwa setiap "CustomerID" dalam "Orders" sesuai dengan "CustomerID" yang ada dalam tabel "Customers."

2. **Aturan Integritas Domain (Domain Integrity Rules):**

   Aturan integritas domain (DI) berkaitan dengan tipe data dan nilai-nilai yang diizinkan dalam kolom tabel.

   Aturan ini memastikan bahwa setiap kolom memiliki tipe data yang sesuai dan hanya mengizinkan nilai-nilai yang valid sesuai dengan jenis data yang didefinisikan.

   Contoh: Jika sebuah kolom didefinisikan sebagai INTEGER, aturan integritas domain akan memastikan

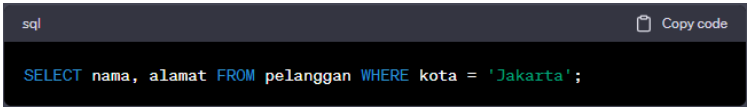
bahwa hanya nilai-nilai bilangan bulat yang valid yang dapat dimasukkan ke dalam kolom tersebut.

G. Bahasa pada Model Data Relasional

Dalam model data relasional, bahasa yang umum digunakan untuk mengelola dan berinteraksi dengan basis data adalah SQL (Structured Query Language). SQL adalah bahasa standar yang digunakan oleh praktis semua sistem basis data relasional untuk melakukan berbagai tugas, seperti mengambil data, memasukkan data baru, memperbarui data, dan menghapus data. Berikut adalah beberapa jenis perintah SQL yang sering digunakan dalam model data relasional:

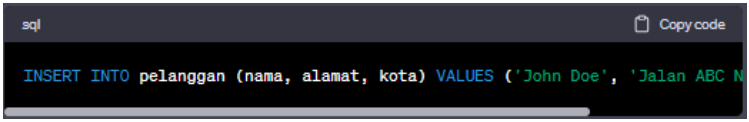
1. SELECT: Perintah SELECT digunakan untuk mengambil data dari satu atau lebih tabel. Ini memungkinkan Anda untuk menentukan kolom mana yang ingin Anda ambil, filter data berdasarkan kriteria tertentu, dan mengurutkan hasil jika diperlukan.

Contoh:



2. INSERT: Perintah INSERT digunakan untuk memasukkan data baru ke dalam tabel.

Contoh:



3. UPDATE: Perintah UPDATE digunakan untuk memperbarui data yang sudah ada dalam tabel.

Contoh:

```sql
UPDATE pelanggan SET kota = 'Surabaya' WHERE nama = 'John Doe';
```

4. DELETE: Perintah DELETE digunakan untuk menghapus data dari tabel.

Contoh:

```sql
DELETE FROM pelanggan WHERE nama = 'John Doe';
```

5. CREATE TABLE: Perintah CREATE TABLE digunakan untuk membuat tabel baru dalam basis data.

Contoh:

```sql
CREATE TABLE produk (
    id_produk INT PRIMARY KEY,
    nama_produk VARCHAR(255),
    harga DECIMAL(10, 2)
);
```

6. ALTER TABLE: Perintah ALTER TABLE digunakan untuk mengubah struktur tabel yang sudah ada, seperti menambahkan kolom baru atau mengganti tipe data kolom.

Contoh:

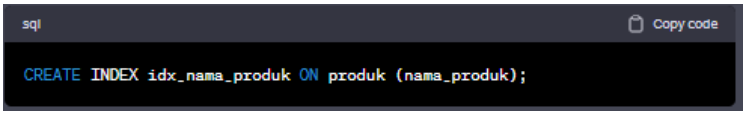
```sql
ALTER TABLE pelanggan ADD nomor_telepon VARCHAR(20);
```

7. DROP TABLE: Perintah DROP TABLE digunakan untuk menghapus tabel dari basis data beserta seluruh data di dalamnya.

Contoh:

8. CREATE INDEX: Perintah CREATE INDEX digunakan untuk membuat indeks pada satu atau lebih kolom dalam tabel, yang dapat meningkatkan kinerja pencarian data.

Contoh:.



4
DEFINISI SISTEM & ANALISIS KEBUTUHAN

BAGIAN ini membahas konsep fundamental dalam pengembangan sistem basis data, yakni Definisi Sistem dan Analisis Kebutuhan. Definisi sistem adalah langkah awal yang penting dalam merancang dan mengembangkan sistem basis data yang efektif. Analisis kebutuhan, di sisi lain, memainkan peran sentral dalam memahami tujuan utama sistem dan mengidentifikasi kebutuhan pengguna. Bab ini akan memandu pembaca melalui konsep-konsep ini secara mendalam, mengilustrasikan pentingnya pemahaman yang tepat tentang kebutuhan pengguna dalam menghasilkan sistem basis data yang sukses.

A. Definisi Sistem

Definisi sistem dalam konteks basis data melibatkan identifikasi komponen utama sistem, interaksi di antara komponen-komponen tersebut, serta tujuan akhir sistem itu sendiri. Sistem basis data adalah entitas yang terdiri dari

perangkat lunak, perangkat keras, data, dan prosedur-prosedur yang saling terkait untuk mengelola dan menyimpan data secara efisien. Definisi yang tepat adalah kunci bagi kelancaran pengembangan selanjutnya.

Dalam konteks basis data, sebuah "sistem" merujuk pada gabungan entitas, proses, dan komponen lainnya yang saling berinteraksi untuk mengelola, menyimpan, dan memanipulasi data. Secara lebih rinci, di dalam basis data, "sistem" dapat merujuk pada:

1. **Perangkat Keras (Hardware):** Ini mencakup semua perangkat fisik seperti server, penyimpanan data, dan perangkat komputer lainnya yang digunakan untuk menyimpan dan mengelola basis data.
2. **Perangkat Lunak (Software):** Ini termasuk sistem manajemen basis data (DBMS), yang merupakan perangkat lunak yang digunakan untuk mengelola basis data. DBMS mengatur akses data, integritas data, keamanan, dan banyak fungsi lainnya yang berhubungan dengan pengelolaan data.
3. **Data Itu Sendiri:** Data adalah informasi yang disimpan dalam basis data. Data ini dapat berupa berbagai jenis informasi, seperti teks, angka, gambar, atau bahkan file multimedia.
4. **Proses Bisnis:** Proses bisnis adalah langkah-langkah atau tugas-tugas yang dilakukan untuk mengambil, memasukkan, mengubah, atau menghapus data dari basis data. Proses ini dapat mencakup penggunaan aplikasi bisnis atau permintaan yang dibuat oleh pengguna.

5. **Pengguna:** Ini adalah individu atau entitas yang berinteraksi dengan basis data untuk membaca, menulis, atau memperbarui data. Pengguna dapat terdiri dari pengguna akhir (seperti pengguna aplikasi) atau administrator basis data.
6. **Aturan dan Kebijakan:** Sistem basis data juga mencakup aturan dan kebijakan yang mengatur bagaimana data dikelola, diakses, dan digunakan. Ini mencakup pembatasan akses, konstrain data, dan kebijakan keamanan.
7. **Jaringan:** Jika basis data diakses melalui jaringan, elemen jaringan juga merupakan bagian dari sistem. Ini mencakup infrastruktur jaringan yang memungkinkan akses ke basis data dari berbagai lokasi.

Sistem basis data dirancang untuk memungkinkan pengelolaan data yang efisien, konsisten, dan aman. Ini memungkinkan entitas atau organisasi untuk menyimpan, mengakses, dan memanfaatkan data mereka sesuai dengan kebutuhan mereka. Sistem basis data yang baik harus memperhatikan aspek-aspek seperti keamanan, integritas data, efisiensi kinerja, dan kemudahan penggunaan.

Sistem basis data mencakup komponen-komponen penting seperti manajemen data, keamanan, integritas data, dan antarmuka pengguna. Pemahaman mendalam tentang aspek-aspek ini diperlukan untuk merancang sistem basis data yang dapat memenuhi kebutuhan bisnis dan teknis.

B. Analisis Kebutuhan

Analisis kebutuhan melibatkan pengumpulan informasi dan pemahaman yang komprehensif tentang apa yang diharapkan dari sistem basis data yang akan dikembangkan. Langkah ini melibatkan interaksi yang cermat dengan para pemangku kepentingan, seperti manajemen, pengguna akhir, dan ahli domain.

Analisis kebutuhan dalam basis data adalah langkah awal yang sangat penting dalam merancang sistem basis data yang efisien dan efektif. Proses ini melibatkan pemahaman mendalam tentang kebutuhan data dan informasi dari organisasi atau entitas yang bersangkutan. Berikut adalah beberapa langkah penting dalam melakukan analisis kebutuhan dalam basis data:

1. **Identifikasi Pengguna dan Stakeholder:**

   Tentukan siapa yang akan menggunakan sistem basis data dan apa peran mereka dalam organisasi.

   Identifikasi pihak-pihak yang memiliki kepentingan dalam penggunaan data, seperti manajemen, departemen, atau entitas eksternal.

2. **Pemahaman Bisnis:**

   Pelajari bisnis atau proses yang akan didukung oleh basis data. Ini melibatkan pemahaman mendalam tentang tujuan organisasi dan bagaimana data mendukung tujuan tersebut.

   Identifikasi masalah atau tantangan yang mungkin diselesaikan oleh basis data.

3. **Pengumpulan Persyaratan:**

   Lakukan wawancara dengan pengguna dan pemangku kepentingan untuk mengidentifikasi kebutuhan mereka terkait data.

   Kumpulkan persyaratan data seperti jenis data yang diperlukan, volume data, frekuensi pembaruan, dan hubungan antar data.

4. **Analisis Proses Bisnis:**

   Identifikasi proses bisnis yang melibatkan penggunaan data. Analisis ini akan membantu dalam pemahaman bagaimana data digunakan dalam organisasi.

   Gambarkan alur kerja dan langkah-langkah dalam proses bisnis untuk mengidentifikasi titik-titik di mana data diperlukan.

5. **Pengenalan Kendala dan Kebijakan:**

   Identifikasi kendala teknis, hukum, dan kebijakan yang mungkin memengaruhi desain dan penggunaan basis data.

   Pastikan kepatuhan terhadap regulasi data yang berlaku, seperti perlindungan data pribadi (GDPR) atau regulasi lainnya.

6. **Pengenalan Kebutuhan Keamanan:**

   Identifikasi kebutuhan keamanan data, termasuk hak akses, enkripsi, dan pemantauan keamanan.

   Tentukan tingkat keamanan yang diperlukan berdasarkan sensitivitas data.

7. **Penentuan Kebutuhan Kinerja:**

   Pertimbangkan aspek kinerja seperti waktu tanggapan yang cepat, skalabilitas, dan toleransi kesalahan.

   Pastikan sistem basis data mampu menangani beban kerja yang diantisipasi.

8. **Dokumentasi Persyaratan:**

   Dokumentasikan semua kebutuhan yang telah diidentifikasi dengan baik. Ini harus menjadi referensi yang jelas untuk seluruh tim pengembangan.

9. **Validasi Persyaratan:**

   Diskusikan dan validasi persyaratan dengan pengguna dan pemangku kepentingan untuk memastikan pemahaman yang benar.

10. **Pemantauan dan Pemutakhiran:**

    Periode pemantauan dan pemutakhiran kebutuhan sesuai dengan perubahan dalam bisnis atau lingkungan teknologi.Analisis kebutuhan yang cermat adalah langkah penting untuk memastikan bahwa basis data dirancang dan dikembangkan sesuai dengan kebutuhan organisasi, yang pada akhirnya akan memastikan efisiensi dan efektivitas pengelolaan data.

Penting untuk mengidentifikasi tujuan utama sistem basis data. Apakah sistem ini akan digunakan untuk pelacakan inventaris, manajemen proyek, atau pelaporan keuangan? Pertanyaan ini membantu dalam mengarahkan pengembangan ke arah yang benar dan mencegah penggunaan sumber daya yang tidak efisien.

Selain itu, analisis kebutuhan melibatkan pengenalan terhadap karakteristik data yang akan dielola oleh sistem. Apa jenis data yang akan disimpan? Bagaimana data ini akan diakses dan dikelola? Pertanyaan-pertanyaan ini membantu dalam merancang struktur basis data yang sesuai.

C. Hubungan antara Definisi Sistem dan Analisis Kebutuhan

Definisi sistem dan analisis kebutuhan adalah dua tahap yang saling terkait dalam pengembangan sistem basis data. Definisi sistem memberikan kerangka kerja untuk memahami bagaimana komponen-komponen berinteraksi dalam sistem, sedangkan analisis kebutuhan membantu dalam mengisi isi dari kerangka kerja ini.

Definisi sistem memberikan arah awal untuk proyek basis data. Ini membantu dalam menentukan lingkup umum dan tujuan dari sistem tersebut. Analisis kebutuhan melengkapi definisi sistem dengan detail yang dibutuhkan. Ini mengidentifikasi persyaratan konkret terkait dengan data, proses bisnis, dan pengguna yang harus diakomodasi oleh basis data.

Informasi yang diperoleh selama analisis kebutuhan membantu dalam merancang struktur data, pemodelan basis data, mengatur tingkat keamanan yang sesuai, dan menentukan kinerja yang diperlukan untuk memenuhi persyaratan pengguna. Analisis kebutuhan juga membantu dalam memastikan bahwa basis data akan efektif dan efisien dalam mendukung operasi organisasi atau proyek yang bersangkutan. Keduanya saling melengkapi: definisi sistem memberikan pandangan umum tentang apa yang akan

dicapai, sementara analisis kebutuhan menguraikan bagaimana sistem akan mencapai tujuan tersebut dengan memenuhi persyaratan yang telah diidentifikasi.

Jadi, definisi sistem adalah langkah pertama dalam merancang basis data, sementara analisis kebutuhan adalah langkah berikutnya yang lebih mendalam dan detail untuk merinci persyaratan spesifik yang harus dipenuhi oleh basis data. Keduanya bekerja bersama untuk mengarahkan pengembangan sistem basis data yang sukses.

Ketika definisi sistem dan analisis kebutuhan dilakukan dengan cermat, hasilnya adalah panduan yang kuat untuk merancang, mengembangkan, dan mengimplementasikan sistem basis data. Definisi yang jelas membantu tim pengembang fokus pada tujuan akhir, sementara analisis kebutuhan memastikan bahwa sistem memenuhi harapan dan kebutuhan pengguna.

5
PEMODELAN RELASI ENTITAS

PEMODELAN Relasi Entitas *(Entity-Relationship Modeling)*, sering disingkat sebagai ERD *(Entity-Relationship Diagram)*, merupakan sebuah metode atau teknik dalam mendisain basis data yang digunakan untuk menggambarkan entitas (objek atau konsep), atribut (informasi yang terkait dengan entitas tersebut), dan hubungan antara entitas dalam suatu sistem atau organisasi. Bab ini akan membahas bagaimana mengidentifikasi dan memodelkan entitas serta hubungan di antara mereka. Pemahaman yang kuat tentang pemodelan relasi entitas adalah langkah pertama yang krusial dalam mengembangkan sistem basis data yang efisien dan efektif.

A. Entitas *(Entity)*

Entitas adalah objek atau konsep yang memiliki makna dan dapat diidentifikasi secara unik dalam lingkungan yang diberikan. Dalam pemodelan basis data, entitas sering kali

mewakili objek-objek dunia nyata, seperti pelanggan, produk, karyawan, atau pesanan. Pemahaman yang mendalam tentang entitas adalah kunci untuk merancang struktur basis data yang akurat.

B. Atribut (Attribute)

Atribut adalah karakteristik atau properti yang terkait dengan sebuah entitas. Atribut menggambarkan informasi yang ingin kita simpan tentang entitas tersebut. Misalnya, atribut untuk entitas "Pelanggan" bisa mencakup nama, alamat, nomor telepon, dan lainnya. Identifikasi dengan cermat atribut-atribut yang relevan adalah langkah penting dalam pemodelan relasi entitas.

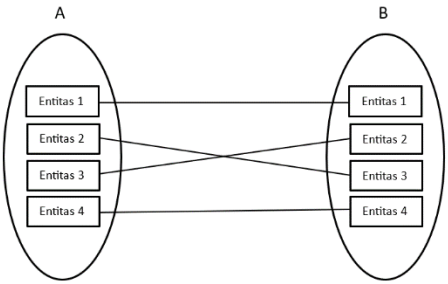
C. Identifikasi

Setiap entitas harus memiliki suatu atribut yang dapat digunakan untuk mengidentifikasinya secara unik dalam basis data. Atribut ini disebut sebagai "kunci primer" atau "primary key." Identifikasi entitas dengan benar adalah penting untuk memastikan setiap baris data dalam basis data memiliki nilai kunci primer yang berbeda.

D. Hubungan *(Relationship)* antar entitas

Relationship menggambarkan jumlah entitas yang terlibat dalam hubungan antara entitas. Kardinalitas dapat berupa satu-ke-satu *(one to one)*, satu-ke-banyak *(one to many)*, atau banyak-ke-banyak *(many to many),* yang menunjukkan seberapa banyak entitas yang dapat terlibat dalam hubungan tertentu.

1. Satu-ke-Satu *(One-to-One)*: Satu entitas dalam satu sisi hubungan terkait dengan satu entitas dalam sisi lain hubungan.

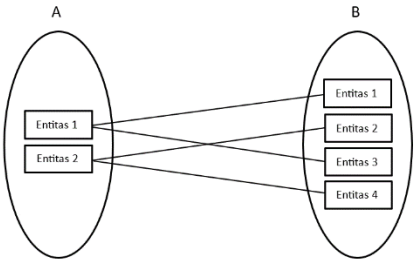


Tantangan dalam Penggunaan Relasi *One-to-One*:

a. Pemilihan yang Tepat: Penting untuk memilih kasus penggunaan yang benar-benar memerlukan relasi *one-to-one.* Terlalu banyak relasi *one-to-one* dalam desain basis data dapat membingungkan dan mempersulit pemahaman.
b. Kesesuaian: Pastikan bahwa relasi *one-to-one* benar-benar sesuai dengan kebutuhan aplikasi atau sistem yang Anda desain. Jika hubungan antara dua entitas dapat berubah menjadi *one-to-many* di masa depan, Anda mungkin perlu mempertimbangkan desain yang lebih fleksibel.

Relasi *one-to-one* adalah komponen penting dalam desain basis data yang memungkinkan pengelolaan data dengan efisien dan akurat. Dalam banyak kasus, relasi ini membantu menjaga integritas data dan memperbaiki efisiensi penyimpanan data.

2. Satu-ke-Banyak *(One-to-Many)*: Satu entitas dalam satu sisi hubungan terkait dengan banyak entitas dalam sisi lain hubungan.

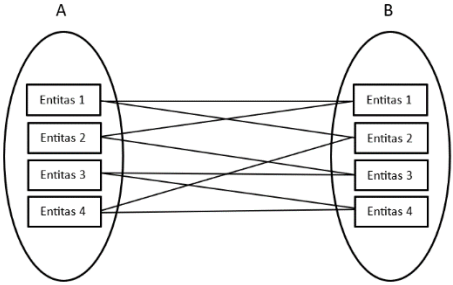


Tantangan dalam Penggunaan Relasi *One-to-Many*:

a. Pengelolaan dan Pemeliharaan: Relasi *one-to-many* memerlukan manajemen yang baik untuk memastikan konsistensi data. Anda perlu memahami bagaimana memasukkan, menghapus, dan memperbarui data dengan benar dalam hubungan ini.
b. Kesesuaian: Pastikan bahwa relasi *one-to-many* memang sesuai dengan kebutuhan aplikasi atau sistem yang Anda desain. Jika ada kemungkinan bahwa hubungan ini akan berkembang menjadi hubungan yang lebih kompleks di masa depan, Anda perlu mempertimbangkan desain yang lebih fleksibel.

Relasi *one-to-many* adalah salah satu jenis hubungan yang paling umum dalam pemodelan basis data. Mereka memungkinkan Anda untuk menggambarkan hubungan hierarkis antara entitas dan mengelola data dengan efisien dalam berbagai konteks aplikasi, termasuk manajemen pelanggan, logistik, perpustakaan, dan banyak lagi.

3. Banyak-ke-Banyak *(Many-to-Many)*: Banyak entitas dalam satu sisi hubungan terkait dengan banyak entitas dalam sisi lain hubungan.



Tantangan dalam Penggunaan Relasi *Many-to-Many*:

a. Desain yang Kompleks: Pengelolaan relasi *many-to-many* sering kali memerlukan desain yang lebih kompleks, dengan penggunaan tabel perantara. Hal ini dapat meningkatkan kompleksitas basis data dan pemahaman yang diperlukan.
b. Pemeliharaan Data: Memasukkan, menghapus, atau memperbarui data dalam relasi *many-to-many* memerlukan manajemen yang baik agar konsistensi data tetap terjaga.
c. Pemahaman yang Mendalam: Penggunaan relasi *many-to-many* memerlukan pemahaman yang kuat tentang bagaimana mengakses dan mengelola data melalui tabel perantara.

Relasi *many-to-many* penting dalam pemodelan basis data karena memungkinkan representasi yang akurat dari banyak situasi dalam dunia nyata di mana berbagai entitas terhubung dengan banyak entitas lainnya. Dengan manajemen yang tepat, relasi *many-to-*

many memungkinkan pemodelan dan pengelolaan data yang efisien dan akurat.

E. Diagram ERD (Entity-Relationship Diagram)

Entity-Relationship Diagram (ERD) adalah alat grafis yang digunakan untuk menggambarkan entitas, atribut, dan hubungan antara entitas dalam suatu basis data. ERD membantu kita memvisualisasikan struktur basis data dengan jelas dan merinci.

Manfaat ERD dalam Pemodelan Basis Data:

1. Visualisasi Struktur: ERD membantu pemodeler dan pemangku kepentingan untuk memvisualisasikan dengan jelas struktur basis data dan bagaimana entitas saling berhubungan.
2. Perencanaan Hubungan: ERD membantu dalam perencanaan dan pemahaman hubungan antara entitas, yang penting untuk merancang basis data yang efisien.
3. Identifikasi Kunci Primer: ERD membantu dalam mengidentifikasi atribut yang harus digunakan sebagai kunci primer, sehingga setiap baris data dapat diidentifikasi secara unik.
4. Analisis Keseluruhan: Dengan melihat ERD, pemodeler dapat melakukan analisis keseluruhan tentang bagaimana data akan disimpan, diakses, dan dimanipulasi dalam basis data.

F. Langkah-Langkah dalam Pemodelan Relasi Entitas

Kita akan membahas langkah-langkah praktis dalam merancang basis data dengan memodelkan entitas dan hubungan di antara mereka. Ini mencakup identifikasi entitas, menentukan atribut, merancang kunci primer, dan menggambarkan ERD. Langkah-langkah dalam pemodelan relasi entitas adalah serangkaian tindakan yang harus diikuti saat merancang struktur basis data yang melibatkan identifikasi entitas, atribut, dan hubungan di antara mereka. Berikut adalah langkah-langkah yang dapat Anda ikuti dalam pemodelan relasi entitas:

1. **Identifikasi Entitas:**

   Identifikasi objek-objek atau konsep-konsep yang akan direpresentasikan dalam basis data. Setiap objek ini akan menjadi entitas. Pastikan setiap entitas dapat diidentifikasi secara unik dan memiliki makna yang jelas dalam konteks sistem atau aplikasi yang Anda rancang.

2. **Definisikan Atribut:**

   Tentukan atribut-atribut yang terkait dengan setiap entitas. Atribut adalah karakteristik atau properti dari entitas. Pastikan atribut-atribut yang Anda pilih mencerminkan informasi yang penting dan relevan untuk entitas tersebut.

3. **Identifikasi Kunci Primer:**

   Tentukan atribut yang akan digunakan sebagai kunci primer untuk setiap entitas. Kunci primer adalah atribut yang digunakan untuk mengidentifikasi secara unik

setiap baris data dalam entitas tersebut. Pastikan kunci primer yang Anda pilih memiliki nilai yang unik dan berhubungan langsung dengan entitas.

4. **Tentukan Hubungan:**

   Identifikasi hubungan antara entitas-entitas yang Anda telah definisikan. Hubungan ini menggambarkan bagaimana entitas saling berinteraksi dalam basis data. Tentukan jenis hubungan antara entitas-entitas, seperti satu-ke-satu (one-to-one), satu-ke-banyak (one-to-many), atau banyak-ke-banyak (many-to-many).

5. **Tentukan Kardinalitas:**

   Tentukan kardinalitas hubungan, yaitu berapa banyak entitas yang dapat terhubung melalui hubungan tersebut. Misalnya, dalam hubungan satu-ke-banyak, Anda harus menentukan apakah setiap entitas dalam entitas pertama hanya dapat terhubung ke satu atau banyak entitas dalam entitas kedua.

6. **Buat Diagram Entitas-Relasi (ERD):**

   Gunakan ERD untuk menggambarkan secara visual entitas, atribut, kunci primer, dan hubungan antara entitas-entitas. ERD membantu Anda memahami struktur basis data secara keseluruhan dan membantu dalam komunikasi dengan tim pengembangan atau pemangku kepentingan.

7. **Normalisasi Basis Data:**

   Terapkan aturan normalisasi untuk mengoptimalkan struktur basis data dan mengurangi redundansi data.

Normalisasi membantu dalam menghindari masalah seperti duplikasi data dan anomali data.

8. **Validasi Desain:**

   Validasi desain basis data dengan berdiskusi dengan anggota tim pengembangan dan pemangku kepentingan untuk memastikan bahwa struktur basis data mencerminkan kebutuhan bisnis atau aplikasi.

9. **Implementasi Basis Data:**

   Setelah desain basis data selesai, langkah berikutnya adalah mengimplementasikan basis data tersebut menggunakan perangkat lunak basis data yang sesuai (seperti MySQL, PostgreSQL, atau MongoDB).

10. **Uji dan Perbaiki:**

    Setelah implementasi, uji basis data untuk memastikan bahwa semuanya berfungsi sesuai rencana.Perbaiki masalah atau kekurangan yang mungkin muncul selama pengujian.

11. **Dokumentasi Basis Data:**

    Buat dokumentasi yang lengkap mengenai struktur basis data, tabel, atribut, hubungan, dan kunci primer. Dokumentasi ini akan berguna untuk pemeliharaan dan pemahaman sistem di masa mendatang. Pemahaman yang kuat tentang langkah-langkah dalam pemodelan relasi entitas adalah kunci untuk merancang basis data yang efisien, akurat, dan sesuai dengan kebutuhan bisnis atau aplikasi Anda. Langkah-langkah ini membantu memastikan bahwa data dapat disimpan, diakses, dan dimanipulasi dengan baik dalam basis data Anda.

6
NORMALISASI BASIS DATA

A. Pengertian

Normalisasi dalam konteks basis data merujuk pada proses pengorganisasian data dalam tabel-tabel agar data-data tersebut tetap konsisten, efisien, dan menghindari redundansi (pengulangan data) serta menghilangkan ambiguitas pada tabel yang ada pada basis data. Sehingga basis data yang telah ternormalisasi adalah basis data yang berkualitas baik. Tujuan normalisasi adalah untuk mengurangi atau menghilangkan anomali dalam basis data, seperti anomali pembaruan, anomali penghapusan, dan anomali penyisipan. Dengan normalisasi yang baik, data akan lebih mudah dipelihara, diperbarui, dan dikelola.

B. Tujuan dan Manfaat

Tujuan utama dari normalisasi dalam basis data adalah untuk memastikan integritas data dan mengurangi redundansi data. Normalisasi mencapai tujuan ini dengan mengatur data dalam tabel-tabel yang lebih kecil dan lebih terfokus, memisahkan data menjadi entitas-entitas yang berbeda, dan menghilangkan anomali data. Berikut adalah beberapa tujuan kunci dari normalisasi dalam database:

1. **Mengurangi Redundansi**: Normalisasi menghindari redundansi data dengan memisahkan data yang bersifat berulang ke dalam tabel-tabel terpisah. Ini menghemat ruang penyimpanan dan mengurangi peluang terjadinya inkonsistensi dalam data.
2. **Menghindari Anomali Pembaruan (Update)**: Dengan normalisasi yang baik, pembaruan data hanya perlu dilakukan di satu tempat, karena data yang sama tidak disimpan di beberapa tempat. Ini mencegah terjadinya anomali pembaruan, di mana perubahan data harus dilakukan di beberapa lokasi yang dapat menyebabkan inkonsistensi.
3. **Menghindari Anomali Penghapusan (Delete)**: Normalisasi memastikan bahwa menghapus data dari satu tabel tidak menghapus data penting lainnya. Tanpa normalisasi, penghapusan data dari satu tempat dapat menghapus data yang relevan di tempat lain.
4. **Menghindari Anomali Penyisipan (Insert)**: Normalisasi membantu menghindari anomali penyisipan, di mana penyisipan data baru dapat gagal atau menghasilkan data yang tidak konsisten. Dengan normalisasi yang baik,

penyisipan data baru selalu dilakukan ke dalam tabel yang tepat.

5. **Meningkatkan Kinerja Query**: Normalisasi dapat memungkinkan query yang lebih efisien karena data yang disimpan dalam tabel-tabel yang lebih kecil dan lebih terfokus, dan indeks dapat digunakan secara lebih efektif.
6. **Mendukung Skalabilitas**: Basis data yang ternormalisasi dapat lebih mudah diperbarui dan dikelola saat aplikasi berkembang dan kebutuhan bisnis berubah. Ini mendukung skalabilitas sistem.
7. **Memastikan Integritas Referensial**: Dengan normalisasi yang baik, integritas referensial (hubungan antara tabel) dapat dijaga dengan baik, sehingga data tetap konsisten dan tidak ada data yang "tertumpang tindih" di antara tabel.
8. **Meningkatkan Pemahaman dan Desain Basis Data**: Normalisasi membantu dalam pemodelan yang lebih baik dan lebih rapi, sehingga pemahaman tentang struktur basis data meningkat dan desain basis data menjadi lebih efisien.

Dalam beberapa kasus, normalisasi dapat berlebihan sehingga menghasilkan basis data yang terlalu kompleks untuk aplikasi sederhana. Oleh karena itu, perlu diperhatikan keseimbangan antara normalisasi dan denormalisasi, tergantung pada kebutuhan spesifik aplikasi dan kinerja yang diinginkan.

C. Bentuk Normalisasi dan Contohnya

Untuk melakukan normalisasi basis data kita harus memahami data seperti apa yang akan disimpan. Berikut adalah bentuk-bentuk normalisasi basis data yang sering digunakan berserta contohnya:

1. **Bentuk Tidak Normal (Unnormalize)**: Bentuk tidak normal dalam basis data mengacu pada struktur data yang belum mengikuti prinsip normalisasi, yang dapat menghasilkan redundansi data dan potensi masalah integritas data. Salah satu contoh bentuk tidak normal yang umum adalah tabel dengan redundansi data. Berikut adalah contoh bentuk tidak normal dalam basis data :

ID Pelanggan	Nama Pelanggan	Alamat Pelanggan	Pesanan
1	John	Jalan A, Kota X	Produk A, B
2	Maria	Jalan B, Kota Y	Produk B, C
3	Ahmad	Jalan C, Kota Z	Produk A, C

2. **1NF (First Normal Form)**: Bentuk tahap pertama dalam proses normalisasi basis data. Pada tahap ini, tabel diorganisasi sedemikian rupa sehingga setiap sel dalam tabel berisi satu nilai atomik, dan tidak ada kolom yang berisi nilai-nilai yang berulang atau bersifat multivalued. Berikut adalah contoh tabel yang tidak memenuhi 1NF dan tabel yang telah dinormalisasi menjadi 1NF:

Tabel Tidak Memenuhi 1NF:

ID	Nama	Bahasa Favorit
1	John	Inggris, Spanyol
2	Maria	Italia
3	Ahmad	Arab, Inggris

Pada tabel di atas, kolom "Bahasa Favorit" berisi nilai-nilai yang berulang dan bersifat multivalued. Untuk mengubahnya menjadi 1NF, kita perlu memisahkan nilai-nilai bahasa favorit ke dalam baris-baris yang berbeda sehingga setiap sel berisi satu nilai:

Tabel dalam 1NF:

ID	Nama	Bahasa Favorit
1	John	Inggris
1	John	Spanyol
2	Maria	Italia
3	Ahmad	Arab
3	Ahmad	Inggris

Dengan tabel yang telah diubah menjadi 1NF, setiap sel hanya berisi satu nilai atomik, dan tidak ada nilai yang berulang atau bersifat multivalued. Ini memungkinkan struktur data yang lebih terorganisir dan memenuhi syarat pertama dalam normalisasi.

3. **2NF (Second Normal Form)**: Bentuk tahap kedua dalam proses normalisasi basis data. Pada tahap ini, kita memastikan bahwa setiap atribut (kolom) non-kunci dalam tabel sepenuhnya bergantung pada seluruh kunci utama. Ini membantu menghindari dependensi parsial yang mungkin ada dalam tabel. Untuk memahami 2NF, mari kita lihat contoh tabel yang belum memenuhi 2NF, dan kemudian contoh tabel yang telah diubah menjadi 2NF.

Contoh: Tabel Tanpa 2NF

Misalkan kita memiliki tabel berikut yang mencatat informasi pesanan pelanggan:

No. Pesanan	ID Pelanggan	Nama Pelanggan	Alamat Pelanggan	Produk	Jumlah
1	1	John	Jalan A, Kota X	Produk A	5
2	1	John	Jalan A, Kota X	Produk B	3
3	2	Maria	Jalan B, Kota Y	Produk A	2
4	3	Ahmad	Jalan C, Kota Z	Produk B	4

Dalam tabel di atas, kolom "Nama Pelanggan" dan "Alamat Pelanggan" hanya bergantung pada "ID Pelanggan." Karena kunci utama adalah kombinasi "No. Pesanan" dan "ID Pelanggan," ini menyebabkan dependensi parsial, di mana atribut "Nama Pelanggan" dan "Alamat Pelanggan" bergantung pada sebagian dari kunci utama.

Maka harus dirubah menjadi 2 tabel dalam agar memenuhi 2NF.

Tabel "Pelanggan":

ID Pelanggan	Nama Pelanggan	Alamat Pelanggan
1	John	Jalan A, Kota X
2	Maria	Jalan B, Kota Y
3	Ahmad	Jalan C, Kota Z

Tabel "Pesanan":

No. Pesanan	ID Pelanggan	Produk	Jumlah
1	1	A	5
2	1	B	3
3	2	A	2
4	3	B	4

4. **3NF (Third Normal Form)**: Bentuk tahap ketiga dalam proses normalisasi basis data. Pada tahap ini, selain memenuhi persyaratan 1NF dan 2NF, kita memastikan bahwa tidak ada dependensi transitif dalam tabel. Dengan kata lain, atribut non-kunci tidak boleh bergantung pada atribut non-kunci lainnya. Untuk memahami 3NF, mari kita lihat contoh tabel yang belum memenuhi 3NF, dan kemudian contoh tabel yang telah diubah menjadi 3NF.

   Contoh tabel tanpa 3NF:

ID Guru	Nama Guru	Mata Pelajaran	Ruang Kelas
1	Mr. A	Matematika	A101
2	Mr. B	Fisika	B202
3	Mr. C	Kimia	A101
4	Mr. D	Matematika	C303

Dalam tabel di atas, kolom "Ruang Kelas" bergantung pada "Mata Pelajaran." Sebagai contoh, "Matematika" selalu terkait dengan "A101." Ini adalah dependensi transitif, karena "Ruang Kelas" tidak bergantung pada kunci utama tabel, yaitu "ID Guru."

Tabel dalam 3NF:

Untuk mengubah tabel di atas menjadi 3NF, kita dapat memisahkan informasi tentang mata pelajaran dan ruang kelas ke dalam tabel terpisah dengan kunci utama yang sesuai. Ini menghindari dependensi transitif:

Tabel "Guru":

ID Guru	Nama Guru
1	Mr. A
2	Mr. B
3	Mr. C
4	Mr. D

Tabel "Mata Pelajaran":

Mata Pelajaran	Ruang Kelas
Matematika	A101
Fisika	B202
Kimia	A101

Tabel "Ajaran":

ID Guru	Mata Pelajaran
1	Matematika
2	Fisika
3	Kimia
4	Matematika

5. **BCNF (Boyce-Codd Normal Form)**: Bentuk normalisasi yang lebih ketat daripada 3NF (Third Normal Form). Dalam BCNF, setiap atribut non-kunci dalam tabel harus bergantung sepenuhnya pada kunci utama, dan tidak boleh ada dependensi fungsional non-trivial antara atribut non-kunci. BCNF sangat fokus pada mengatasi dependensi fungsional yang dapat menyebabkan anomali data. Untuk memahami BCNF, mari kita lihat contoh tabel yang belum memenuhi BCNF dan kemudian contoh tabel yang telah diubah menjadi BCNF.

   Tabel Tanpa BCNF

No. Pesanan	Nama Pelanggan	Produk	Harga
1	John	Produk A	100
1	John	Produk B	150
2	Maria	Produk A	100
3	Ahmad	Produk B	150

   Dalam tabel di atas, atribut "Harga" bergantung pada kombinasi "Produk" dan "Nama Pelanggan," yang bukan merupakan kunci utama. Ini adalah dependensi fungsional non-trivial yang melanggar BCNF.

   Tabel dalam BCNF

   Untuk mengubah tabel di atas menjadi BCNF, kita harus memisahkan informasi produk dan harga ke dalam tabel terpisah dengan kunci utama yang sesuai:

Tabel "Pesanan":

No. Pesanan	Nama Pelanggan
1	John
2	Maria
3	Ahmad

Tabel "Detail Pesanan":

No. Pesanan	Produk	Harga
1	Produk A	100
1	Produk B	150
2	Produk A	100
3	Produk B	150

Dengan pemisahan ini, kita telah memastikan bahwa setiap atribut non-kunci ("Harga") sepenuhnya bergantung pada kunci utama ("No. Pesanan" dan "Produk"). Ini memenuhi persyaratan BCNF. BCNF membantu memastikan bahwa tabel tidak memiliki dependensi fungsional yang kompleks dan menghindari masalah anomali data yang mungkin terjadi akibatnya.

7
ENHANCED ENTITY RELATIONSHIP

A. Pengertian Model EER (Enhanced Entity Relationship)

Model *Enhanced Entity Relationship* (EER) adalah perluasan dari model *Entity Relationship* (ER) yang digunakan dalam desain basis data. EER menawarkan alat dan ide yang lebih kuat untuk menggambarkan hubungan antar entitas dalam basis data. EER adalah hubungan entitas yang menggabungkan kemampuan semantiknya dengan konsep yang lebih kompleks(Darmanto, 2016). Berikut adalah beberapa karakteristik utama dari hubungan entitas yang ditingkatkan:

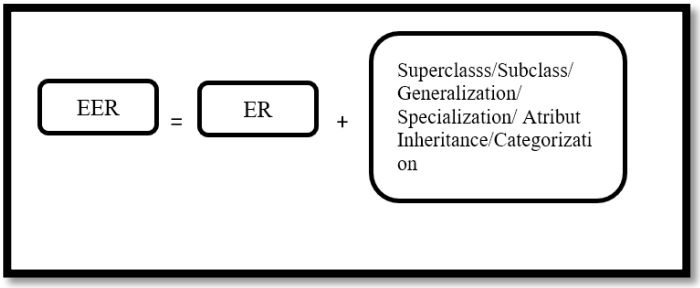
1. Entitas : Seperti pada model ER, enitas dalam EER menggambarkan objek atau konsep dalam dunia nyata yang memiliki atribut yang mendefinisikan karakteristiknya. Contohnya dalam basis data untuk

sistem perpustakaan, entitas utama bisa menjadi “Buku” dengan atribut judul, penulis dan ISBN

2. Atribut : Atribut adalah properti atau karakteristik yang menggambarkan entitas. Dalam EER, atribut dapat memiliki tipe data yang lebih kompleks, seperti daftar atau himpunan nilai. Misalnya atribut “Penulis” pada entitas “Buku” dapat memiliki tipe data daftar yang mencakup beberapa penulis.
3. Relasi : EER mendukung berbagai jenis relasi antara entitas, seperti relasi satu-satu, satu-banyak, dan banyak-banyak. Misalnya hubungan antara entitas “Peminjam” dan “Buku” dengan menggambarkan bahwa satu peminjam dapat meminjam banyak buku (relasi satu-banyak).
4. Spesialisasi dan Generalisasi : EER memungkinkan untuk menggambarkan hierarki antar entitas dengan menggunakan konsep spesialisasi dan generalisasi. Misalnya enitias “Kendaraan” yang kemudian dapat diperinci menjadi entitas khusus seperti “Mobil” dan “Kendaraan”.
5. Karakteristik Khusus : EER memungkinkan entitas khusus dalam hierarki untuk memiliki atribut atau hubungan tambahan yang tidak dimiliki oleh entitas umumnya. Misalnya enitias “Mobil” dalam hierarki “Kendaraan” dapat memiliki atribut tambahan seperti “Jumlah Pintu”.
6. Agregasi : Dapat digambarkan dengan entitas yang terkait dalam suatu agregat, yang dapat memiliki atribut dan hubungan sendiri. Misalnya, entitas “Pesanan” dapat menjadi agregat yang terdiri dari entitas “Barang” dan “Pelanggan”

7. Atribut Turunan : EER memungkinkan untuk mendifinisikan atribut yang nilainya dihitung berdasarkan atribut lain dalam basis data. Misalnya, entitas Pegawai memiliki atribut turunan seperti "Usia" yang dihitung berdasarkan atribut "Tanggal Lahir"
8. Multi-Valued Atrributes : Entitas tertentu memiliki atribut dimana Atribut tersebut dapat memiliki lebih dari satu nilai. Misalnya, entitas "Mobil" dapat memiliki atribut "Fitur" yang dapat memiliki beberapa nilai seperti "AC","Radio" dan "CD Player".

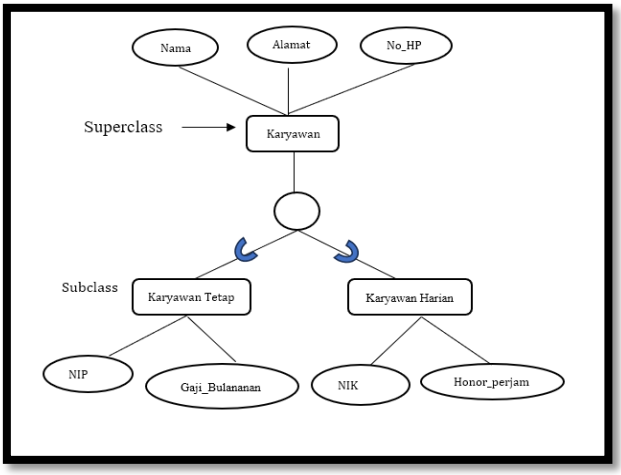
B. Konsep Model EER (Enhanced Entity Relationship)



1. Superclass

Merupakan Entitas yang bersifat umum dan merupakan induk dari subclass-subclassnya.

Contoh : Entitas Karyawan merupakan superclass (Gambar 1)



Gambar 1. Superclass dan Subclass

Keterangan :

a. Notasi dasar yang digunakan untuk superclass digambarkan dengan penghubung lingkaran pada garis.
b. Garis yang menghubungkan dengan arah ke subclass diberi simbol berbentuk U.
c. Arah pewarisan dari superclass ke subclass juga menggunakan simbol U.

2. Subclass

Merupakan Entitas yang mempunyai atribut atau relasi tambahan yang berbeda atau terspesialisasi.

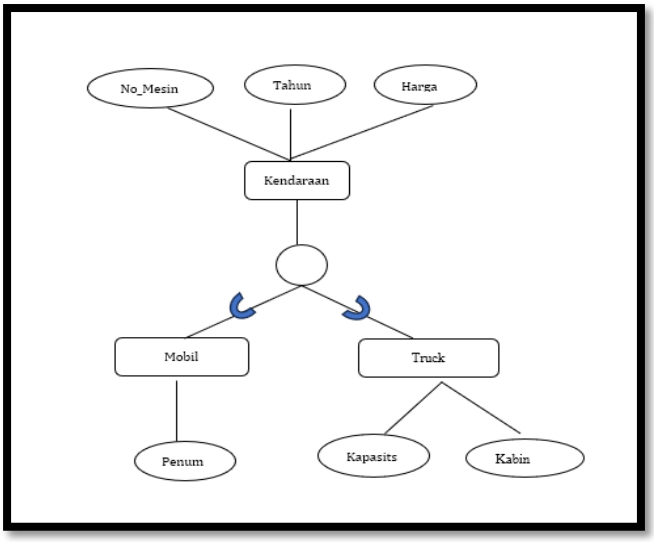
Contoh : entitas Karyawan mempunyai beberapa subclass seperti Karyawan Tetap dan Karyawan Honorer (Gambar 1).

3. Generalization

Dipandang sebagai kebalikan dari proses spesialisasi, generalisasi adalah proses penggabungan

subclass-subclass menjadi suatu entitas yang lebih umum. Proses pendefinisian entitas-entitas yang disatukan menjadi satu entitas superclass dari entitas aslinya yang merupakan subclass istimewa disebut dengan Generalisasi.

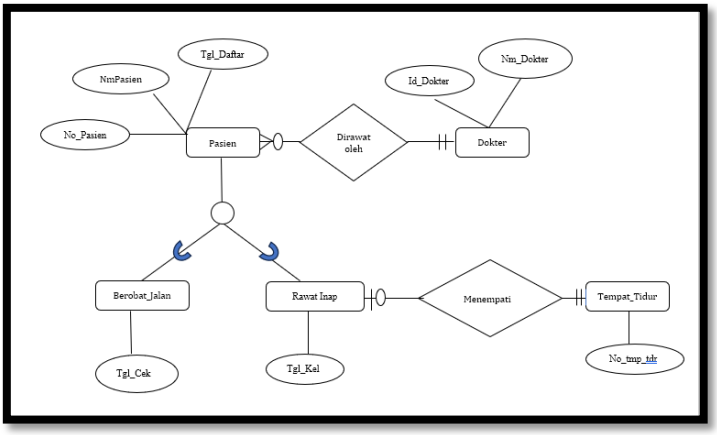
Contoh : Generalisasi dari Mobil dan Truck menjadi Kendaraan.



Gambar 2. Generalization

4. Atribut Inheritance

Merupakan pewarisan sifat dari kelas superiornya. Superclass akan mewariskan semua atribut entity pada anggota subclass . Pewaris, atau pewarisan, adalah karakteristik yang sangat penting karena membuat atribut-atribut entity superclass tidak perlu lagi ditulis sebagai atribut subclass. Ini mengurangi redundansi yang tidak perlu.

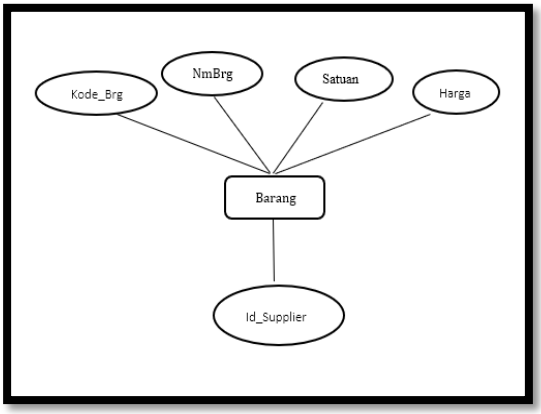


Gambar 3. Atribut Inheritance

5. Specialization

Specialization merupakan proses pendefinisian suatu himpunan subclass dari suatu entitas, yang disebut sebagai superclass dari specialization. Himpunan subclass tersebut membentuk specialization yang dibentuk berdasarkan beberapa sifat atau karakteristik unik dari suatu entitas pada suatu superclass, yang menunjukkan perbedaan yang jelas antara entitas tersebut(Rédei, 2008).

Contoh : Entitas Barang dengan beberapa atribut dengan Kode_Brg sebagai kunci utama/ pengidentifikasi. Id_Suplier merupakan atribut bernilai ganda karena ada lebih dari 1 suplier untu satu Barang.



Gambar 4. Specialization

6. Specialization Hierarchy

Proses pemecahan entitas menjadi subclass-subclass beserta atributnya. Terdapat beberapa jenis spesialisasi seperti Disjoint Total, Disjoint Partial, Overlapping Total, Overlaping parsial.

a. Disjoint Total

Setiap entitas dari suatu superclass harus menjadi anggota dari salah satu atau beberapa subclass dalam suatu relasi

b. Disjoint Partial

Setiap instansiasi entitas dari suatu superclass diijinkan bukan merupakan bagian dari subclass manapun.

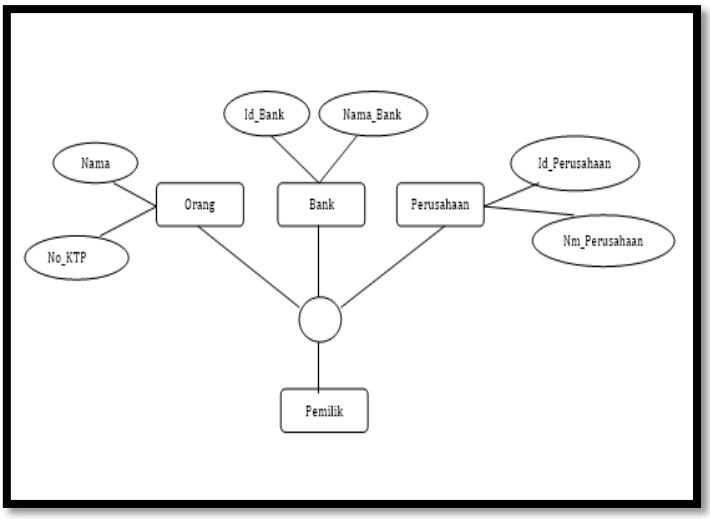
c. Overlapping

Instansiasi dari superclass mungkin merupakan bagian dari 2 atau lebih subclass.

7. Categorization

Kategorisasi adalah proses mendefinisikan subclass yang memiliki lebih dari satu superclass. Ini diperlukan untuk model hubungan superclass/subclass tunggal dengan lebih dari satu superclass, di mana superclass-superclass tersebut menggambarkan jenis entitas yang berbeda. Sebuah kategori memiliki satu atau lebih superclass yang dapat mewakili setiap jenis entitas di mana superclass/subclass lainnya mungkin hanya memiliki satu superclass(Codd, 1983).

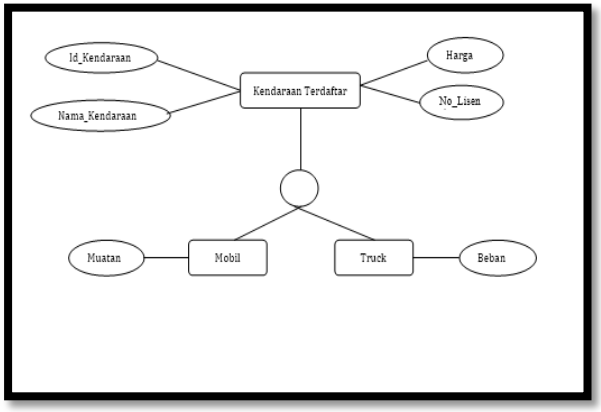
Contoh : Pada Gambar 5 Entitas Pemilik merupakan kategori utama (Superclass) dari orang, bank dan perusahaan. Pemilik dapat menjadi/memiliki orang, bank atau perusahaan atau yang lainnya.



Gambar 5. Relasi antar file 1

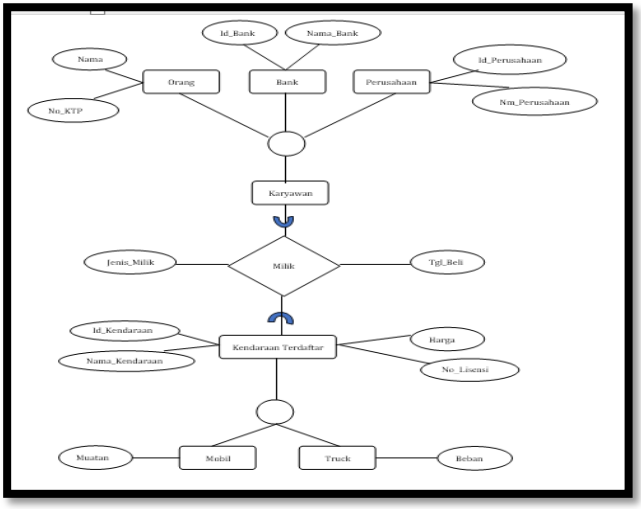
Pada Gambar 6 Kendaraan Terdaftar merupakan kategori utama (Superclass) dari mobil atau Truck.

Kendaraan terdaftar dapat berupa mobil atau truck atau yang lainnya.



Gambar 6. Relasi antar File 2

Kedua kategori tersebut dapat direlasikan dengan tambahan atribut jenis_milik dan tgl_beli seperti pada Gambar 7 berikut :



Gambar 7. Kategorisasi

8
DESAIN BASIS DATA

PROSES desain dimulai berdasarkan hasil analisis data. Ada dua cara untuk mendesain basis data. Yang pertama adalah desain top-down, yang kedua yaitu desain bottom – up.

1. Desain top-down menggunakan model hubungan entitas (ER). Desain ini dimulai dengan mengidentifikasi entitas dan kemudian mendesain hubungan (hubungan) antar entitas dan kardinalitas atau multiplisitas. Setiap entitas memiliki atribut, kunci utama, dan kunci luar, jika ada.
2. Desain dari bawah ke atas (bottom-up) menggunakan proses normalisasi: Desain dimulai dengan mengidentifikasi fitur dan memasukkannya ke dalam kumpulan data untuk membentuk relasi.

Kedua pendekatan bekerja sama. Pada tahap desain basis data logis, pemetaan dari model ER ke tabel relasional dilakukan; jika tabel memiliki data ganda, normalisasi dilakukan terhadap entitas atau hubungan yang memiliki atribut dengan data ganda.

Ini digunakan untuk mengubah model ER awal untuk menghasilkan model ER akhir yang lebih baik.

A. Pemodelan Data Konseptual

Pemodelan data konseptual tidak memerlukan penggunaan perangkat lunak atau perangkat keras, seperti program aplikasi, DBMS, sistem operasi, bahasa pemrograman, dan sebagainya. Namun, pemodelan data konseptual dapat diterapkan pada platform apa pun di masa mendatang. Dengan menggunakan model entity-relationship (ER), pemodelan data konseptual harus menunjukkan fungsi bisnis yang ada dalam organisasi dan menjelaskan kebutuhan data pengguna secara menyeluruh dan akurat.

Desain logis dan desain fisik adalah dua tahap yang terpisah, namun sering kali digabungkan menjadi satu. Mereka tumpang tindih karena sebagian besar DBMS saat ini (termasuk MariaDB) mencocokkan catatan logis dengan catatan fisik pada disk dengan basis 1:1. Default mesin penyimpanan MariaDB, XtraDB, mendukung batasan kunci asing (foreign key), namun beberapa mesin penyimpanan, seperti MyISAM tidak mendukungnya. Klausa ON DELETE CASCADE dan ON DELETE RESTRICT digunakan untuk mendukung kunci asing. ON DELETE RESTRICT berarti bahwa catatan tidak dapat dihapus kecuali semua catatan yang terkait dengan kunci asing juga dihapus.

Menormalkan tabel merupakan langkah penting saat mendesain database. Proses ini membantu menghindari adanya redundansi data dan meningkatkan integritas data.

Perancang database pemula biasanya membuat sejumlah kesalahan umum. Langkah untuk mengantisipasi kesalahan yaitu dengan cara mengidentifikasi entitas dan atribut dengan cermat dan menormalkan data, sehingga kesalahan bisa terhindarkan.

B. 15 Kesalahan Umum

1. Simpan data yang tidak terkait di tabel berbeda. Orang yang terbiasa menggunakan spreadsheet sering melakukan kesalahan ini karena terbiasa melihat semua datanya dalam satu tabel dua dimensi. Basis data relasional jauh lebih kuat.
2. Jangan simpan nilai yang dapat Anda hitung. Katakanlah Anda tertarik pada tiga angka: /A, B dan hasil kali A dan B (A*B). Jangan simpan produknya. Ini membuang-buang ruang dan dapat dengan mudah dihitung jika Anda membutuhkannya. Ini akan membuat database Anda lebih sulit dipelihara. Jika Anda mengubah A, Anda juga harus mengubah semua produknya.
3. Apakah desain Anda memenuhi semua kondisi yang telah Anda analisis? Di tengah kesibukan dalam membuat diagram hubungan entitas, Anda dapat dengan mudah mengabaikan suatu kondisi. Diagram hubungan entitas biasanya lebih baik dalam membuat pemangku kepentingan menemukan aturan yang salah daripada menemukan aturan yang hilang. Logika bisnis sama pentingnya dengan logika database dan lebih cenderung diabaikan. Misalnya, mudah untuk mengetahui bahwa Anda tidak dapat mengadakan penjualan tanpa pelanggan terkait, namun sudahkah Anda menyadari

bahwa pelanggan tersebut tidak dapat menyetujuinya untuk penjualan kurang dari $500 jika pelanggan lain yang disetujui tidak merekomendasikan mereka?

4. Apakah atribut Anda yang akan menjadi nama bidang sudah dipilih dengan baik? Bidang harus diberi nama dengan jelas. Misalnya, jika Anda menggunakan f1 dan f2 sebagai pengganti nama belakang dan nama_depan, waktu yang dihemat dalam pengetikan yang lebih sedikit akan hilang dalam mencari ejaan yang benar pada kolom tersebut, atau dalam kesalahan ketika pengembang mengira f1 adalah nama depan, dan f2 adalah nama depan. nama belakang. Hindari nama yang sama untuk bidang yang berbeda. Gunakan istilah yang lebih deskriptif, seperti kode_penjualan atau kode_pelanggan.
5. Jangan membuat terlalu banyak relasi. Hampir setiap tabel dalam suatu sistem dapat dihubungkan dengan logika tertentu, namun hal ini tidak perlu dilakukan. Misalnya, seorang pemain tenis tergabung dalam klub olahraga. Klub olahraga merupakan milik suatu daerah. Para pemain tenis juga termasuk dalam suatu wilayah, namun hubungan ini dapat diperoleh melalui klub olahraga, sehingga tidak perlu menambahkan kunci asing lainnya (kecuali untuk mencapai manfaat kinerja untuk jenis query tertentu). Normalisasi dapat membantu Anda menghindari masalah seperti ini (dan bahkan saat Anda mencoba mengoptimalkan kecepatan, biasanya lebih baik melakukan normalisasi lalu secara sadar melakukan denormalisasi daripada tidak melakukan normalisasi sama sekali).
6. Sebaliknya, apakah Anda sudah melayani semua relasi? Apakah semua relasi dari diagram hubungan entitas

Anda muncul sebagai bidang umum dalam struktur tabel Anda? Sudahkah Anda meliput semua hubungan? Apakah semua relasi banyak-ke-banyak dipecah menjadi dua relasi satu-ke-banyak, dengan entitas persimpangan?

7. Sudahkah Anda mencantumkan semua kendala? Batasan mencakup jenis kelamin yang hanya boleh m atau f, usia anak sekolah yang tidak boleh lebih dari dua puluh tahun, atau alamat email yang harus memiliki tanda @ dan setidaknya satu titik (.; jangan anggap remeh hal-hal seperti ini).
8. Apakah Anda berencana menyimpan terlalu banyak data? Haruskah pelanggan diminta memberikan warna mata, jenis ikan favorit, dan nama kakek neneknya jika mereka hanya mencoba mendaftar untuk buletin online? Terkadang pemangku kepentingan menginginkan terlalu banyak informasi dari pelanggannya. Jika pengguna berada di luar organisasi, mereka mungkin tidak memiliki suara dalam proses desain, namun mereka harus selalu dianggap sebagai yang terdepan. Pertimbangkan juga kesulitan dan waktu yang dibutuhkan untuk menangkap semua data. Jika operator telepon perlu mencatat semua informasi ini sebelum melakukan penjualan, bayangkan betapa lambatnya mereka akan melakukannya. Pertimbangkan juga dampak data terhadap kecepatan database. Tabel yang lebih besar umumnya lebih lambat untuk diakses.
9. Sudahkah Anda menggabungkan bidang yang harus terpisah? Menggabungkan nama depan dan nama belakang menjadi satu bidang adalah kesalahan umum pemula. Nanti Anda akan menyadari bahwa mengurutkan nama berdasarkan abjad itu rumit jika

Anda menyimpannya sebagai John Ellis dan Alfred Ntombela. Jaga kerahasiaan data yang berbeda.

10. Apakah setiap tabel mempunyai kunci utama? Sebaiknya harus ada alasan yang tepat untuk mengabaikan kunci utama. Nama depan dan nama belakang mungkin unik di database Anda saat ini, namun mungkin tidak selalu unik. Membuat kunci utama yang ditentukan sistem akan memastikan kunci tersebut selalu unik.
11. Pikirkan indeks Anda yang lain. Bidang apa yang mungkin digunakan dalam kondisi ini untuk mengakses tabel?
12. Apakah kunci asing Anda ditempatkan dengan benar? Dalam hubungan satu-ke-banyak, kunci asing muncul di tabel banyak, dan kunci utama terkait di tabel satu. Mencampurnya dapat menyebabkan kesalahan.
13. Apakah Anda memastikan integritas referensial? Kunci asing tidak boleh berhubungan dengan kunci utama di tabel lain yang sudah tidak ada lagi.
14. Sudahkah Anda mencakup semua rangkaian karakter yang mungkin Anda perlukan? Huruf Jerman, misalnya, memiliki rangkaian karakter yang diperluas, dan jika database ingin melayani pengguna Jerman, maka hal ini harus diperhitungkan. Demikian pula, tanggal dan format mata uang harus dipertimbangkan secara hati-hati jika sistemnya ingin bersifat internasional.
15. Apakah keamanan Anda cukup? Ingatlah untuk menetapkan izin minimum yang Anda bisa. Jangan izinkan siapa pun melihat tabel jika mereka tidak membutuhkannya. Mengizinkan pengguna melihat data, dapat menimbulkan niat kejahatan dalam membobol data, walaupun mereka tidak dapat mengubahnya, dan

hal ini dimanfaatkan sebagai langkah pertama bagi penyerang.

C. Desain Basis Data Logis

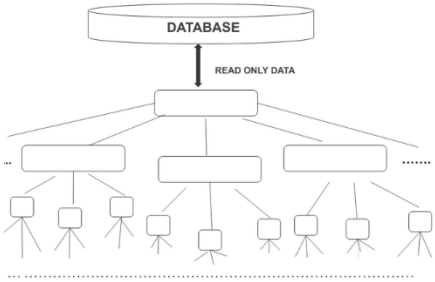
Berdasarkan model data konseptual dan set aturan. Setiap entitas dan hubungan yang memiliki atribut dipetakan menjadi relasi. Relasi dengan kelompok atribut dengan data ganda menyebabkan anomali saat menambah, mengubah, atau menghapus data. Tabel seperti itu harus sekurang-kurangnya dinormalisasi sampai dengan bentuk normal ketiga, Tipe dan domain data setiap relasi ditentukan, termasuk apakah datanya unik atau tidak. Hasilnya adalah spesifikasi untuk setiap relasi.

Model Desain Basis Data Logis menentukan apakah semua persyaratan bisnis telah terpenuhi dengan memberikan deskripsi tingkat rendah tentang entitas, hubungan mereka satu sama lain, dan jenis data apa yang akan disimpan. Administrator dan pengembang basis data biasanya membuat desain basis data fisik, yang membahas bagaimana data akan disimpan dalam DBMS yang sesuai.

Logical Database adalah jenis khusus ABAP *(Advance Business Application and Programming*) yang digunakan untuk mengambil data dari berbagai tabel dan data tersebut saling terkait satu sama lain. Selain itu, database logis menyediakan tampilan Data yang hanya dapat dibaca. Database logika bertujuan untuk membuat tabel yang menyimpan data dengan cara yang tidak berlebihan dan menggunakan kunci asing untuk mendukung hubungan antar entitas dan tabel.

D. Struktur Basis Data Logis

Data disimpan sebagai catatan yang terhubung satu sama lain melalui tautan dan disusun dalam struktur seperti pohon. Logika Database memiliki pernyataan Open SQL yang dapat digunakan untuk membaca data database. Program dibaca oleh basis data logis, disimpan dalam program jika diperlukan, dan kemudian ditransfer baris demi baris ke program aplikasi.



Gambar 1. Struktur Basis Data Logis

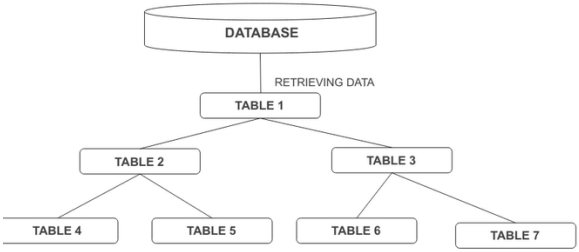
Berikut ini adalah beberapa tugas penting yang dilakukan oleh Basis Data Logis:

1. Database Logis membantu kita membaca data yang sama dari berbagai program.
2. Basis data logis mendefinisikan antarmuka pengguna yang sama untuk masing-masing program.
3. Logika Database memastikan otorisasi untuk memeriksa database secara terpusat.
4. Database Logis meningkatkan kinerja. Kita akan meningkatkan waktu respons dan kinerja Database

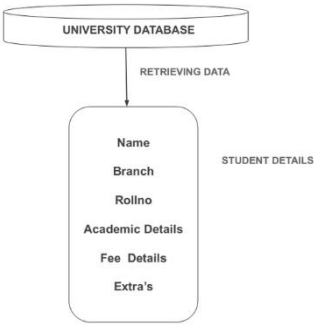
Logical dengan menggunakan gabungan daripada beberapa pernyataan SELECT.

Beberapa fitur basis data logis yaitu dapat memilih jenis data yang kita butuhkan saja, Data divalidasi untuk keamanan, sehingga untuk menjaga integritas data, Basis Data Logis menggunakan struktur hierarki.

Basis data logis menawarkan tampilan tertentu dari tabel basis data logis. Basis data logis digunakan dengan benar ketika struktur basis data besar disusun. Penggunaan syntax atau query yaitu : select, read, process, display. Untuk bekerja dengan database dengan mudah, maka Data *Logical Database* disusun dalam hierarki. Tabel-tabel ini terhubung melalui hubungan Kunci Asing. Data *Logical Database* ditampilkan sebagai diagram berikut:



Contoh: Misalkan HOD *(Head of Departmen)* sebuah institusi pendidikan ingin mengumpulkan data tentang siswa tertentu. Untuk mencapai tujuan ini, dia terlebih dahulu mengumpulkan data batch dan cabangan dari kumpulan data yang sangat besar. Dengan demikian, dia dapat dengan mudah mendapatkan informasi yang diperlukan tentang siswa tanpa mengubahnya.



E. Keuntungan dan Kekurangan Basis Data Logis

Berikut ini merupakan keuntungan dalam mendesain data dengan basis data logis :

1. Kita dapat memilih data yang bermakna dari banyak data dalam database Logis.
2. Basis Data Logis terdiri dari otorisasi pusat yang menentukan apakah seseorang dapat mengakses Basis Data Diautentikasi.
3. Dibandingkan dengan pengkodean database lain, jumlah langkah yang diperlukan untuk mendapatkan data dari database lebih sedikit.
4. Struktur hierarki database meningkatkan akses kinerja membaca data.
5. Antarmuka pengguna yang intuitif
6. Logika Database pertama-tama memeriksa fungsi. Setelah itu, database ini memeriksa apakah input pengguna lengkap, benar, dan masuk akal.

Namun demikian, database logis ini memiliki kekurangan berupa:

1. Ketika data yang dibutuhkan berada di urutan terakhir, database logis membutuhkan lebih banyak waktu karena semua tabel tingkat atas harus dibaca terlebih dahulu, yang membutuhkan lebih banyak waktu dan memperlambat kinerja.
2. Perintah ENDGET tidak ada dalam Database Logical karena blok kode yang terkait dengan peristiwa diakhiri dengan pernyataan peristiwa berikutnya.

F. Desain Basis Data Fisik:

Desain basis data fisik membutuhkan pemahaman tentang DBMS tertentu yang akan digunakan untuk melaksanakan basis data. Dalam desain dan definisi basis data fisik, organisasi daftar, pilihan organisasi file, penggunaan indeks, dan faktor lain ditentukan. Sasarannya adalah desain penyimpanan data yang memastikan integritas, keamanan, dan pemulihan basis data dengan kinerja yang memadai. Sebagai contoh, denormalisasi dapat digunakan untuk meningkatkan kinerja proses dominan frekuensi tinggi, volume tinggi, atau prioritas eksplisit. Oleh karena itu, desain basis data fisik dilakukan seiring dengan desain elemen lain, seperti program, sistem operasi, jaringan komunikasi data, dan perangkat keras.

Desain fisik database dapat meningkatkan kinerja sekaligus memastikan integritas data dengan menghindari redundansi data yang tidak perlu. Pada saat membangun desain database fisik maka anda harus terus memantau kinerja dan integritas data seiring berjalannya waktu. Desain fisik memerlukan perbaikan berkala karena banyak hal. Mengubah model data dari sistem manajemen basis data

(DBMS) tertentu menjadi struktur data fisik adalah proses yang dikenal sebagai desain basis data fisik. Proses ini biasanya terdiri dari transformasi model bisnis menjadi model yang diatribusikan sepenuhnya/ Fully Attribute Model (FAM), yang kemudian diubah menjadi model desain fisik.

Karakteristik umum model data fisik adalah sebagai berikut:

1. Memberikan penjelasan tentang kebutuhan data untuk proyek atau aplikasi tertentu.
2. Lihat semua tabel dan kolom.
3. dilengkapi dengan kunci asing yang digunakan untuk menunjukkan hubungan antar tabel.
4. Model data fisik dan logis dapat berbeda karena pertimbangan model data fisik.

Struktur database yang sebenarnya dapat dicapai melalui proses desain fisik. Selanjutnya, Anda mengubah entitas menjadi tabel, instance menjadi baris, dan atribut menjadi kolom.

9
DATA DEFINITION LANGUANGE

DATA definition language atau yang biasa dikenal dengan singkatan DDL, adalah bagian dari perintah SQL (Structure Query Language) yang berfungsi untuk melakukan perubahan pada struktur data tertentu yang terdapat dalam sebuah basis data.

SQL merupakan bahasa query terstruktur yang berguna untuk mengolah informasi dari sistem basis data. Dengan kata lain SQL merupakan bahasa pemrograman untuk manajemen basis data relasional. Basis data relasional merupakan kumpulan data yang berbentuk tabel, yang terdiri dari baris dan kolom untuk setiap atribut nilai datanya.

DDL dalam basis data memiliki beberapa perintah dasar yang berguna untuk membuat, mengubah dan menghapus sebuah struktur data, tetapi yang diubah bukan data yang terdapat didalamnya. Perintah dasar tersebut adalah create, alter dan drop.

A. Create

Merupakan bahasa pemrograman basis data yang berfungsi untuk membuat objek baru, seperti basis data baru, tabel, indeks dsb.

Contoh penggunaan :

Misal akan membuat sebuah basis data baru dengan nama prodi maka penulisan bahasa DDL yang tepat adalah :

```
CREATE DATABASE prodi;
```

Jika akan membuat sebuah tabel baru dengan nama mata_kuliah maka penulisan bahasa DDL adalah :

```
CREATE TABLE mata_kuliah
(
        kode_mk INT(12),
        nama_mk VARCHAR(30),
        dosen_pengampu VARCHAR(100),
);
```

Bandingkan dengan penulisan bahasa DDL di bawah ini untuk membuat sebuah tabel baru bernama mahasiswa, dilengkapi dengan kode unik atau disebut *primary key* dalam basis data :

```
CREATE TABLE mahasiswa
(
        nim INT(10) NOT NULL PRIMARY KEY,
        nama_mhs VARCHAR(20),
        alamat VARCHAR(50),
        no_hp VARCHAR(15) NULL
);
```

Berdasarkan contoh penggunaan diatas, maka diketahui bahwa kita akan membuat sebuah tabel baru bernama mahasiswa. Dengan ketentuan kolom menunjukkan *field* yang akan digunakan yaitu : nim, nama_mhs, alamat dan no_hp. Sedangkan baris menunjukkan *record* atau isi dari tabel tersebut.

Tipe data yang digunakan untuk masing-masing *field* adalah INT(10) menunjukkan integer merupakan tipe data angka dengan isi maksimal 10 digit angka. VARCHAR(20) menunjukkan tipe data yang menyimpan nilai string dengan isi maksimal 20 karakter. VARCHAR(50) menunjukkan tipe data yang menyimpan nilai string dengan isi maksimal 50 karakter. VARCHAR(15) menunjukkan tipe data string dengan isi maksimal 15 karakter.

Penggunaan NOT NULL yaitu sebagai penanda bahwa *field* nim harus diisi dan tidak boleh kosong. Sedangkan pada *field* no_hp terdapat kata NULL yang berarti *field* tersebut boleh tidak diisi atau boleh kosong. PRIMARY KEY menunjukkan bahwa *field* harus bernilai unik dengan kata lain nim mahasiswa tidak boleh sama antara yang satu dengan yang lain.

B. Alter

Merupakan bahasa pemrograman basis data yang berfungsi untuk melakukan modifikasi struktur tabel seperti menambah, menghapus, mengubah.

Contoh penggunaan :

Misal akan menambahkan *field* tempat_lahir pada tabel mahasiswa maka penulisan bahasa DDL yang tepat adalah :

```
ALTER TABLE mahasiswa
ADD tempat_lahir VARCHAR(20);
```

Di atas terdapat perintah alter untuk melakukan penambahan *field* pada tabel mahasiswa, *field* baru tersebut diberi nama tempat_lahir dengan ketentuan tipe data yang digunakan adalah VARCHAR dengan jumlah maksimal 20 karakter.

Contoh kedua akan menghapus *field* no_hp yang terdapat pada tabel mahasiswa :

```
ALTER TABLE mahasiswa
DROP no_hp VARCHAR(20);
```

Contoh berikutnya yaitu akan mengubah *field* nama_mhs pada tabel mahasiswa dari tipe data VARCHAR(20) menjadi VARCHAR(30) maka penulisan bahasa DDL yang tepat adalah :

```
ALTER TABLE mahasiswa
MODIFY COLUMN nama_mhs VARCHAR(30);
```

Perintah alter dapat juga digunakan bersama dengan perintah rename. Fungsi dari perintah ini adalah untuk mengubah nama. Contohnya yaitu akan mengubah *field* nama_mhs menjadi nama_lengkap pada tabel mahasiswa, maka penulisan bahasa DDL yang tepat adalah :

```
ALTER TABLE mahasiswa
RENAME COLUMN nama_mhs TO nama_lengkap;
```

Sedangkan untuk mengubah nama sebuah tabel dengan nama baru maka penulisan bahasa DDL yang tepat adalah :

```
ALTER TABLE mahasiswa
RENAME TO data_mahasiswa;
```

Arti baris perintah diatas adalah melakukan perubahan nama tabel yang awalnya bernama mahasiswa berganti menjadi data_mahasiswa.

C. Drop

Merupakan bahasa pemrograman basis data yang berfungsi untuk menghapus. Jika perintah ini diberikan untuk menghapus tabel maka secara otomatis segala hal yang berhubungan dengan tabel tersebut akan ikut terhapus, seperti *record* atau isi tabel, indeks tabel dan deskripsi tabel.

Contoh penggunaan :

Misal akan menghapus tabel data_mahasiswa yang telah dibuat maka penulisan bahasa DDL adalah :

```
DROP TABLE data_mahasiswa;
```

Jika perintah drop ini diberikan untuk menghapus database dengan nama prodi yang telah dibuat dalam contoh sebelumnya, maka penulisan bahasa DDL yang tepat adalah :

```
DROP DATABASE prodi;
```

10
DATA ENTRY LANGUANGE

A. Data Entry Language (DEAL)

1. Definisi DEAL

Data Entry Language (DEAL) adalah bahasa khusus yang digunakan untuk berinteraksi dengan sistem basis data. DEAL memberikan cara terstruktur untuk mengambil, memasukkan, mengubah, dan menghapus data dalam basis data (Solichin, 2010). Dalam DEAL, pengguna atau aplikasi dapat menginstruksikan basis data untuk melakukan tugas-tugas seperti pencarian data, menambahkan entri baru, memperbarui informasi, atau menghapus informasi yang tidak diperlukan (-, 2014). DEAL menghadirkan antarmuka yang terfokus pada operasi data dalam basis data.

Pentingnya DEAL dalam Sistem Basis Data (Jr and Schell, 2007)**:**

a. **Kemudahan Penggunaan**: DEAL memungkinkan pengguna, baik yang memiliki latar belakang teknis maupun non-teknis, untuk berinteraksi dengan basis data dengan cara yang lebih mudah dimengerti dan lebih terstruktur. Ini mengurangi kompleksitas akses ke data dalam basis data.
b. **Konsistensi Data**: DEAL memastikan bahwa operasi pada data dilakukan dengan benar dan konsisten. Ini menghindari masalah seperti data yang duplikat atau data yang tidak konsisten.
c. **Pemisahan Aplikasi dan Data**: DEAL memungkinkan aplikasi untuk terpisah dari detail teknis basis data. Ini mempermudah pengembangan dan pemeliharaan aplikasi.
d. **Keamanan**: DEAL memungkinkan pengendalian akses ke data. Hanya pengguna yang memiliki hak akses yang sesuai yang dapat melakukan operasi tertentu pada data.

Contoh Penggunaan Data Entry Language (DEAL) dalam SQL:

```
SELECT nama, gaji FROM pegawai WHERE departemen = 'IT';
```

B. Fitur-Fitur Data Entry Language

1. Manipulasi Data

Salah satu fitur kunci dari Data Entry Language (DEAL) adalah kemampuannya untuk memanipulasi data

dalam sistem basis data. Ini berarti DEAL memungkinkan pengguna atau aplikasi untuk melakukan operasi-operasi seperti mengambil data dari basis data, menambahkan data baru, mengubah data yang sudah ada, dan menghapus data yang tidak diperlukan.

Contoh Manipulasi Data:

```
INSERT INTO customers (customer_id, customer_name, email)
VALUES (1, 'John Doe', 'johndoe@email.com');
```

2. **Query Language**

Fitur penting lainnya dari DEAL adalah kemampuannya untuk menjalankan *Query Language*. Ini memungkinkan pengguna untuk menulis pertanyaan atau kueri yang kompleks untuk mengambil data yang spesifik dari basis data. Misalnya, Anda dapat menulis kueri untuk mencari semua produk dengan harga di atas 100 dolar atau semua pelanggan yang tinggal di kota tertentu. Query Language memungkinkan pencarian data yang tepat sesuai dengan kriteria yang diberikan.

Contoh Query Language:

```
SELECT * FROM customers WHERE customer_name = 'John';
```

3. **Transaksi**

Transaksi adalah serangkaian operasi yang dianggap sebagai satu kesatuan. DEAL memastikan bahwa transaksi dapat dilaksanakan dengan benar dan dalam keadaan yang konsisten. Artinya, jika satu bagian dari transaksi gagal, maka seluruh transaksi dibatalkan (*roll*

back) sehingga data tetap konsisten. Ini sangat penting dalam menghindari masalah seperti pengambilan uang ganda dari rekening bank atau pesanan ganda dari sistem penjualan online.

Contoh Transaksi di SQL:

```
BEGIN TRANSACTION;
-- Lakukan operasi-operasi dalam transaksi
COMMIT; -- Menyelesaikan transaksi
-- Atau jika ada masalah
ROLLBACK; -- Membatalkan transaksi
```

4. **Keamanan**

DEAL menyediakan kontrol akses ke data dalam basis data. Ini berarti hanya pengguna atau aplikasi yang memiliki izin yang sesuai yang dapat melakukan operasi pada data. Keamanan juga termasuk melindungi data dari ancaman seperti *SQL injection*, yang dapat digunakan oleh pihak yang tidak berwenang untuk mengakses atau merusak data dalam basis data.

Contoh Keamanan di SQL:

```
GRANT SELECT ON customers TO user1;
```

C. Jenis-Jenis Data Entry Language (Jamaludin *et al.*, 2022)

1. **SQL (Structured Query Language)**

SQL adalah bahasa yang sangat kuat untuk mengambil, memasukkan, memperbarui, dan menghapus data dalam tabel relasional. SQL digunakan

secara luas di industri dan memiliki sintaks yang terstruktur dengan baik untuk berinteraksi dengan basis data.

Contoh SQL:

Misalnya, dalam SQL, Anda dapat menggunakan perintah SELECT untuk mengambil data dari tabel. Berikut adalah contoh sederhana untuk mengambil semua produk dengan harga di atas 100 dolar:

```
SELECT * FROM products WHERE price > 100;
```

2. **NoSQL Query Languages**

Sistem basis data NoSQL menggunakan DEAL yang dirancang khusus untuk mengakses data yang disimpan dalam format non-relasional seperti dokumen, grafik, atau data berorientasi kolom. DEAL dalam basis data NoSQL dapat bervariasi tergantung pada jenis basis data dan model data yang digunakan.

Contoh NoSQL Query Language:

Sebagai contoh, dalam MongoDB, sebuah basis data dokumen NoSQL, Anda dapat menggunakan perintah find() untuk mengambil dokumen-dokumen yang cocok dengan kriteria tertentu. Berikut adalah contoh untuk mengambil semua dokumen yang memiliki nilai "status" adalah "active": (menggunakan Bahasa javascript)

```
db.collection("orders").find({ status: "active" });
```

3. Bahasa Pemrograman Terintegrasi

Beberapa sistem basis data memungkinkan pengguna untuk menggunakan bahasa pemrograman seperti Java, Python, atau C# sebagai DEAL mereka. Ini memungkinkan integrasi yang lebih baik antara aplikasi dan basis data, sehingga pengembang dapat menggunakan bahasa yang mereka kuasai dalam mengelola data.

D. Pengenalan SQL (Fikry, 2019)

1. Struktur Dasar Perintah SQL

SQL memiliki sintaks yang terstruktur dengan baik. Setiap perintah SQL terdiri dari kata kunci dan klausa yang digunakan untuk menggambarkan operasi yang akan dilakukan. Perintah SQL dapat berupa perintah untuk mengambil data (SELECT), menambahkan data baru (INSERT), memperbarui data yang sudah ada (UPDATE), atau menghapus data (DELETE).

2. Penggunaan SQL dalam Sistem Basis Data Relasional

Sistem basis data seperti MySQL, PostgreSQL, Microsoft SQL Server, dan Oracle Database menggunakan SQL sebagai bahasa untuk berinteraksi dengan basis data. SQL memungkinkan pengguna untuk mengambil, memanipulasi, dan mengelola data dalam tabel-tabel yang terhubung secara relasional.

E. Pencarian dan Manipulasi Data dengan SQL (Jamaludin *et al.*, 2022)

1. SELECT Statement

Pernyataan SELECT digunakan untuk mengambil data dari tabel dalam basis data. Dengan menggunakan SELECT, Anda dapat menentukan kolom mana yang ingin Anda ambil, kriteria pencarian, dan urutan hasilnya. Ini memungkinkan Anda untuk mengambil data yang tepat yang Anda butuhkan dari basis data.

Contoh SELECT Statement:

Misalnya, jika kita memiliki tabel "employees" dan ingin mengambil nama dan gaji dari semua karyawan yang bekerja di departemen "HR," kita dapat menggunakan pernyataan SELECT seperti berikut:

```
SELECT name, salary FROM employees WHERE department = 'HR';
```

2. INSERT Statement

Pernyataan INSERT digunakan untuk menambahkan data baru ke dalam tabel. Anda dapat mengidentifikasi nilai-nilai yang akan dimasukkan ke dalam kolom-kolom tertentu dalam tabel. Ini digunakan untuk membuat catatan baru dalam basis data.

Contoh INSERT Statement:

Misalnya, jika kita ingin menambahkan data baru tentang seorang karyawan ke dalam tabel "employees," kita dapat menggunakan pernyataan INSERT seperti berikut:

INSERT INTO employees (name, department, salary) VALUES ('Jane Doe', 'IT', 60000);

3. UPDATE Statement

Pernyataan UPDATE digunakan untuk memperbarui data yang sudah ada dalam tabel. Anda dapat mengubah nilai-nilai dalam kolom-kolom tertentu berdasarkan kriteria tertentu. Ini memungkinkan Anda untuk memperbarui informasi yang sudah ada.

Contoh UPDATE Statement:

Misalnya, jika kita ingin mengubah gaji karyawan dengan nama "John Smith" menjadi $65,000, kita dapat menggunakan pernyataan UPDATE seperti berikut:

```
UPDATE employees SET salary = 65000 WHERE name = 'John Smith';
```

4. DELETE Statement

Pernyataan DELETE digunakan untuk menghapus data dari tabel berdasarkan kriteria tertentu. Ini digunakan untuk menghapus catatan yang tidak diperlukan atau tidak relevan dari basis data.

Contoh DELETE Statement:

Misalnya, jika kita ingin menghapus semua data karyawan yang memiliki gaji di bawah $40,000, kita dapat menggunakan pernyataan DELETE seperti berikut:

```
DELETE FROM employees WHERE salary < 40000;
```

F. Transaksi dalam SQL (Jamaludin *et al.*, 2022)

1. Pengenalan Transaksi

Transaksi mengacu pada serangkaian operasi yang dianggap sebagai satu kesatuan. Misalnya, jika Anda ingin memindahkan uang antar rekening bank, itu adalah transaksi yang terdiri dari beberapa langkah, seperti pengurangan uang dari satu rekening dan penambahan uang ke rekening lain. SQL memungkinkan Anda untuk menjalankan transaksi dengan benar dan menjaga konsistensi data.

2. Perintah COMMIT dan ROLLBACK

Dalam SQL, Anda dapat menggunakan perintah COMMIT untuk menyimpan perubahan yang Anda buat dalam transaksi ke dalam basis data secara permanen. Namun, jika terjadi kesalahan atau masalah, Anda dapat menggunakan perintah ROLLBACK untuk membatalkan semua perubahan yang dilakukan dalam transaksi.

G. Keamanan dalam SQL (Jamaludin *et al.*, 2022)

1. Pengendalian Akses dengan GRANT dan REVOKE

SQL juga menyediakan alat untuk mengendalikan akses ke data. Dengan perintah GRANT, Anda dapat memberikan izin kepada pengguna atau peran tertentu untuk melakukan operasi tertentu pada data. Di sisi lain, perintah REVOKE digunakan untuk mencabut izin yang telah diberikan.

2. Perlindungan Terhadap SQL Injection

Salah satu ancaman keamanan utama dalam SQL adalah SQL injection, di mana pihak yang tidak berwenang mencoba memanipulasi perintah SQL yang dieksekusi pada basis data. SQL memiliki mekanisme perlindungan yang memungkinkan Anda untuk mencegah serangan SQL injection dengan benar merancang perintah SQL Anda.

H. Data Entry Language pada Sistem Basis Data NoSQL (Jamaludin *et al.*, 2022)

1. NoSQL Query Languages

NoSQL (Not Only SQL) adalah kategori basis data yang berfokus pada penyimpanan dan pengambilan data dalam format non-relasional, seperti dokumen, grafik, atau data berorientasi kolom. NoSQL juga memiliki Data Entry Language (DEAL) yang dirancang khusus untuk mengakses dan mengelola data dalam jenis basis data ini.

2. Pengenalan Basis Data NoSQL

Basis data NoSQL berbeda dari basis data relasional karena mereka tidak mengikuti model tabel tradisional. Mereka memiliki struktur yang lebih fleksibel dan dapat menangani volume data yang sangat besar. Basis data NoSQL digunakan dalam berbagai aplikasi, termasuk media sosial, e-commerce, dan analisis data besar.

Contoh NoSQL Query Languages

Contoh-contoh NoSQL Query Languages meliputi bahasa seperti MongoDB Query Language untuk basis

data dokumen, Cypher untuk basis data grafik, atau Cassandra Query Language (CQL) untuk basis data berorientasi kolom. Masing-masing dari bahasa ini dirancang untuk mengakses dan memanipulasi data dalam jenis basis data NoSQL tertentu.

I. Integrasi DEAL dengan Bahasa Pemrograman (Jamaludin *et al.*, 2022)

1. Menggunakan Java sebagai DEAL

Salah satu aspek penting dari DEAL adalah integrasinya dengan bahasa pemrograman. Anda dapat menggunakan bahasa pemrograman seperti Java sebagai DEAL Anda. Ini memungkinkan Anda untuk membangun aplikasi yang berinteraksi dengan basis data menggunakan bahasa yang sudah Anda kuasai.

2. Menggunakan Python sebagai DEAL

Python juga sering digunakan sebagai DEAL dalam berbagai proyek pengembangan perangkat lunak. Bahasa ini populer karena sintaks yang mudah dimengerti dan banyaknya pustaka dan kerangka kerja yang mendukung integrasi dengan basis data.

11
DATA MANIPULATION LANGUANGE

DATA Manipulation Languange (DML) merupakan bahasa pemrograman komputer yang digunakan untuk melakukan manipulasi dan pengambilan data. Manipulasi data dapat berupa penambahan (menyisipkan), penghapusan, dan perubahan (memperbarui). Salah satu bahasa pemrograman basis data popular dalam model data relational, yaitu Structure Query Languange (SQL), dapat mengelola data di dalam basis data menggunakan perintah DML INSERT, UPDATE, DELETE dan SELECT. Bab ini akan menjelaskan perintah DML dalam bahasa SQL yang disertai dengan contoh.

A. Insert

INSERT adalah suatu instruksi SQL untuk memasukkan record baru ke dalam sebuah tabel. Syarat proses penambahan data adalah telah terciptanya tabel pada sebuah basis data. Sintaks yang digunakan adalah:

1. Notasi Pertama

```
INSERT INTO table_name VALUES
```

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

2. **Notasi Kedua**

```
INSERT INTO table_name (column1, column2,
column3, ...) VALUES (value1, value2, value3,
...);
```

3. **Notasi Ketiga**

```
INSERT INTO table_name VALUES
(value1, value2, value3, ...),
(value1, value2, value3, ...),
(value1, value2, value3, ...);

atau

INSERT INTO table_name (column1, column2,
column3, ...)
VALUES (value1, value2, value3, ...),
(value1, value2, value3, ...),
(value1, value2, value3, ...);
```

Keterangan :

a. Parameter **table_name** sebagai indentitas nama tabel yang akan dimasukan data baru

b. Parameter **column_name** sebagai field atau kolom tabel yang akan diberikan nilai

c. Parameter **value1**, **value2**, dst, sebagai nilai yang akan dimasukan pada kolom tabel

d. Notasi pertama digunakan untuk pemberian nilai semua kolom yang terdaftar pada tabel. Jumlah value sama persis dengan jumlah kolom tabel dan

sesuai dengan domain constrain pada setiap kolom tabel.

e. Notasi kedua digunakan untuk pemberian nilai sesuai dengan kolom tabel yang dicantumkan secara berutan sebagai record baru tabel. Jumlah parameter column_name dan value harus sama dan nilai yang diberikan harus memenuhi domain constraint dari setiap kolom tabel.

f. Notasi ketiga digunakan untuk penambahan data sejumlah data yang dimasukan.

Contoh penggunaan sintaks

```
INSERT INTO customer VALUES
('C0001','Asep','L',
'Bandung',' 081223456789');
```

Perintah sintaks INSERT diatas diperlukan untuk pengisian data seluruh field dan urutan isi field dimasukan berdasarkan urutan nama field pada tabel customer. Misalnya, nilai data pada urutan pertama 'C0001' akan diisikan pada field urutan pertama yaitu ID_Cust, sedangkan nilai data 'Asep' pada urutan ke dua akan diisikan kedalam urutan field ke dua yaitu 'Nama_Cust' dst. Permasalahannya adalah bagaimana yang diisikan datanya adalah pada field tertentu saja, tidak untuk seluruh field. Oleh karena itu, notasi sintaks kedua digunakan untuk pengisian beberapa data saja adalah:

```
INSERT INTO customer (ID_Cust,
Nama_Cust,
Jenis_kelamin) VALUES
```

```
('C0001','Asep','L');
```

Dengan demikian, data yang masuk ke dalam tabel adalah data ID_Cust, Nama_Cust, dan Jenis_kelamin, sedangkan yang lainnya diisi dengan nilai NULL Sementara, pengisian data lebih dari satu pada sebuah tabel dapat menggunakan sintaks notasi ketiga:

```
INSERT INTO customer VALUES
('C0001','Asep','L', 'Bandung','
081223456789'),
('C0002','Wina','P', 'Jakarta','
081223456788'),
('C0003','Rani','P', 'Bogor','
0812234567897);
```

Proses penambahan data pada suatu table yang memiliki primary menjadi prasyarat yang menyebabkan data pada suatu kolom tabel tidak boleh sama. Field yang menjadi primary key tidak boleh bernilai kembar. Misalnya, table customer telah memiliki data dengan ID C0001, kemudian terdapat perintah memasukan data dengan ID customer yang sama. Sistem akan merespon dengan penolakan INSERT data dan menampilkan pesan error seperti yang ditunjukkan di bawah ini,

```
INSERT INTO customer (ID_Cust,
Nama_Cust,
Jenis_kelamin) VALUES ('C0001','Yulia
Ningsih','P');

ERROR 1062 (00000); Duplicate entry ‘C0001’
for key 1
```

B. Update

UPDATE adalah suatu instruksi SQL untuk memperbarui isi record dalam sebuah proses meremajakan data lama menjadi data yang lebih baru di dalam basis data. Namun, tidak semua data perlu diremajakan, melainkan sebagian data yang dianggap perlu untuk diremajakan. Sintaks yang digunakan adalah:

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

Keterangan :

a. Parameter **table_name** sebagai indentitas nama tabel yang akan diubah datanya

b. Parameter **column_name** sebagai field atau kolom tabel yang akan diberikan nilai

c. Parameter **value1**, **value2**, dst, sebagai nilai baru pada kolom tabel yang akan diubah

d. Klausa **where** digunakan sebagai kondisi record tertentu yang memenuhi kondisi yang akan diubah. Klausa **where** bersifat opsional, jika tidak disertakan pada syntax SQL maka semua record pada tabel akan diubah. Namun, jika disertakan, hanya baris-baris tertentu saja yang memenuhi kondisi dapat diubah.

Contoh penggunaan sintaks:

```
UPDATE customer SET
Nama_Cust = 'Asep Hendro'
```

```
WHERE ID_Cust = 'C0001';
```

Pada contoh diatas, hanya data yang memiliki ID 'C0001' saja yang diubah pada field nama_Cust dengan data 'Asep Hendro'. Namun, jika di dalam tabel customer tidak ada yang memiliki ID 'C0001', maka data tidak akan berubah karena hasil pencarian data tidak ditemukan.

C. Delete

DELETE adalah suatu instruksi SQL untuk menghapus record data yang tersimpan dalam basis data, Sintaks yang digunakan adalah:

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

Keterangan :

a. Parameter **table_name** sebagai indentitas nama tabel yang akan dihapus datanya

b. Klausa **where** digunakan sebagai kondisi record tertentu yang memenuhi kondisi yang akan dihapus. Klausa where bersifat opsional sama seperti di dalam perintah UPDATE.

c. Nilai parameter condition pada klausa where dapat lebih dari satu condition. Jika lebih dari satu dihubungkan dengan operator relational seperti AND, OR, XOR.

Contoh penggunaan sintaks:

```
DELETE customer
WHERE ID_Cust = 'C0007';
```

Pada contoh diatas, data dengan ID_Cust C0007 yang terdapat pada tabel customer akan dihapus jika terdapat data customer dengan id C0007. Namun, jika tidak ada data customer di dalam basis data, maka data tidak berubah. Pada proses penghapusan data, sintaks DELETE perlu diperhatikan penggunaan klausa Where. Hal ini karena, jika klausa Where tidak disertakan maka semua data customer di dalam basis data akan terhapus.

D. Select

SELECT adalah instruksi SQL untuk mengambil data dari basis data berdasarkan kebutuhan data. Hasil data dari kembalian operasi select dalam bentuk tabel dan seringkali disebut sebagai result-set. Sintaks yang digunakan adalah:

1. **Notasi Pertama: mengambil semua kolom**

```
SELECT * FROM table_name;
```

2. **Notasi Kedua: mengambil data tertentu sesuai kolom**

```
SELECT column_name1, column_name2, ...
FROM table_name;
```

3. **Notasi Ketiga: mengambil data sesuai kondisi tertentu**

```
SELECT column_name1, column_name2, ...
FROM table_name
WHERE condition;atau
```

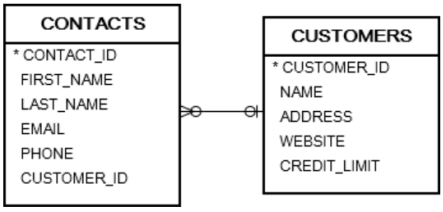
Keterangan :

a. Parameter **table_name** sebagai indentitas nama tabel yang akan dimasukan data baru

b. Parameter **column_name** sebagai data field atau kolom tabel yang akan diambil dari sebuah tabel.

c. Klausa **where** digunakan sebagai kondisi record tertentu yang memenuhi kondisi yang akan diambil datanya. Klausa where bersifat opsional, jika tidak disertakan pada syntax SQL maka semua record pada tabel akan diambil pada saat query.

Agar lebih jelas, mari kita gunakan contoh tabel customers dan contacts di bawah ini, dimana customer bisa dapat memiliki banyak kontak sedangkan contacts hanya boleh dimiliki oleh satu customer. Jenis partisipasi yang terjadi pada hubungan antara customer dengan contacts tidak wajib. Sementara, hubungan antara contacts dengan customer bersifat wajib.



Gambar 1. Contoh relasi Table Custome dan Contacts

Pada saat pengambilan data terdapat berbagai cara sesuai dengan kebutuhan. Di bawah ini dijelaskan penggunaan perintah SELECT pada contoh di atas,

1. Pengambilan data pada satu buah tabel

Gunakan sintaks perintah SELECT dengan mengganti nama table sesuai table yang dituju. Contoh:

a. mengambil semua data customer dengan menggunakan wildcard (*):

```
SELECT * FROM customers;
```

b. mengambil semua data customer dengan menyebutkan nama kolomnya:

```
SELECT customer_id name, address, website,
credit_limit FROM customers;
```

c. mengambil beberapa data customer sesuai kondisi:

```
SELECT custome_id, name FROM customers
WHERE custome_id IN (“C0001”, “C0002”,
“C0003”);
```

2. Pengambilan data dengan melakukan pemilahan atau filtering, yaitu dengan menambahkan kondisi pada klausa where. Contoh penggunaan sintaks:

a. Mengambil data id customer dan nama customr dengan credit limit diantara 10 sampai dengan 100.

```
SELECT customer_id, name FROM customers
WHERE credit_limit BETWEEN 10 and 100;
```

b. Mengambil data id customer dan nama customer dengan data alamat mengandung kata “and”. Penggunaan operator LIKE untuk memilih record data berpola dengan menggunakan keyword %.

```
SELECT custome_id, name FROM customers
WHERE address LIKE '%and%';
```

c. Mengambil data id customer dan nama customer dengan alamat berada di Bandung atau Jakarta.

Penggunaan operator IN untuk memilih record data berdasarkan kemungkinan nilai yang spesifik.

```
SELECT custome_id, name FROM customers
WHERE address IN ('Bandung', 'Jakarta');
```

d. Pengambilan data menggunakan operator logika, OR, AND, dan NOT, pada klausa where. Operator AND akan mengambil record data dengan semua kondisi benar. Operator OR akan mengambil record data dengan salah satu kondisi benar. Operator NOT akan mengambil record data dengan keadaan sebaliknya.

SELECT custome_id, name FROM customers WHERE name LIKE 's%' AND address = 'Bali'; mengambil data customer yang memiliki nama berawalan ‘S’ dan tinggal di bali
SELECT custome_id, name FROM customers WHERE name LIKE 's%' OR address = 'Bali'; mengambil data customer yang memiliki nama berawalan ‘S’ atau tinggal di bali
SELECT custome_id, name FROM customers WHERE NOT (address = 'Bali'); mengambil data customer yang tidak tinggal di bali

1) Pengambilan data dengan mengurutkan record data

Result set dari query dapat diurutkan secara menaik (ASC) atau menurun (DESC) berdasarkan atribut tabel tertentu. Sintaks SQL untuk pengurutan menggunakan

perintah ORDER BY yang ditulis setelah Klausa WHERE atau setelah FROM jika tidak ada klausa WHERE. Secara default, jika pengurutan tidak diikuti keyword ASC atau DESC, maka pengurutan dilakukan dengan ascending. Contoh penggunaan sintaks:

e. Mengambil data id customer dan nama customer dengan data terurut menaik berdasarkan nama.

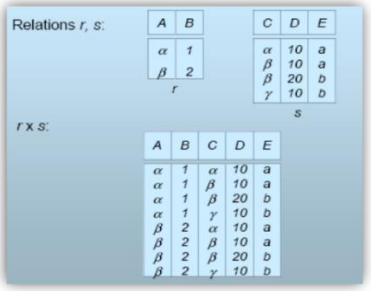
```
SELECT customer_id, name FROM customers
ORDER BY name;
```

f. Mengambil data id customer dan nama customer dengan data terurut menurun berdasarkan nama.

```
SELECT customer_id, name FROM customers
ORDER BY name DESC;
```

3. Pengambilan data yang melibatkan lebih dari satu tabel

Pada saat pengambilan data yang melibatkan lebih dari satu tabel, cross product (|x|) akan dijalankan. Misalnya customers |x| contacts, table result yang dihasilkan sebanyak nRecord Tabel Customer dikali (x) nRecord Tabel Contacts. Pada Gambar 2 terlihat pengabungan dua buah tabel akan menghasilkan 8 record, yaitu Tabel r |x| Tabel s = 2 x 4 = 8 record.



Gambar 2. Cross Product Table Custome dan Contacts

Pada saat pengambilan data (Gambar 2) terdapat terdapat atribut key yang berperan sebagai atribut relasi yang berperan untuk menghubungkan kedua tabel, yaitu field A pada Tabel r dan field C pada Table s. Pada table result, proses penggabungan record tidak saling berhubungan, sehingga dapat menyebabkan data yang diambil tidak benar. Hal ini dapat dilihat pada record 2, 3, 4, 5, dan 8 yang tidak saling berhubungan, yaitu nilai field A ≠ field c. Dengan demikian, operasi Tabel r |x| Tabel s memerlukan filtering untuk mengambil record data yang memiliki relasi pada kedua tabel, yaitu field A (Tabel r) = field c (Tabel s).

Pengambilan data yang melibatkan lebih dari dua tabel dapat menggunakan JOIN dengan menghubungkan kolom terkait diantara relationship table. Penggunaan JOIN dapat dengan menambahkan sintaks JOIN atau melakukan filtering pada klausa WHERE. Contoh penggunaan sintaks:

a. Pengambilan data customer yang memiliki data contacts menggunakan klausa where pada identifier.

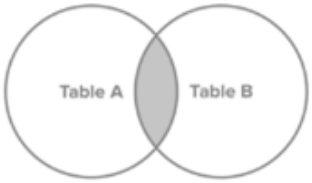
```
SELECT cs.customer_id, cs.name, ct.email
FROM customers cs, contacts ct
WHERE cs. customer_id = ct.customer_id;
```

b. Pengambilan data customer yang memiliki data contacts menggunakan INNER JOIN pada identifier masing-masing table.

```
SELECT cs.customer_id, cs.name, ct.email
FROM customers cs INNER JOIN contacts ct
```

```
ON cs. customer_id = ct.customer_id;
```

c. Konsep JOIN dapat dianalogikan dengan diagram ven Gambar 3. Misalnya, dua set data dalam relational basis data: Tabel A dan Tabel B yang memiliki hubungan berdasarkan identifier table, yaitu primary key dan foreign key, yang telah ditentukan. Kemudian, pengambilan data record data anggota A yang merupakan anggota B. Hal yang dilakukan adalah pencarian data overlapping pada kedua tabel ditentukan oleh berapa banyak record pada Tabel A yang bersesuaian dengan record data pada Tabel B.



Gambar 3. Analogi Konsep JOIN pada Subset Data

d. Sintaks query SELECT dapay menggunakan INNER JOIN, karena data yang dibutuhkan adalah kesamaan data pada kedua table. Namun, jika ingin melihat subset data yg lain, sintaks yang dapat digunakan adalah LEFT JOIN, RIGHT JOIN dan FULL JOIN.

e. LEFT JOIN, pengambilan semua data dari Tabel A (tabel sebelah kiri) termasuk data yang memenuhi syarat join dengan tabel B (tabel sebelah kanan).

```
SELECT <select_list>
FROM TableA A LEFT JOIN TabelB B
        ON A.key_name = B.key_name
```

f. RIGHT JOIN, pengambil semua data dari Tabel B (tabel sebelah kanan) termasuk data yang memenuhi syarat join dengan tabel A (tabel sebelah kiri).

```
SELECT <select_list>
FROM TableA A RIGHT JOIN TabelB B
    ON A.key_name = B.key_name
```

g. FULL JOIN, mengambil semua data dari Tabel A dan Tabel B baik yang memenuhi syarat join maupun yang tidak memenuhi syarat join.

```
SELECT <select_list> FROM TableA A FULL
    OUTER JOIN TabelB B
    ON A.key_name = B.key_name
```

h. Pengambilan data dengan subquery

i. Subquery menyatakan query di dalam SQL query lain yang dapat disematkan pada filed di dalam SELECT, atau di dalam FROM sebagai tabel yang dibentuk dari query atau dapat juga ditambahkan pada Klausa WHERE. Contoh penggunaan sintaks:

```
SELECT cs.custome_id, name,
(SELECT email FROM contats as ct
WHERE cs. customer_id = ct.customer_id;)
FROM customers AS cs
```

j. Pada contoh diatas, query untuk pengambilan data customer berserta emailnya menggunakan subquery berada pada perintah SELECT dengan menghubungkan kolom customer id pada kedua table. Subquery dapat digunakan pula pada klausa WHERE, INSERT, DELETE dan UPATE.

12
DATA MANIPULATION LANGUANGE

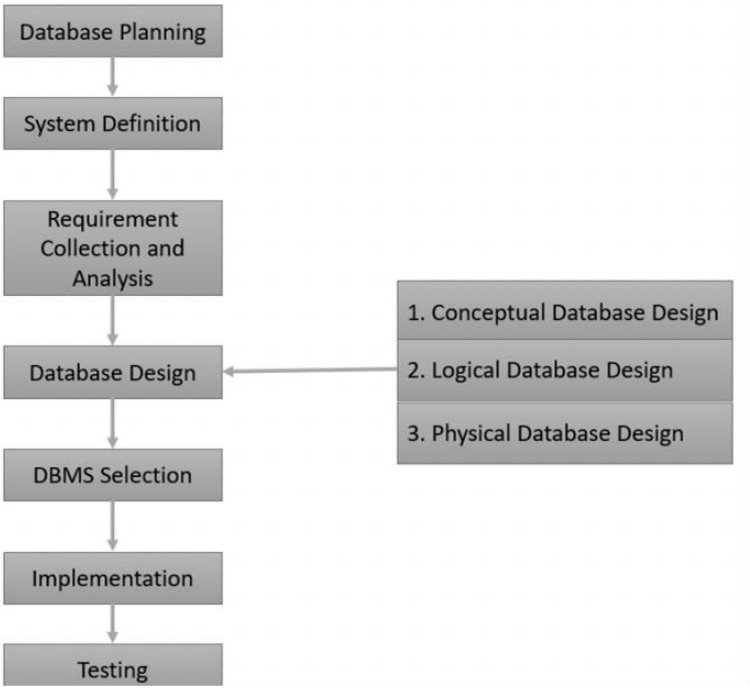
A. Pengertian pengembangan basis data

Pengembangan basis data atau Database System Development Life Cycle (DSDLC) adalah proses merancang, mengembangkan, dan mengelola sistem basis data. Metodologi pengembangan basis data adalah pendekatan atau langkah-langkah yang digunakan untuk memandu proses ini. Perkembangan suatu sistem informasi tidak dapat dipisahkan dari keberadaannya Pengembangan basis data dapat mendukung pengelolaan dan penggunaan data pada sistem yang sedang berjalan. (Pradipta, 2022).

Penting untuk disadari bahwa tahapan siklus pengembangan sistem basis data tidak sepenuhnya berurutan, tetapi melibatkan pengulangan langkah sebelumnya melalui putaran umpan balik. Misalnya masalah ditemui selama proses desain database mungkin memerlukan pengumpulan dan analisis persyaratan tambahan. Untuk sistem database kecil dengan jumlah pengguna yang sedikit,

siklus hidupnya tidak perlu terlalu rumit. Namun, ketika merancang sistem database menengah hingga besar dengan puluhan atau bahkan ribuan pengguna, Menggunakan ratusan query dan program aplikasi, siklus hidup bisa menjadi sangat kompleks (Hardyansyah dan Dewi, 2020).

Menurut Connolly & Begg (2014) Database Development System life Cycle (DSDLC) terdiri dari tahapan yaitu Perencanaan Database, Definisi sistem, Analisis Kebutuhan Pengguna, Perancangan basis data, Penentuan DBMS, Implementasi dan Testing. Ilustrasi DSDLC tertera pada gambar 1 dibawah ini :



Gambar 1 Sistem Pengambangan Basis Data

B. Tahapan Perencanaan Database

Tujuan perencanaan database adalah untuk memastikan bahwa sistem basis data dibangun dan dielola dengan efektif, sesuai dengan kebutuhan organisasi, dan dapat memberikan manfaat yang signifikan. Perencanaan database dapat dilakukan dengan melakukan wawancara untuk menggali kebutuhan yang dibutuhkan oleh terhadap pengguna sistem basis data.

C. Tahapan Pendefinisian Sistem

Definisi sistem adalah langkah yang menjelaskan ruang lingkup dan batasan sistem database serta perspektif pengguna yang berbeda. Tampilan pengguna adalah definisi persyaratan yang dibutuhkan dalam suatu sistem database dari berbagai sudut pandang atau sudut pandang pengguna.

D. Tahapan Analisis Kebutuhan Pengguna

Aktifitas yang dilakukan pada tahap ini adalah melakukan Analisis dan pengumpulam permintaan pengguna.Detail permintaan dikumpulkan dengan berinteraksi dengan sekelompok pengguna atau pengguna individu. Identifikasi masalah dan persyaratan yang ada, ketergantungan aplikasi, prosedur komunikasi dan pelaporan. Tahapan analisis menghasilkan kebutuhan fitur-fitur apa saja yang diperlukan pada saat pengembangan sistem.

E. Tahapan perancangan basis data

Perancangan basis data dibagi menjadi tiga tahap yaitu desain basis data konseptual, desain basis data logis, dan desain basis data. Pendekatan DBLC menjelaskan siklus hidup database yang akan terus kembali ke titik awal karena database memerlukan perbaikan seiring perkembangannya (Kurnia & Budiman, 2020).

1. Perancangan basis data konseptual

Setelah proses Analisa kebutuhan dibuat, maka tahapan selanjutnya adalah membuat Entity Relationship Data (ERD) konseptual dimana terdiri dari beberapa entity, atribut, relasi dan alur (Latukolan, et al, 2019). Komponen ERD (Entity Pelationship Diagram) terdiri dari:

a. Entitas

Entitas dalam ERD merupakan sebuah objek atau simbol yang berfungsi sebagai identitas pada kesatuan yang mempunyai nama dan label. Entitas digambarkan dengan simbol/objek sebuah persegi panjang.

b. Relasi (Hubungan Antar Entitas)

Relasi dalam ERD merupakan sebuah objek atau simbol yang menghubungkan antara satu entitas atau lebih yang tidak memiliki fisik namun ia hanya sebagai konseptual, relasi juga berfungsi untuk mengetahui jenis hubungan antara 2 data. Relasi digambarkan dengan simbol/bentuk belah ketupat.

Derajat relasi atau kardinalitas rasio, yiatu jumlah maksimum relasi antara entitas dengan entitas lainnya. Bentuk Relasi:

1) One to One (1:1), yaitu setiap satu anggota entitas hanya boleh ber-relasi dengan satu anggaota entitas lain dan sebaliknya.
2) One to many (1:M / Many), yaitu setiap satu anggota entitas boleh berrelasi dengan anggota entitas lain lebih dari satu.
3) Many to Many (M:M), yaitu setiap satu anggota entitas ber-relasi dengan banyak himpunan anggota entitas lain.

c. Atribut

Atribut dalam ERD merupakan sebuah objek atau simbol sebagai karakteristik dari entitas atau relasi yang menampilkan penjelasan informasi detail tentang keduanya. Atribut ini juga digambarkan dengan simbol/bentuk elips, dan memiliki beberapa fungsi yaitu:

1) Attribute Key (atribut yang memiliki satu atau gabungan dengan atribut lain, dalam tabel unik).
   • Attribute Simple (atribut yang bernilai atomik, yaitu tidak bisa dipecahpecah lagi).
2) Attribute Multivalue (atribut yang memiliki lebih dari satu nilai / miltivalue).
3) Attribute Composite (atribut dengan bentuk oval yang lebh kecil dari atribut lain sebagai sub-atribut).

4) Attribute Derrivatif (atribut dengan bentuk oval dengan garis putus-putus, yaitu sebuah hasil dari atribut lain atau dari relasi).

d. Alur.

Alur dalam ERD merupakan sebuah objek/simbol yang berfungsi sebagai simbol penghubung antara atribut dan entitas dan hubungan antara entitas dan relasi. Alur disimbolkan dengan bentuk garis.

2. Perancangan basis data logis

Perancangan basis data logis adalah proses membangun model informasi yang digunakan dalam suatu perusahaan berdasarkan model data tertentu tetapi tanpa DBMS tertentu dan pertimbangan fisik lainnya (Indrajani, 2011).

3. Perancangan basis data fisik

Untuk detail desain fisik, diperlukan data sebagai berikut (Yulianti Suryana, 2023):

a. Relasi yang dinormalisasi (tabel)
b. Definisi untuk setiap atribut dari nilai maksimum setiap atribut.
c. mengidentifikasi hak data pengguna.
d. Persyaratan waktu respons dan fungsi terkait data lainnya seperti cadangan, integritas data
e. Deskripsi teknologi yang digunakan untuk mengim-plementasikan basis data (DBMS)

Pedoman pemilihan desain basis data fisik antara lain :

a. Waktu respons: waktu. dari objek data yang ditunjuk oleh acara penggunaan basis data.

b. Space Utility : Jumlah ruang penyimpanan yang digunakan oleh file database dan struktur titik akses.
c. Throughput transaksi: Jumlah rata-rata transaksi yang dapat diproses oleh sistem database per menit.
d. Menjaga integritas data : merupakan keutuhan dan konsistensi data dalam database agar data tersebut dapat diolah sebagai sumber data. used
e. Menghemat ruang penyimpanan dengan kode: Selain menghemat memori, pengkodean juga dapat mengurangi kesalahan atau ketidakkonsistenan.

F. Tahapan Penentuan DBMS

Setelah dilakukan perancangan basis data, tahap berikutnya adalah menentukan DBMS. Tahap penentuan Database Management System (DBMS) adalah salah satu langkah kunci dalam implementasi database. DBMS adalah perangkat lunak yang digunakan untuk mengelola database dan pemilihan yang tepat akan sangat memengaruhi kinerja, skalabilitas, dan keseluruhan keberhasilan pengembangan sistem basis data. Jenis DBMS yang paling sesuai dengan kebutuhan antara lain :

1. SQL (Structured Query Language). DBMS ini Cocok untuk aplikasi yang membutuhkan data terstruktur dan relasional. Contohnya MySQL, PostgreSQL, Microsoft SQL Server.
2. NoSQL DBMS: Cocok untuk aplikasi yang membutuhkan fleksibilitas dalam penyimpanan data yang tidak terstruktur atau semi-struktur. Contohnya MongoDB, Cassandra, Redis.

1. **Implementasi**

Implementasinya adalah implementasi fisik dari database. Penyebaran basis data dapat dilakukan dengan menggunakan DBMS yang menggunakan pernyataan SQL. Pernyataan SQL berupa DDL, DML dan DCL dapat digunakan untuk membuat struktur database (Hardiyansyah & Dewi, 2020).

2. **Testing**

Pengujian basis data ini dimaksudkan untuk memastikan hasil implementasi dapat berjalan sesuai dengan yang diharapkan. Skenario pengujian ini dilakukan dengan membuat suatu algoritma program array data yang tujuannya untuk melakukan proses insert data pada database di server master. Untuk mengetahui hasil replikasi dapat berjalan dilakukan dengan melihat isi database di server slave dan memvalidasi bahwa isi database di server master akan sama dengan isi database di server slave (Yuliansyah, 2014).

13
DATA CONTROL LANGUANGE & ADMINISTRASI BASIS DATA

PADA bagian-bagian sebelumnya kita telah belajar mendefinisikan struktur object basis data seperti misalnya table, view, dll beserta dengan atributnya. Kita juga telah mengerti cara memanipulasi database seperti select, insert, update, dan delete. Sekarang yang belum kita pelajari adalah bagaimana cara mengendalikan hak aksesnya. Untuk memudahkan pembaca, kami pecah menjadi 2 bagian yaitu bagian Data Control Language dan bagian Adminstrasi Basis Data.

A. Data Control Language

Pengendalian user merupakan bagian yang wajib dipahami administrator database. Pengendalian user meliputi pembuatan/penghapusan user, role, dan hak akses spesifiknya. Pada bagian ini kita mengatur keamanan dari ijin akses database pada user atau role.

1. Hak Akses

Pada saat DML kita belajar bagaimana melakukan insert, update, delete, select. Dan pada bagian DDL kita telah belajar melakukan create, drop, dan alter. Ketujuh hal itu yaitu insert, update, delete, select, create, drop, dan alter harus dibatasi keamanan. Hak akses membatasi hak-hak pada DDL dan DML pada satu pengguna atau role.

Hak akses dapat diberikan kepada user maupun role dengan perintah GRANT dan REVOKE. Perintah GRANT digunakan untuk memberikan hak akses. Sedangkan perintah REVOKE digunakan untuk mencabut hak akses.

```
GRANT
        priv_type [(column_list)]
                [, priv_type [(column_list)]] ...
        ON [object_type] priv_level
        TO user_or_role [, user_or_role] ...
        [WITH GRANT OPTION]
        [AS user
                [WITH ROLE
                        DEFAULT
                        | NONE
                        | ALL
                        | ALL EXCEPT role [, role ] ...
                        | role [, role ] ...
                ]
        ]
```

Keterangan :

a. Priv_type : bisa berupa insert, update, delete, select, create, drop, dan alter

b. Object_type : pada object bisa berupa database, table, maupun view.

c. User_or_role : bisa berupa user atau juga role

WITH GRANT OPTION : adalah mengenai user atau role tersebut yang mendapatkan hak akses apaka dapat membetikan hak akses juga ke pengguna atau role lain.

Tabel 1. Level hak akses

Hak Akses	Perijinan	Object
ALL	Semua Akses	Global, DB, Tabel
ALTER	TABEL, fungsi, dan prosedur	Global, DB, Tabel
CREATE	CREATE/DROP TABEL	Global, DB, Tabel, dan fungsi/prosedur
	CREATE/ DROP prosedur dan fungsi	Global, DB
	CREATE/DROP TEMPORARY TABLE	Global, DB
	CREATE/DROP/ RENAME USER	Global
	TRIGGER	Global/ DB/ Tabel
	CREATE VIEW	Global, DB, Tabel,
EXCUTE	Memanggil fungsi/ prosedur	Global/DB
INDEX	CREATE/DROP	Global/ DB/ Tabel

DELETE	DELETE	Global/ DB/ Tabel
INSERT	INSERT	Global/ DB/ Tabel
LOCK TABLES	Pada semua table read write	Global/ DB
SELECT	SEKECT	Global/ DB/ Tabel/ Kolom
UPDATE	UPDATE	Global/ DB/ Tabel/ kolom

Sebagai contoh di bawah ini kita membuat user ucoba.

```
create user 'ucoba'@'localhost' identified by 'ucoba123';
```

Kemudian kita berikan hak akses kepada database dbcoba

```
grant all on dbcoba.* to 'ucoba'@'localhost';
```

Pada contoh tersebut grant all memberikan hak akses pada semua operasi select, insert, update, delete, create, drop, dan alter. Dbcoba.* berarti memeberikan hak akses pada semua object pada database dbcoba. Hak akses tersebut diberkan kepada user 'ucoba'. Nah tapi kenapa alamat localhost juga ikut-ikutan, akan dijelaskan pada sub bab berikutnya.

2. Role

Target dari role bisa user bisa juga role. Di sub bab ini kita bahas bagian role. Role dapat berfungsi semacam domain hak akses yang sama. Beberapa user dapat

dimasukkan ke dalam role tersebut. Dan kelak kita dapat mengganti role-role tersebut sesuai kebutuhan kita. Dengan adanya role tersebut, maka seluruh user yang masuk kedalam role tersebut akan berubah mengikuti role tersebut.

Untuk membuat role kitab bisa menggunakan sintak sbb:

```
CREATE ROLE [IF NOT EXISTS] role1 [, role2 ] ...
```

Penjelasan pada perintah di atas. [IF NOT EXISTS] artinya jika role dengan nama role1, role2, dst jika tidak ada maka akan dibuatkan role baru dengan nama role1, role2, dan seterusnya.

Sebagai contoh kita akan membuat role ‘admin’, role ‘programmer’, role ‘tester’.

```
CREATE ROLE ‘admin’, 'programmer';
CREATE ROLE 'tester'@'192.168.0.4';
```

Kusus user tester hanya dapat diterapkan jika pengguna dengan akun tester koneksi ke server mysql pada IP 192.168.0.4. selebihnya user admin dan user programmer akan terisi secara otomatis alamat ‘%’. Jadi jika kita tidak menspesifikasi @alamat pengguna maka secara otomatis terisi *any address* %. IP dimaksud disana

adalah alamat database server kita. Sebagai contoh pada tester alamat database berada pada IP 192.168.0.4

Untuk GRANT ROLE kepada pengguna lain atau kepada role lain memiliki sintaks sebagai berikut :

GRANT role1 [, role2] ...

TO user1_or_role1 [, user2_or_role2] ...

[WITH ADMIN OPTION]

Syntak GRANT artinya kita memberikan hak berupa role1, role2, dst kepada user ataupun role.

Sintaks user1_or_role1, user2_or_role2 artinya pengguna pertama atau mungkin bis akita gunakan role pertama, kemudiaan user2_or_role2 artinya kitab isa memilih user2 atau role2. Sintaks ini adalah tujuan kepada siapa role diberikan bisa satu user saja atau multi user bahkan kepada multi role.

Sintaks [WITH ADMIN OPTION] artinya dapat memberikan kewenangan kepada sasaran yang kita kasih hak akses. Apakah dia dapat ngasih ijin ke orang lain juga atau tidak boleh ngasih ijin lagi. Jika opsi [WITH ADMIN OPTION] kita sematkan, artinya sasaran role atau user yang kita masukkan ke role1, role2, dst berhak memberi grant juga pada yang lain.

Sebagai contoh

GRANT programmer, admin TO ‘ucoba’@’localhost’ WITH GRANT OPTION.

Sekarang user ucoba, dapat ngasih grant role juga kepada user atau kepada role yang lain.

Nah bagaimanakah cara untuk membatalkan hak akses perintahnya menggunakan REVOKE. Caranya sama dengan kita menggunakan perintah GRANT. Sambil jalan, nanti kita akan menemui penggunaan REVOKE.

3. Pengelolaan *User*

Setelah kita membaca level hak akses pada table 1. kita tahu bahwa sebenarnya manajemen user tersemat pada tulisan tersebut. Yaitu create, drop, dan rename user. Hanya saja sebenarnya mysql tidak menyediakan rename secara eksplisit.

Langsung saja kita mencoba bagaimanakah cara membuat user baru. Untuk sintak membuat user baru sbb:

```
CREATE USER 'namaakun'@'alamat' IDENTIFIED BY 'passwordsaya';
```

Pada sintaks diatas CREATE USER suatu perintah yang dikenali console mysql sebagai pembuatan user baru. Adapun namaakun adalah nama user yang akan dibuat. Setelah tanda @, kita membaca '**alamat**'. Nah apakah ini maksudnya?.

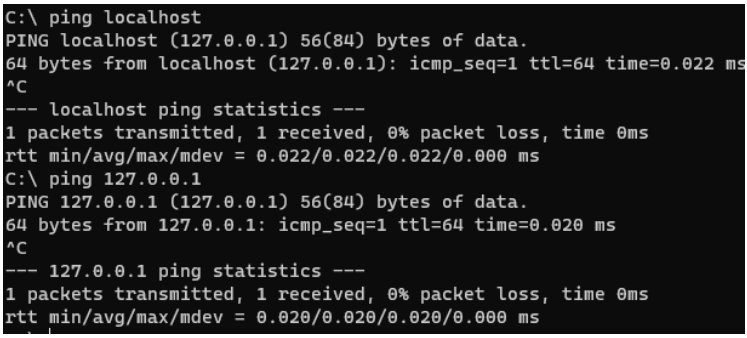
Di dalam jaringan komputer, setiap perangkat pasti memiliki alamat IP apalagi jika itu adalah server database. Sebagai contoh 192.168.0.4 adalah alamat IP server database contoh pada sub bab 14.1.2.

Ada lagi alamat **localhost**, nah alamat apakah ini. localhost itu sebenarnya adalah penamaan dari alamat IP 127.0.0.1 (atau local IP anda). Setiap perangkat laptop/computer pasti memiliki alamat ini. Jika tidak percaya, coba anda buka **command prompt**. Kemudian ketik ping localhost kemudian ketik ping 127.0.0.1, maka hasilnya dipastikan sama.

Gambar 1. ping localhost vs ping 127.0.0.1

Dari gambar diatas kita mengetahui bahwa 127.0.0.1 bernama localhost.

Kemudian penjelasan berikutnya pada create user adalah IDENTIFIED BY passwordsaya, ini artinya adalah



password yang akan anda pakai untuk masuk ke dalam database mysql.

Selanjutnya untuk menghapus databse mysql kita dapat melakukannya dengan sintaks sbb :

DROP USER 'namauser'@'alamat';

Nah lagi-lagi kita ketemu dengan @alamat. Dari sini semoga kita bisa mengambil hikmah, bahwa

pengetahuan jaringan itu perlu meski tidak *advance*. Perintah DROP USER ini maksudnya memberitahu kepada console mysql bahwa kita akan menghapus user tertentu.

Pada sub bab 14.1.1 kita telah membuat user ucoba@localhost dengan password ucoba134. Sekarang mari kita hapus user tersebut menggunakan syntak yang sekarang kita baca.

DROP USER ucoba@localhost;

Nah cukup mudah bukan untuk menhapus sebuah user yang telah kita buat. Bahkan telah kita kasih hak akses all privilege pada semua object di dalam database dbcoba. Coba anda baca lagi pada akhir sub bab 14.1.1. lebih cepat menghapus daripada membuat dan memberinya hak akses.

Nah bagaimanakah cara untuk mengubah nama database. Ini hanya trik saja :

CREATE USER 'ucoba2'@'localhost' IDENTIFIED BY 'password1ku';

GRANT INSERT, UPDATE, DELETE, SELECT ON dbcoba.* TO 'ucoba2'@'localhost';

REVOKE ALL ON dbcoba.* FROM 'ucoba'@'localhost';

DROP USER 'ucoba'@'localhost';

Kita telah berhasil merubah nama **ucoba** menjadi **ucoba2** hanya saja hak akses hanya CRUD (insert, select, update, dan delete) saja.

B. Administrasi Database

Seorang DBA(Data Base Administrator) adalah seseorang yang bertugas untuk mengelola dan memelihara server database. Tugasnya meliputi instalasi, pembuatan user dan hak akses, melakukan backup database sebagai jaga-jaga jika suatu saat databasenya collapse.

1. Basis Data (Database) dan Database Management System (DBMS)

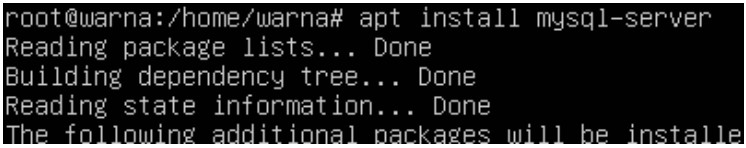
Database berupa struktur data atau object-object seperti tabel yang dibantu oleh view, fungsi, prosedur, dll. Sedangkan DBMS mengacu pada system software untuk mengelola database tersebut. Pada kasus ini kita memilih database mysql yang paling dekat dengan kebutuhan kita sehari-hari. Alasan utamanya adalah free. Tetapi jika Sebagian kita beranggapan bahwa mysql telah dibeli oleh oracle, maka kita masih punya alternatif yaitu mariadb untuk mengerjakan project-project anda. Tapi kali ini saya contohkah penggunaan mysql.

2. Instalasi Server dan Client DBMS

Jika kita menggunakan windows cukup tinggal klik saja. Maka semuanya beres. Di sini saya contohkan jika menggunakan linux :

Gambar 2. update ubuntu

Sebelum kita install kita perlu update repository ubuntu kita supaya mendapatkan informasi release terbaru.



Gambar 3, install mysql server di ubuntu

Perintah apt install mysql-server digunakan untuk mengisntall server mysql. Nanti jika ada pertanyaan-pertanyaan jawab saja “yes”.

3. Koneksi Database

Untuk masuk ke database mysql gunakan perintah

mysql -u root -p

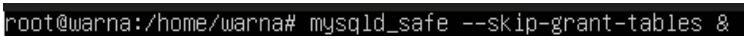
Nah kemudian akan dimintai password. Bagaimana jika lokasi server, ditempat lain. Maka gunakan perintah

mysql -u root -p -h 192.168.0.4.

Opsi -h mengacu kepada host(alamat server tujuan). Adakalanya kita lupa password. Maka matikan service mysql yang sedang berjalan dengan perintah

sudo service mysql stop

kemudian



Setelah itu dipersilahkan untuk masuk tanpa password

```
root@warna:/home/warna# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.6.12-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04
```

Setelah masuk, gunakan perintah alter. Untuk kasus ini tidak perlu mengkawatirkan hak akses, karena grant pada tabel telah diskip dengan opsi **–skip-grant-tables.** Kemudian setelah masuk maka sekarang kita dapat mengubah password.

Posisi user dan password ada di tabel **user**. Tabel user berada pada database mysql. Untuk itu setelah kita login menggunakan user root. Maka selanjutnya gunakan perintah **use mysql** seperti di bawah ini :

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

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

MariaDB [(none)]> use mysql
Database changed
MariaDB [mysql]> select user,password, host from user;
+-------------+-------------------------------------------+-----------------+
| User        | Password                                  | Host            |
+-------------+-------------------------------------------+-----------------+
| mariadb.sys |                                           | localhost       |
| root        | *7B6A19FE64CD08B4273C05CC8BA55CC631237223 | localhost       |
| root        | *7B6A19FE64CD08B4273C05CC8BA55CC631237223 | desktop-lhd7pq2 |
| root        | *7B6A19FE64CD08B4273C05CC8BA55CC631237223 | 127.0.0.1       |
| root        | *7B6A19FE64CD08B4273C05CC8BA55CC631237223 | ::1             |
| root        | *7B6A19FE64CD08B4273C05CC8BA55CC631237223 | %               |
| warna       | *CCBDFAB164C7C6C3D9F87AF2815A8AFC3618CFB1 | localhost       |
| ucoba2      | *7891BC288A346893698D563E69D4FC7BFDF42162 | localhost       |
+-------------+-------------------------------------------+-----------------+
8 rows in set (0.002 sec)

MariaDB [mysql]>
```

Gambar 4. memilih database mysql, dan menampilkan user

Pada gambar terserbut terdapat beberapa user root dengan berbagai pilihan domain host, dan juga user ucoba2 yang telah kita buat. Pada kolom password kita juga dapat mengganti dengan password baru. Untuk itu

ada banyak cara untuk mengganti password ucoba2. Kita persingkat saja menggunakan perintah

```
MariaDB [mysql]> set password=password('passwordbaru')
```

Perintah tersebut digunakan untuk merubah password kita sendiri.

Berikut beberapa perintah untuk merubah password pada user tertentu :

SET PASSWORD FOR 'ucoba2'@'localhost' =PASSWORD('passwordbaru');

Penjelasan diatas ketika terdapat perintah SET PASSWORD FOR diikuti user dan host, maka console mysql mengenalinya sebegai perintah untuk merubah password. Kemudian fungsi PASSWORD(") merupakan enkripsi hash password pada DBMS mysql.

Berikutnya kita juga bisa merubah langsung pada tabel user, dengan catatan kita telah mendapatkan grant pada database mysql pada tabel user. Bagaimanakah caranya, sebagaimana pada akhir sub bab 14.1.1, menggunakan user root kita kasih grant ke user ucoba2 dengan cara :

grant select, update on mysql.user to ucoba2@localhost;

Dengan perintah tersebut maka ucoba2 memiliki hak untuk masuk ke tabel user di database coba dan dapat update saja.

Kembali pada topik kita yaitu merubah password secara langsung yaitu

UPDATE mysql.user SET password = PASSWORD('password123') where user='ucoba2' and host='localhost'

Hanya saja merubah password dengan cara seperti itu kita harus hati-hati karena DBMS mysql telah running. Data di memory untuk DBMS dengan data file tabel mysql perlu di reload supaya system yang running menggunakan variable-variabel baru. Oleh karena itu kita wajib menggunakan flush privilege

FLUSH PRIVILEGES.

Perintah tersebut akan mereload data privileges dari tabel ke database yang running.

14
ANALISIS SISTEM BERJALAN

ANALISIS Sistem Berjalan (juga dikenal sebagai Analisis Sistem yang Berjalan atau Analisis Proses Berjalan) adalah suatu metode yang digunakan untuk memahami, menggambarkan, dan menganalisis proses atau sistem yang sedang berlangsung di dalam suatu organisasi atau entitas bisnis. Tujuan dari analisis sistem berjalan adalah untuk meningkatkan pemahaman tentang bagaimana suatu sistem atau proses bekerja, mengidentifikasi masalah atau kelemahan, dan mengidentifikasi peluang perbaikan. Berikut adalah beberapa langkah penting dalam melakukan analisis sistem berjalan adalah Identifikasi Tujuan: Mulai dengan mengidentifikasi tujuan analisis sistem berjalan. Apa yang ingin Anda capai dengan analisis ini? Apakah Anda mencari cara untuk meningkatkan efisiensi, mengurangi biaya, meningkatkan kualitas, atau mencapai tujuan bisnis lainnya?; Pilih Lingkup: Tentukan batasan dari analisis Anda. Apa yang akan Anda fokuskan? Adakah bagian tertentu dari sistem

atau proses yang akan dianalisis?; Kumpulkan Data: Kumpulkan data tentang sistem atau proses yang akan Anda analisis. Data ini bisa berupa dokumen, laporan, wawancara dengan pekerja, pengamatan, atau data lain yang relevan. Modelkan Proses: Buat model visual atau deskripsi naratif dari proses atau sistem yang sedang dianalisis. Ini bisa berupa aliran kerja, diagram aliran data, atau catatan tentang langkah-langkah yang terlibat. Identifikasi Masalah: Identifikasi masalah atau kelemahan dalam proses atau sistem. Apakah ada bottlenecks, redundansi, atau hambatan yang menghambat kinerja?; Evaluasi Kinerja: Ukur kinerja sistem atau proses yang ada. Gunakan metrik yang relevan untuk menilai sejauh mana sistem memenuhi tujuan bisnis. Identifikasi Peluang Perbaikan: Cari tahu apakah ada peluang perbaikan yang dapat meningkatkan efisiensi atau kualitas sistem atau proses. Ini mungkin melibatkan perubahan dalam alur kerja, penggunaan teknologi baru, pelatihan karyawan, atau perubahan kebijakan. Rancang Solusi: Jika diperlukan, rancang solusi untuk masalah yang diidentifikasi atau untuk menerapkan peluang perbaikan. Buat rencana tindakan yang jelas. Implementasi: Terapkan perubahan yang direncanakan, dan pastikan untuk memonitor dan mengevaluasi dampaknya. Evaluasi Kembali: Setelah implementasi, evaluasi kembali sistem atau proses untuk melihat apakah perbaikan telah mencapai tujuan yang ditetapkan. Analisis Sistem Berjalan adalah alat yang sangat berguna dalam manajemen bisnis untuk memahami dan meningkatkan proses bisnis. Ini dapat membantu organisasi mengidentifikasi masalah, mengoptimalkan efisiensi, dan meningkatkan daya saing mereka.

A. Tujuan

Tujuan dari Analisis Sistem Berjalan adalah untuk memahami, mengevaluasi, dan meningkatkan sistem atau proses yang sedang berlangsung di dalam suatu organisasi. Dalam konteks ini, analisis sistem berjalan memiliki beberapa tujuan utama, yaitu Memahami Proses atau Sistem yang Ada: Salah satu tujuan utama adalah untuk mendapatkan pemahaman mendalam tentang bagaimana suatu sistem atau proses bekerja dalam situasi nyata. Ini mencakup pemahaman tentang alur kerja, interaksi antar komponen, peran individu atau departemen, dan bagaimana data atau informasi mengalir melalui sistem. Identifikasi Masalah dan Kelemahan: Tujuan penting lainnya adalah mengidentifikasi masalah, hambatan, atau kelemahan dalam sistem atau proses yang dapat menghambat kinerja dan produktivitas. Dengan mengidentifikasi masalah ini, organisasi dapat mencari solusi untuk meningkatkan sistem. Meningkatkan Efisiensi: Analisis sistem berjalan bertujuan untuk mengidentifikasi cara-cara untuk meningkatkan efisiensi operasional. Ini bisa melibatkan penghapusan redundansi, mengoptimalkan aliran kerja, atau menghilangkan bottlenecks yang menghambat kinerja. Meningkatkan Kualitas: Analisis sistem juga dapat digunakan untuk meningkatkan kualitas produk atau layanan yang dihasilkan oleh sistem. Ini bisa mencakup identifikasi penyebab masalah atau cacat dan perbaikan proses untuk mengurangi kesalahan. Mengurangi Biaya: Salah satu tujuan umum adalah mengidentifikasi cara untuk mengurangi biaya operasional. Ini dapat mencakup pengurangan pemborosan, efisiensi penggunaan sumber daya, atau perbaikan dalam

manajemen inventaris. Meningkatkan Kepuasan Pelanggan: Dengan memahami sistem atau proses yang ada, organisasi dapat mengidentifikasi cara untuk meningkatkan kepuasan pelanggan dengan menyediakan layanan yang lebih baik, lebih cepat, atau dengan kualitas yang lebih tinggi. Peningkatan Daya Saing: Analisis sistem berjalan dapat membantu organisasi mempertahankan atau meningkatkan daya saing di pasar. Dengan memahami bagaimana pesaing bekerja dan mengidentifikasi praktik terbaik, organisasi dapat beradaptasi dan berkembang. Perencanaan Strategis: Hasil analisis sistem berjalan dapat digunakan untuk mendukung perencanaan strategis jangka panjang dan pemilihan teknologi yang sesuai untuk mendukung tujuan bisnis. Kepatuhan Peraturan: Tujuan lain mungkin adalah memastikan bahwa sistem atau proses mematuhi semua regulasi dan peraturan yang berlaku, seperti hukum privasi data atau peraturan keamanan. Inovasi: Analisis sistem berjalan juga dapat membantu mengidentifikasi peluang inovasi dalam proses bisnis atau penggunaan teknologi yang lebih canggih. Dengan mencapai tujuan-tujuan ini, analisis sistem berjalan membantu organisasi menjadi lebih efisien, efektif, dan adaptif dalam menghadapi perubahan lingkungan bisnis.

B. Manfaat

Analisis Sistem Berjalan memberikan berbagai manfaat bagi organisasi dan entitas bisnis. Dengan memahami sistem atau proses yang sedang berlangsung dan mengidentifikasi cara untuk meningkatkannya, organisasi dapat mencapai berbagai tujuan dan mengatasi masalah yang mungkin

terjadi. Berikut adalah beberapa manfaat utama dari Analisis Sistem Berjalan Pemahaman yang Mendalam: Analisis sistem berjalan membantu organisasi untuk memahami bagaimana sistem atau proses bekerja dalam situasi sebenarnya. Ini memberikan gambaran yang lebih jelas tentang alur kerja, peran individu, dan interaksi antar komponen sistem. Identifikasi Masalah: Melalui analisis sistem berjalan, organisasi dapat mengidentifikasi masalah, hambatan, atau kelemahan dalam sistem atau proses. Ini dapat berupa penyebab performa rendah, ineffisiensi, bottlenecks, atau masalah kualitas. Peningkatan Efisiensi: Salah satu manfaat utama adalah kemampuan untuk meningkatkan efisiensi operasional. Analisis sistem berjalan membantu menghilangkan pemborosan, memperbaiki alur kerja, dan mengoptimalkan penggunaan sumber daya. Peningkatan Kualitas: Organisasi dapat menggunakan analisis sistem berjalan untuk meningkatkan kualitas produk atau layanan yang dihasilkan. Ini mencakup pengurangan cacat atau kesalahan yang dapat merugikan reputasi perusahaan. Pengurangan Biaya: Dengan mengidentifikasi penyebab pemborosan dan kegiatan yang tidak perlu, analisis sistem berjalan dapat membantu mengurangi biaya operasional secara signifikan. Peningkatan Kepuasan Pelanggan: Analisis sistem berjalan membantu organisasi untuk memberikan layanan yang lebih baik dan lebih cepat kepada pelanggan, meningkatkan kepuasan pelanggan, dan mempertahankan loyalitas. Pengambilan Keputusan yang Lebih Baik: Dengan data yang lebih akurat dan pemahaman yang lebih dalam tentang sistem, manajemen dapat membuat keputusan yang lebih baik berdasarkan fakta. Daya Saing yang Lebih Tinggi: Organisasi dapat meningkatkan daya saing mereka dengan

mengidentifikasi praktik terbaik, mengadopsi teknologi terbaru, dan beradaptasi dengan perubahan di pasar. Inovasi: Analisis sistem berjalan dapat membantu mengidentifikasi peluang inovasi dalam proses bisnis atau penggunaan teknologi yang lebih canggih. Kepatuhan Regulasi: Organisasi dapat memastikan bahwa sistem atau proses mereka mematuhi semua peraturan dan regulasi yang berlaku. Pemantauan dan Evaluasi: Analisis sistem berjalan memberikan kerangka kerja untuk pemantauan dan evaluasi terus-menerus terhadap sistem atau proses yang ada. Perencanaan Strategis: Hasil analisis sistem berjalan dapat digunakan untuk mendukung perencanaan strategis jangka panjang dan pengambilan keputusan yang lebih baik terkait dengan investasi teknologi. Dengan menggabungkan berbagai manfaat ini, Analisis Sistem Berjalan membantu organisasi untuk menjadi lebih efisien, kompetitif, dan beradaptasi dengan perubahan yang terus-menerus dalam lingkungan bisnis.

C. Metode

Ada beberapa metode yang dapat digunakan dalam melakukan Analisis Sistem Berjalan. Pilihan metode tergantung pada sifat sistem atau proses yang akan dianalisis, tujuan analisis, dan sumber daya yang tersedia. Berikut beberapa metode yang umum digunakan dalam Analisis Sistem Berjalan yaitu Wawancara: Metode ini melibatkan berbicara langsung dengan individu yang terlibat dalam sistem atau proses yang sedang dianalisis. Wawancara digunakan untuk mendapatkan wawasan dari perspektif mereka tentang bagaimana sistem berjalan, peran mereka

dalam sistem, dan masalah yang mereka hadapi. Observasi: Observasi langsung proses atau sistem adalah metode penting dalam analisis sistem berjalan. Dengan mengamati proses secara langsung, analis dapat mengidentifikasi bagaimana aktivitas sebenarnya berjalan dan memahami alur kerja. Survei: Survei adalah metode yang melibatkan pengumpulan data melalui kuesioner atau formulir yang diberikan kepada orang yang terlibat dalam sistem atau proses. Survei dapat digunakan untuk mengumpulkan pandangan dan masukan dari berbagai pemangku kepentingan. Pengumpulan Dokumen: Dokumen yang terkait dengan sistem atau proses, seperti laporan, kebijakan, panduan, dan catatan, dapat digunakan sebagai sumber data penting dalam analisis sistem berjalan. Diagram Alur Kerja: Membuat diagram alur kerja adalah metode visual yang digunakan untuk menggambarkan bagaimana aliran informasi atau barang melalui sistem atau proses. Ini membantu dalam memahami alur kerja yang ada. Diagram Aliran Data (DFD): Diagram Aliran Data adalah alat visual yang digunakan untuk menggambarkan bagaimana data mengalir melalui sistem. DFD membantu mengidentifikasi entitas data, proses, dan entitas luar yang terlibat dalam sistem. Analisis Perbandingan: Metode ini melibatkan perbandingan antara sistem yang ada dengan praktik terbaik atau model referensi yang sudah ada. Hal ini membantu dalam mengidentifikasi perbedaan dan peluang perbaikan. Teknik Permodelan Proses: Beberapa organisasi menggunakan teknik permodelan proses yang lebih canggih, seperti Business Process Model and Notation (BPMN) atau Unified Modeling Language (UML) untuk menggambarkan sistem atau proses dengan lebih rinci. Pengukuran Kinerja: Metode ini melibatkan pengumpulan data kuantitatif yang

berkaitan dengan kinerja sistem, seperti waktu yang dibutuhkan untuk menyelesaikan tugas, jumlah kesalahan yang terjadi, atau biaya yang dikeluarkan. Analisis SWOT: Analisis SWOT (Strengths, Weaknesses, Opportunities, Threats) digunakan untuk mengevaluasi kekuatan, kelemahan, peluang, dan ancaman yang terkait dengan sistem atau proses yang sedang dianalisis. Pemodelan Simulasi: Pemodelan simulasi memungkinkan analis untuk memahami bagaimana sistem atau proses akan berperilaku dalam berbagai skenario berbeda melalui simulasi komputer. Pilihan metode tergantung pada kompleksitas sistem yang akan dianalisis, tujuan analisis, dan sumber daya yang tersedia. Seringkali, beberapa metode digunakan secara bersamaan untuk memberikan pemahaman yang komprehensif tentang sistem atau proses yang sedang berjalan.

Sistem Basis Data adalah suatu komponen penting dalam teknologi informasi yang digunakan untuk mengelola dan menyimpan data. Analisis Sistem Berjalan dalam konteks sistem basis data mengacu pada pemahaman, pengevaluasian, dan perbaikan sistem basis data yang sedang beroperasi. Berikut adalah beberapa materi yang mungkin dibahas dalam Sistem Basis Data dengan sub-topik Analisis Sistem Berjalan: Pendahuluan Sistem Basis Data: Pengertian dan pentingnya sistem basis data. Konsep entitas dan atribut. Konsep relasi dan tabel. Analisis Sistem Basis Data: Tujuan dan manfaat analisis sistem basis data. Peran analis sistem basis data. Pengumpulan informasi tentang sistem basis data yang ada. Metode Analisis Sistem Berjalan: Penggunaan wawancara dan observasi untuk memahami bagaimana pengguna sistem berinteraksi dengan basis data. Penggunaan

diagram alur kerja dan diagram aliran data (DFD) untuk menggambarkan alur data dalam sistem. Teknik pengumpulan dan analisis data, termasuk pengecekan kualitas data, pengidentifikasian masalah, dan perubahan yang diperlukan. Identifikasi Masalah dalam Sistem Basis Data: Identifikasi kelemahan dalam struktur basis data, seperti redundansi data. Identifikasi masalah kinerja, seperti waktu respon yang lambat. Identifikasi masalah keamanan, seperti akses yang tidak sah ke data. Pemahaman Alur Data dalam Basis Data: Menganalisis bagaimana data mengalir dalam sistem, dari entitas satu ke entitas lain. Penggunaan DFD untuk menggambarkan alur data dalam sistem. Pengukuran Kinerja Basis Data: Mengukur kinerja basis data dengan menggunakan metrik seperti waktu tanggap, waktu pengolahan, dan utilitas. Evaluasi kualitas data dan pengukuran kecepatan akses data. Perbaikan dan Perancangan Ulang: Menentukan perubahan yang diperlukan dalam basis data atau struktur basis data. Menerapkan solusi untuk mengatasi masalah yang diidentifikasi, seperti normalisasi basis data atau pengoptimalan kueri. Menerapkan praktik terbaik dalam perancangan ulang basis data. Penerapan Perubahan: Proses implementasi perubahan basis data, termasuk migrasi data dan pelatihan pengguna. Pengawasan pelaksanaan perubahan untuk memastikan konsistensi dan integritas data. Evaluasi Kembali dan Pemantauan: Melakukan evaluasi kembali untuk memeriksa apakah perubahan yang diterapkan telah mencapai tujuan yang ditetapkan. Pemantauan dan pemeliharaan sistem basis data setelah perubahan. Kepatuhan Regulasi dan Kebijakan: Memastikan bahwa sistem basis data mematuhi regulasi dan kebijakan yang berlaku, seperti hukum privasi data dan

peraturan keamanan. Analisis Sistem Berjalan dalam konteks sistem basis data membantu organisasi untuk menjaga dan meningkatkan kualitas, efisiensi, dan efektivitas pengelolaan data mereka. Hal ini penting dalam lingkungan bisnis modern yang sangat bergantung pada data yang akurat dan mudah diakses.

Proses yang baik dalam Analisis Sistem Berjalan melibatkan serangkaian langkah sistematis yang membantu dalam memahami, mengevaluasi, dan memperbaiki sistem atau proses yang sedang berlangsung. Berikut adalah langkah-langkah umum yang dapat membantu Anda menjalankan proses Analisis Sistem Berjalan dengan baik yaitu Definisikan Tujuan dan Ruang Lingkup: Tentukan tujuan analisis sistem berjalan Anda. Apa yang ingin Anda capai dengan analisis ini?; Tentukan batasan atau ruang lingkup analisis, yaitu bagian sistem atau proses yang akan dianalisis. Identifikasi Pemangku Kepentingan: Identifikasi orang, departemen, atau kelompok yang terlibat dalam sistem atau proses yang sedang dianalisis. Cari tahu siapa yang akan terpengaruh oleh hasil analisis ini. Pengumpulan Data: Kumpulkan data yang diperlukan untuk memahami sistem atau proses. Ini bisa berupa dokumen, laporan, data kuantitatif, atau wawancara dengan pemangku kepentingan.

Pastikan data yang Anda kumpulkan akurat, relevan, dan komprehensif. Modelkan Proses: Buat model visual atau deskripsi naratif dari proses atau sistem yang sedang dianalisis. Ini bisa berupa diagram alur kerja, diagram aliran data (DFD), atau catatan langkah-langkah. Ini membantu Anda memahami alur kerja yang ada. Analisis Data: Evaluasi data yang telah Anda kumpulkan. Identifikasi masalah,

hambatan, dan peluang perbaikan. Gunakan metrik atau indikator kinerja yang relevan untuk menilai kinerja sistem. Perbanyak Interaksi dengan Pemangku Kepentingan: Terus berinteraksi dengan pemangku kepentingan, termasuk pengguna sistem. Dapatkan masukan dan umpan balik dari mereka tentang pengalaman mereka dengan sistem. Pastikan Anda memahami perspektif mereka dan kebutuhan mereka. Identifikasi Solusi dan Peluang Perbaikan: Temukan solusi untuk masalah yang diidentifikasi dan identifikasi peluang perbaikan yang dapat meningkatkan sistem atau proses. Diskusikan solusi ini dengan pemangku kepentingan dan tim yang relevan. Rancang Solusi: Jika perlu, rancang solusi atau perubahan yang akan diterapkan dalam sistem. Buat rencana tindakan yang jelas untuk implementasi. Implementasi dan Uji Coba: Terapkan perubahan yang telah dirancang dan uji coba dalam situasi nyata. Pastikan perubahan berjalan sesuai rencana dan tidak menyebabkan masalah baru. Evaluasi Kembali: Setelah implementasi, evaluasi kembali sistem atau proses untuk memastikan bahwa perbaikan telah mencapai tujuan yang ditetapkan.

Jika ada masalah atau tantangan baru, segera atasi. Dokumentasi: Dokumentasikan semua langkah dalam analisis sistem berjalan, termasuk temuan, rekomendasi, dan solusi yang diterapkan. Dokumentasi ini akan bermanfaat untuk pengawasan, pelaporan, dan pemantauan berkelanjutan. Pemantauan Berkelanjutan: Terus pantau kinerja sistem setelah perubahan diterapkan. Pastikan bahwa perbaikan berkelanjutan dan sistem tetap relevan dengan kebutuhan bisnis. Proses analisis sistem berjalan harus terus menerus dan dapat berulang untuk memastikan bahwa sistem dan proses selalu dioptimalkan sesuai dengan

perubahan kebutuhan bisnis dan teknologi. Dengan menjalankan proses ini dengan baik, organisasi dapat meningkatkan efisiensi, kualitas, dan daya saing mereka.

Inti dari Analisis Sistem Berjalan adalah memahami, mengevaluasi, dan meningkatkan sistem atau proses yang sedang berlangsung dalam suatu organisasi. Pada dasarnya, analisis sistem berjalan bertujuan untuk mencapai pemahaman yang mendalam tentang bagaimana sistem tersebut berfungsi dan memastikan bahwa sistem tersebut beroperasi dengan efisien, efektif, dan sesuai dengan tujuan bisnis. Inti dari analisis sistem berjalan mencakup hal-hal berikut: Pemahaman yang Mendalam: Inti dari analisis sistem berjalan adalah pemahaman yang mendalam tentang sistem atau proses yang ada. Ini mencakup pemahaman tentang alur kerja, peran individu atau departemen, interaksi antar komponen sistem, dan bagaimana data atau informasi mengalir melalui sistem. Identifikasi Masalah: Analisis sistem berjalan memungkinkan identifikasi masalah, hambatan, atau kelemahan dalam sistem yang mungkin menghambat kinerja. Ini mencakup masalah seperti redundansi data, bottlenecks, performa yang buruk, atau cacat dalam proses. Perbaikan Efisiensi: Analisis sistem berjalan bertujuan untuk meningkatkan efisiensi operasional. Ini dapat mencakup penghapusan pemborosan, pengoptimalan alur kerja, dan pengurangan waktu yang dibutuhkan untuk menyelesaikan tugas. Peningkatan Kualitas: Analisis sistem juga berfokus pada peningkatan kualitas produk atau layanan yang dihasilkan oleh sistem. Ini mencakup identifikasi masalah atau cacat yang dapat merugikan reputasi perusahaan.

Pengurangan Biaya: Salah satu inti dari analisis sistem berjalan adalah mengidentifikasi cara untuk mengurangi biaya operasional. Ini dapat mencakup pengurangan pemborosan, efisiensi penggunaan sumber daya, atau perbaikan dalam manajemen inventaris. Peningkatan Kepuasan Pelanggan: Dengan memahami sistem atau proses yang ada, organisasi dapat mengidentifikasi cara untuk meningkatkan kepuasan pelanggan dengan menyediakan layanan yang lebih baik, lebih cepat, atau dengan kualitas yang lebih tinggi. Daya Saing yang Lebih Tinggi: Analisis sistem berjalan membantu organisasi mempertahankan atau meningkatkan daya saing di pasar dengan mengidentifikasi praktik terbaik, mengadopsi teknologi terbaru, dan beradaptasi dengan perubahan di pasar. Inovasi: Inti dari analisis sistem berjalan juga melibatkan identifikasi peluang inovasi dalam proses bisnis atau penggunaan teknologi yang lebih canggih. Dengan inti ini, analisis sistem berjalan membantu organisasi untuk mempertahankan dan meningkatkan efisiensi, efektivitas, dan daya saing mereka dalam lingkungan bisnis yang terus berubah. Ini juga memastikan bahwa sistem dan proses yang ada selalu mendukung tujuan bisnis yang lebih luas.

Pendapat para ahli tentang Analisis Sistem Berjalan dapat bervariasi tergantung pada konteks dan bidang spesifik yang mereka geluti. Namun, umumnya, para ahli dalam bidang manajemen, teknologi informasi, dan analisis bisnis memberikan pandangan positif tentang pentingnya analisis sistem berjalan. Berikut beberapa pendapat umum yang bisa ditemukan di antara para ahli: W. Edwards Deming: Deming, seorang ahli manajemen kualitas, mengemukakan pentingnya analisis sistem berjalan dalam perbaikan

berkelanjutan. Ia terkenal dengan siklus PDCA (Plan-Do-Check-Act) yang merupakan dasar dari analisis sistem berjalan. Peter Drucker: Peter Drucker, seorang ahli manajemen terkemuka, menekankan pentingnya analisis sistem berjalan dalam manajemen modern. Ia mengatakan, "If you can't measure it, you can't improve it," yang menggambarkan pentingnya pengukuran dan analisis dalam perbaikan proses.

Joseph Juran: Juran adalah seorang ahli manajemen kualitas yang berkontribusi besar dalam analisis sistem berjalan. Ia mengembangkan konsep Manajemen Kualitas Total (Total Quality Management) yang melibatkan analisis sistem berjalan untuk mencapai kualitas yang tinggi. Michael Hammer dan James Champy: Dalam bukunya yang terkenal, "Reengineering the Corporation," Hammer dan Champy menekankan pentingnya analisis sistem berjalan dalam merancang kembali proses bisnis. Mereka mengklaim bahwa analisis sistem berjalan adalah kunci untuk mencapai perubahan transformasional. Ivar Jacobson: Ivar Jacobson, salah satu pencipta Unified Modeling Language (UML), mendukung penggunaan model dalam analisis sistem berjalan. Model-model seperti Use Case dan Activity Diagrams digunakan dalam analisis sistem berjalan untuk memahami dan merancang proses bisnis. Tom Davenport: Tom Davenport, seorang pakar di bidang manajemen proses bisnis, telah menyoroti pentingnya analisis sistem berjalan sebagai komponen kunci dalam manajemen proses bisnis. Ia mendorong organisasi untuk terus memantau dan memperbaiki proses mereka. Pandangan para ahli ini menunjukkan bahwa analisis sistem berjalan adalah elemen penting dalam manajemen bisnis modern dan perbaikan

proses. Dalam lingkungan bisnis yang cepat berubah, pemahaman dan perbaikan terus-menerus terhadap sistem dan proses adalah kunci untuk mencapai efisiensi, efektivitas, dan daya saing yang lebih tinggi.

Dalam kesimpulan, Analisis Sistem Berjalan adalah proses penting yang bertujuan untuk memahami, mengevaluasi, dan meningkatkan sistem atau proses yang sedang berlangsung dalam suatu organisasi. Tujuan utamanya adalah untuk mencapai pemahaman yang mendalam tentang sistem tersebut, mengidentifikasi masalah, dan mengejar peluang perbaikan. Berikut beberapa poin penting yang dapat diambil sebagai kesimpulan: Pemahaman Mendalam: Analisis Sistem Berjalan memungkinkan organisasi untuk memahami dengan baik bagaimana sistem atau proses beroperasi. Ini mencakup pemahaman tentang alur kerja, interaksi komponen, peran individu, dan bagaimana data mengalir melalui sistem. Identifikasi Masalah dan Peluang Perbaikan: Melalui analisis sistem berjalan, organisasi dapat mengidentifikasi masalah, hambatan, atau kelemahan dalam sistem yang dapat menghambat kinerja. Ini juga membuka peluang untuk perbaikan yang dapat meningkatkan efisiensi, efektivitas, dan daya saing. Perbaikan Efisiensi dan Kualitas: Proses ini bertujuan untuk meningkatkan efisiensi operasional dengan menghilangkan pemborosan, mengoptimalkan alur kerja, dan mengurangi biaya. Ini juga berfokus pada peningkatan kualitas produk atau layanan yang dihasilkan. Kepuasan Pelanggan: Dengan memahami sistem atau proses yang ada, organisasi dapat mengidentifikasi cara untuk meningkatkan kepuasan pelanggan dengan menyediakan layanan yang lebih baik, lebih cepat, atau dengan kualitas yang lebih tinggi.

Daya Saing yang Lebih Tinggi: Analisis sistem berjalan membantu organisasi untuk mempertahankan atau meningkatkan daya saing di pasar dengan mengidentifikasi praktik terbaik, mengadopsi teknologi terbaru, dan beradaptasi dengan perubahan dalam lingkungan bisnis. Inovasi: Analisis sistem berjalan juga membantu mengidentifikasi peluang inovasi dalam proses bisnis atau penggunaan teknologi yang lebih canggih. Proses Berkelanjutan: Analisis sistem berjalan adalah suatu proses yang berkelanjutan dan harus terus-menerus. Organisasi perlu terus memantau dan mengevaluasi sistem serta beradaptasi dengan perubahan lingkungan. Kepatuhan dan Regulasi: Analisis sistem berjalan juga memastikan bahwa sistem mematuhi regulasi dan peraturan yang berlaku, seperti peraturan privasi data atau keamanan. Dengan menerapkan Analisis Sistem Berjalan secara efektif, organisasi dapat mencapai perbaikan berkelanjutan, meningkatkan efisiensi operasional, memenuhi kebutuhan pelanggan, dan tetap kompetitif dalam dunia bisnis yang terus berubah.

Untuk melakukan Analisis Sistem Berjalan secara efisien dan efektif, Anda dapat mengikuti serangkaian langkah dan praktik terbaik berikut: Tentukan Tujuan yang Jelas: Sebelum Anda memulai analisis, pastikan Anda memiliki tujuan yang jelas. Apa yang ingin Anda capai dengan analisis sistem berjalan? Tujuan yang jelas membantu memandu seluruh proses. Definisikan Ruang Lingkup Analisis: Batasi ruang lingkup analisis. Jelaskan dengan jelas bagian sistem atau proses yang akan dianalisis. Ini membantu mencegah analisis yang terlalu luas dan tidak terfokus.

Identifikasi Pemangku Kepentingan: Tentukan siapa yang terlibat dalam sistem atau proses dan identifikasi pemangku kepentingan. Dapatkan masukan dan perspektif dari mereka. Ini penting untuk memahami kebutuhan pengguna dan pemangku kepentingan. Gunakan Metode dan Alat yang Tepat: Pilih metode dan alat yang sesuai untuk analisis sistem berjalan Anda. Ini dapat mencakup wawancara, observasi, diagram alur kerja, diagram aliran data (DFD), atau permodelan proses. Pastikan alat-alat ini mendukung tujuan dan ruang lingkup analisis Anda. Pengumpulan Data yang Teliti: Kumpulkan data yang teliti dan relevan. Pastikan data yang Anda kumpulkan akurat dan komprehensif. Hal ini mungkin melibatkan wawancara dengan pengguna sistem, pengumpulan dokumen, dan pengukuran kinerja. Modelkan Proses dengan Jelas: Gunakan alat seperti diagram alur kerja atau DFD untuk menggambarkan alur kerja dan aliran data dalam sistem. Model ini harus jelas dan dapat dimengerti oleh semua pemangku kepentingan.

Analisis Data dengan Teliti: Analisis data dengan cermat untuk mengidentifikasi masalah, hambatan, dan peluang perbaikan. Gunakan metrik kinerja yang relevan untuk menilai kinerja sistem. Keterlibatan Pemangku Kepentingan: Melibatkan pemangku kepentingan selama seluruh proses analisis. Dapatkan masukan dan perspektif mereka, dan ajak mereka untuk berpartisipasi dalam merancang solusi. Perbaikan dan Perancangan Ulang: Berdasarkan temuan analisis, rancang solusi atau perbaikan yang diperlukan. Pastikan solusi ini sesuai dengan tujuan dan kebutuhan bisnis. Implementasi dan Pemantauan: Terapkan perubahan yang direncanakan dan pantau pelaksanaannya. Pastikan

bahwa perubahan berjalan sesuai rencana dan tidak menimbulkan masalah baru. Evaluasi Kembali dan Pelaporan: Setelah implementasi, evaluasi kembali sistem untuk memastikan bahwa perbaikan telah mencapai tujuan yang ditetapkan. Buat laporan hasil analisis dan perbaikan untuk pemangku kepentingan. Pemantauan Berkelanjutan: Terus pantau kinerja sistem setelah perubahan diterapkan. Pastikan bahwa perbaikan berkelanjutan dan sistem tetap relevan dengan kebutuhan bisnis. Kepatuhan Regulasi dan Kebijakan: Pastikan bahwa sistem atau proses mematuhi semua regulasi dan kebijakan yang berlaku. Tim yang Kompeten: Pastikan bahwa Anda memiliki tim yang kompeten dan berpengalaman dalam analisis sistem berjalan. Mereka harus memiliki pemahaman yang mendalam tentang sistem yang sedang dianalisis. Komunikasi yang Efektif: Pastikan komunikasi yang efektif dengan semua pemangku kepentingan. Informasikan mereka tentang perkembangan analisis dan perubahan yang akan diterapkan. Dengan mengikuti praktik-praktik terbaik ini, Anda dapat melakukan Analisis Sistem Berjalan secara efisien dan efektif, yang akan membantu organisasi mencapai perbaikan yang signifikan dalam sistem dan proses mereka.

DAFTAR PUSTAKA

Anjelia, V., Rahman, A., Destiarini, D., 2023. Rancang Bangun Sistem Informasi Perpustakaan Berbasis Web Pada SMAN 10 OKU. INTECH 4, 7–12. doi:10.54895/intech.v4i1.1995

Coronel, C. and Morris, S.A. (2022) *Database System Design, Implementation & Management*. 14th Edition. Boston: Cengage.

Connolly, Thomas & Begg, Carolyn 2014. Database Systems: A Practical Approcah to Design. Implementation, and Management (Sixth Edition). USA : Pearson education.

Connolly, T. Begg, C (2014). Database Systems: A Practical Approach to Design, Implementation, and Management (4th Ed). England: Pearson Inc.

Coronel, C., & Morris, S. (2019). Database Systems: Design, Implementation, and Management (ISBN 9781337627900, 13th ed.). Boston, MA, USA: Cengage Learning.

Codd, E. F. (1983) 'A Relational Model of Data for Large Shared Data Banks', *Communications of the ACM*, 26(1), pp. 64–69. doi: 10.1145/357980.358007.

Cheng, C., Liu, M., ... Zhou, Y., 2022. Slow feature analysis-aided detection and diagnosis of incipient faults for running

gear systems of high-speed trains. ISA Transactions 125, 415–425. doi:10.1016/j.isatra.2021.06.023

Cristian, L., Meutia, D., 2012. Sistem Informasi Akuntansi Pembelian Bahan Baku untuk Proyek. ComTech: Computer, Mathematics and Engineering Applications 3, 118. doi:10.21512/comtech.v3i1.2390

Chandra, A., 2010. Perancangan Data Warehouse Pada Software Laboratory Center. ComTech: Computer, Mathematics and Engineering Applications 1, 585. doi:10.21512/comtech.v1i2.2558

Christian, L., Urfan, M., 2012. Perancangan Sistem Informasi Penjualan pada Perusahaan Jasa Service Provider. ComTech: Computer, Mathematics and Engineering Applications 3, 128. doi:10.21512/comtech.v3i1.2391

Datson, N., Drust, B., ... Gregson, W., 2017. Match Physical Performance of Elite Female Soccer Players during International Competition. Journal of Strength and Conditioning Research 31, 2379–2387. doi:10.1519/JSC.0000000000001575

Duncan, R., Schreuders, Z.C., 2019. Security implications of running windows software on a Linux system using Wine: a malware analysis study. Journal of Computer Virology and Hacking Techniques 15, 39–60. doi:10.1007/s11416-018-0319-9

Darmanto, E. (2016) 'Analisa Perbandingan Pemodelan Basis Data Menggunakan Er- Diagram Dan Eer-Diagram Pada Kasus Sistem Asistensi Perkuliahan Praktikum', *Simetris : Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 7(1), p. 405. doi: 10.24176/simet.v7i1.532.

Date, C. J. (2004). An Introduction to Database Systems (8th ed.). Pearson.

Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson.

Elmasri, R., B. Navathe, S., 2016. FUNDAMENTALS OF DATABASE SYSTEMS, SEVENTH. ed. PEARSON.

Eizentals, P., Katashev, A., Oks, A., 2019. A smart socks system for running gait analysis, in: IcSPORTS 2019 - Proceedings of the 7th International Conference on Sport Sciences Research and Technology Support. SciTePress, pp. 47–54. doi:10.5220/0008070000470054

Fujimura, S., Kojima, T., ... Hori, R., 2020. Real-Time Acoustic Voice Analysis Using a Handheld Device Running Android Operating System. Journal of Voice 34, 823–829. doi:10.1016/j.jvoice.2019.05.013

Fathansyah. (2012). Basis Data (Edisi Revisi). Penerbit Informatika.

Fikry, M. (2019) *Basis Data*. Unimal Press.

Ferrari, L. and Pirozzi, E. (2020) *Learn PostgreSQL : Build and Manage High Performance Database Solutions Using PostgreSQL 12 and 13*. Birmingham: Packt.

Gillenson, M.L. (2023) *Fundamentals of Database Management Systems*. Third. Hoboken: Wiley.

Hardiansyah, A. D., & Dewi, C. N. P. (2020, November). Perancangan basis data sistem informasi perwira tugas belajar (sipatubel) pada kementerian pertahanan. In *Prosiding Seminar Nasional Mahasiswa Bidang Ilmu Komputer dan Aplikasinya* (Vol. 1, No. 2, pp. 222-233).

Hoffer, J. A., Ramesh, V., & Topi, H. (2019). Modern Database Management (ISBN 9781292263359, 13th Global ed.). Harlow, Essex, England: Pearson Education.

Hidayat, W., Pramono, B., Afdulloh, M., 2019. System Analysis Of Inventory Information On Raw Material Companies. Aptisi Transactions on Management (ATM) 3, 126–130. doi:10.33050/atm.v3i2.903

Hodun, M., Clarke, R., ... Hughes, J.D., 2016. Global Positioning System Analysis of Running Performance in Female Field Sports: A Review of the Literature. Strength and Conditioning Journal. doi:10.1519/SSC.0000000000000200

Nugroho, A., 2011. PERANCANGAN DAN IMPLEMENTASI SISTEM BASIS DATA. Andi Publisher.

Indrajani, I. (2011). Perancangan Sistem Basis Data pada Klinik. *ComTech: Computer, Mathematics and Engineering Applications*, *2*(1), 218-228.

Jatnika, H. (2012) *Pengantar Sistem Basis Data Memahami Konsep Dasar & Tuntunan Praktis Perancangan Database*. Jogjakarta: ANDI Offset.

Jamaludin *et al.* (2022) 'Sistem Basis Data', in. PT Global Eksekutif Teknologi.

Jr, R.M. and Schell, G.P. (2007) *Sistem Informasi Manajemen*. 10th edn. Edited by N. Setyaningsih. Jakarta: Salembea Empat.

Jaén-Carrillo, D., García-Pinillos, F., ... Roche-Seruendo, L.E., 2020. Test–retest reliability of the OptoGait system for the analysis of spatiotemporal running gait parameters and lower body stiffness in healthy adults. Proceedings of the Institution of Mechanical Engineers, Part P: Journal of

Sports Engineering and Technology 234, 154–161. doi:10.1177/1754337119898353

Jaén-Carrillo, D., Ruiz-Alias, S.A., ... García-Pinillos, F., 2022. Test-Retest Reliability of the MotionMetrix Software for the Analysis of Walking and Running Gait Parameters. Sensors 22. doi:10.3390/s22093201

Kim, J.H., Kim, SuR., ... Park, J.Y., 2023. Development of automatic gain-tuning algorithm for heading control using free-running test data. International Journal of Naval Architecture and Ocean Engineering 15. doi:10.1016/j.ijnaoe.2023.100517

Kurnia, A., & Budiman, A. (2020). Rancangan Basis Data Sistem Informasi Pencarian Rumah Kos. *JASIKA (Jurnal Aplikasi Sistem Informasi dan Informatika)*, *1*(01), 18-26.

Latukolan, M. L. A., Arwan, A., & Ananta, M. T. (2019). Pengembangan Sistem Pemetaan Otomatis Entity Relationship Diagram Ke Dalam Database. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *3*(4), 4058-4065.

Liu, J., Saquib, N., ... Tai, C.L., 2022. PoseCoach: A Customizable Analysis and Visualization System for Video-based Running Coaching. IEEE Transactions on Visualization and Computer Graphics. doi:10.1109/TVCG.2022.3230855

Lokapitasari Belluano, P.L., Indrawati, I., ... Lantara, D., 2019. ANALISIS TINGKAT KEPUASAN PENGGUNA SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN PIECES FRAMEWORK. ILKOM Jurnal Ilmiah 11, 118–128. doi:10.33096/ilkom.v11i2.398.118-128

Liu, R., Qian, D., ... Chen, Yi, 2021. Investigation of normal knees kinematics in walking and running at different speeds using a portable motion analysis system. Sports Biomechanics. doi:10.1080/14763141.2020.1864015

Modul, T.P. (2009) *Modul Praktikum Perancangan Basis Data*. Jakarta: Bina Sarana Informatika.

Ota, M., Tateuchi, H., ... Ichihashi, N., 2021. Verification of validity of gait analysis systems during treadmill walking and running using human pose tracking algorithm. Gait and Posture 85, 290–297. doi:10.1016/j.gaitpost.2021.02.006

Putri, R.A. (2022) *Buku Ajar Basis Data Edisi Kedua*. Bandung: Media Sains Indonesia.

Pradipta, R. A., Wintoro, P. B., & Budiyanto, D. (2022). Perancangan Pemodelan Basis Data Sistem Informasi Secara Konseptual Dan Logikal. *Jurnal Informatika dan Teknik Elektro Terapan*, *10*(2).

Rachmadi, T. (2020) *Sistem Basis Data MySQL*. Bandar Lampung: Tiga Ebook.

Rédei, G. P. (2008) *Dbms, Encyclopedia of Genetics, Genomics, Proteomics and Informatics*. doi: 10.1007/978-1-4020-6754-9_4159.

Rebrov, O., Kalchenko, B., ... Leonenko, A., 2023. EVALUATION ANALYSIS OF THE RUNNING SYSTEM SOIL INTERACTION OF WHEELED AGRICULTURAL TRACTORS. Bulletin of the National Technical University «KhPI». Series: Automobile and Tractor Construction 36–43. doi:10.20998/2078-6840.2022.1.05

Simoni, L., Scarton, A., ... Pogliaghi, S., 2021. Quantitative and qualitative running gait analysis through an innovative video-based approach. Sensors 21. doi:10.3390/s21092977

Solichin, A. (2010) *MySQL5: Dari Pemula Hingga Mahir*. Achmad Solichin.Silberschatz, A. Korth, H,F. Sudarshan, S (2011). Database System Concepts (6th ed.). New York: McGraw-Hill.

Tohari, H., 2022. PERANCANGAN BASIS DATA DALAM TEORI DAN PRAKTIK, I. ed. Andi Publisher.

Thomas Connolly and Carolyn Begg, "Database Systems: A Practical Approach to Design, Implementation, and Management 6th Ed", Pearson Inc., 2014.

Widodo, A.W. and Kurnianingtyas, D. (2017) *Sistem Basis Data*. Malang: UB Press.

Yuliansyah, H. (2014). Perancangan replikasi basis data mysql dengan mekanisme pengamanan menggunakan ssl encryption. *jurnal Informatika*, *8*(1), 826-836.

Yulianti Suryana. 2023. Perancangan Data Base Lanjutan. Researchgate.net.

https://mariadb.com/kb/en/

https://www.geeksforgeeks.org/

TENTANG PENULIS

Putri Ariatna Alia lahir pada hari selasa tanggal 5 juli 1994 di Surabaya provinsi Jawa Timur. Penulis telah menyelesaikan pendidikan S2 Teknik Elektro dengan Konsentrasi Teknik Telematika di Institut Teknologi Sepuluh Nopember dan D4 Teknik Elektro konsentrasi Teknik Telekomunikasi di Politeknik Elektronika Negeri Surabaya – Institut Teknologi Sepuluh Nopember.

Johan Suryo Prayogo, S.Kom., M.T., seorang Dosen prodi Sistem Informasi di Universitas Anwar Medika. Mengambil studi Strata 1 Teknik Informatika di Universitas Surabaya (UBAYA) serta melanjutkan program Magister Teknik Informatika di Universitas Atma Jaya Yogyakarta (UAJY). Penulis yang memiliki kegemaran membaca dan mempelajari sesuatu yang baru khususnya bidang *UI/UX* dan *web development* tersebut saat ini memiliki sepasang putra dan putri dari seorang istri bernama Sofie Ayuningtyas. Besar harapan penulis dengan terbitnya buku ini menjadi sarana bagi mahasiswa dan

masyarakat umumnya untuk mengenal dan mempelajari basis data. Penulis dapat dihubungi melalui email :

jodimasjolie@gmail.com

Erik Yohan Kartiko, lahir di kota malang pada tanggal 29 Desember 1995. Menyelesaikan studi Sarjana Pendidikan Teknik Informatika di Universitas Negeri Malang pada tahun 2018. Pada tahun 2022 menyelesaikan studi Magister Ilmu Komputer du Universitas Brawijaya. Bekerja di bidang pendidikan mulai tahun 2018..

Dr. Dwi Prasetyo, Dipl.Inf, S.Kom, M.Si.

Lahir di Bogor, Jawa Barat pada bulan Desember 1966. Penulis merupakan Dosen PNS di Universitas Nusa Cendana.

Pendidikan: Doktor Teknologi Pendidikan Universitas Negeri Jakarta, 2013, Master Ilmu Komputer di Jurusan Ilmu Komputer, Sekolah Pascasarjana Institut Pertanian Bogor (IPB), 2005, Sarjana Ilmu Komputer di Jurusan Ilmu Komputer, FMIPA IPB, 1997, Diploma Informatika Komputer di FAPOLTAN IPB, 1988, Tahun 1990-2004 menjadi Staf pada Jurusan Ilmu Komputer, FMIPA IPB.

Tahun 2005 sampai sekarang menjadi Staf Pengajar pada Jurusan Ilmu Komputer, FST Universitas Nusa Cendana (UNDANA) Kupang.

Sejak 2005-2009 menjabat Kepala Laboratorium Komputer Jurusan Matematika, FST UNDANA

Sejak tahun 1990 menjadi staf Pengajar mata kuliah Algoritma, Pemrograman Komputer, Sistem Informasi Manajemen, Sistem Informasi Geografis, Analisis Perancangan Sistem, *Human Computer Interaction*, Kecerdasan Buatan, Grafika Komputer & PJJ: E-Learning - Mobile Learning.

Membuat buku tentang Sistem Informasi Manajmen, *Human Computer Interaction*, *Mobile Computing* & Sistem Informasi Geografis .

Andi Wijaya, S. Kom., M.Kom, Lahir pada 3 Mei 1987 di bondowoso, provinsi jawa timur. Penulis menyelesaikan pendidikan S1 di Sekolah tinggi teknologi Nurul Jadid dan S2 Teknik Informatika di Universitas Dian Nuswantoro Semarang. Penulis merupakan dosen Prodi Rekayasa Prangkat Lunak Fakutas Teknik Universitas Nurul Jadid. Bidan keilmuan Teknologi Informasi dan Komunikasi, Teknik Sistem Informasi, Pengembangan Perangkat Lunak.

Agung Teguh Setyadi, S.Kom., M.Kom. lahir pada tanggal 26 September 1995 di Malang. Penulis telah meyelesaikan studi sarjana pada jurusan S1 Teknik Informatika ITS (Institut Teknologi Sepuluh Nopember) Surabaya (2013-2018) dan magister pada jurusan S2 Teknik Informatika ITS (Institut Teknologi Sepuluh Nopember) Surabaya (2019-2021) dengan mengambil konsentrasi bidang ilmu data science. Saat ini penulis berkerja sebagai dosen di jurusan S1

Rekayasa Perangkat Lunak Universitas Anwar Medika. Penulis dapat dihubungi melalui email:

agung.teguh.setyadi@gmail.com.

Dwi Remawati, Lahir di Ngawi Jawa Timur, 6 Maret 1973. S1 Teknik Informatika di IST Palapa Malang Jawa Timur. S2 Teknik Informatika di UDINUS Semarang. Jawa Tengah. Saat ini sebagai Dosen PNS Dpk di Program Studi Teknologi Informasi STMIK Sinar Nusantara Surakarta – Jawa Tengah – Indonesia.

Zaid Romegar Mair, S.T., M.Cs. Sejak tahun 2014 berkarir sebagai dosen tetap yang tersertifikasi nasional, sekarang aktif tridharma di Universitas Indo Global Mandiri. Beliau juga sebagai International Coding Educator Brightchamps dari India, sebuah perusahaan Pendidikan teknologi yang berfokus pada perkembangan teknologi holistic untuk anak kelas 1 – 12 dalam pengajaran teknologi yang lebih maju. Aktif dalam publikasi, baik nasional maupun international. Dipercaya menjadi review paper pada International Conference on Industrial Science, Engineering and Technology toward Digital Era (eICISET) 2023. Beberapa buku yang pernah diterbitkan diantaranya adalah Pedoman Praktis Multimedia Dengan Authorware 7.0 ISBN:978-602-401-886-3 Tahun 2017. Teori Dan Praktek Sistem Operasi ISBN : 978-602-475-682-6 Tahun 2018. Pada 5 April 2023 mendapatkan gelar penghormatan Doktor Honoris Causa di bidang IT and Computer Sciences dari Universal

Institute of Professional Management Kuala Lumpur Malaysia. Beberapa kali juga meraih peserta terbaik pada beberapa pelatihan dan workshop nasional. Lulusan S1 Teknik Informatika (2008) dari Universitas Ahmad Dahlan Yogyakarta. S2 Computer Science (2013) dari Universitas Gadjah Mada Yogyakarta. Pernah menjadi Ketua Program Studi Teknik Informatika Politeknik Sekayu Tahun 2016-2021. Direktur Utama PT. Angkasa Perwira Nusantaramair Tahun 2021 – Sekarang. Dewan Sekolah, Sekolah Alam Indonesia Palembang 2021 – Sekarang.

Rusina Widha Febriana, S.Kom., M.Kom. adalah seorang Dosen Program Studi S1 Sistem Informasi di Universitas Anwar Medika, salah satu perguruan tinggi swasta bergengsi di Sidoarjo Jawa Timur. Lahir di Surabaya pada tanggal 17 Februari 1988. Penulis telah berkecimpung di dunia IT semenjak menamatkan pendidikan di SMK Telkom Malang. Penulis berhasil menyelesaikan pendidikan program sarjana (S1) di Universitas 17 Agustus 1945 Surabaya pada program studi Teknik Informatika dan menyelesaikan pendidikan program pasca sarjana (S2) di Institut Sains Dan Teknologi Terpadu Surabaya pada program studi Teknologi Informasi. Penulis aktif berkecimpung di dunia pendidikan, penelitian dan penulisan di bidang Data Mining dan Prediction System. Penulis dapat dihubungi menggunakan surel di widha.ennvil@gmail.com

Radinal Setyadinsa, S.Pd., M.T.I merupakan seorang PNS dosen Sarjana Informatika di Universitas Pembangunan Nasional "Veteran" Jakarta (UPNVJ). Penulis dilahirkan di Jakarta. Penulis menempuh pendidikan S1 Pendidikan Teknik Elektronika di Universitas Negeri Jakarta (UNJ) (lulus tahun 2012) dan S2 Magister Teknologi Informasi di Universitas Indonesia (UI) (lulus tahun 2018).

Asri Maspupah, S.S.T., M.T., lahir di Kota Bandung, Provinsi Jawab Barat. Penulis menyelesaikan Pendidikan D4 Teknik Informatika di Politeknik Bandung dan S2 Informatika di Institut Teknologi Bandung. Penulis merupakan Dosen Pegawai Negri Sipil di Prodi D3 Teknik Informatika Jurusan Teknik Komputer dan Informatika. Pengajaran dan penelitian penulis berkonsentrasi pada bidang Rekayasa Prangkat Lunak, khususnya Pengujian Perangkat Lunak. Pada edisi perdana ini, Penulis menyajikan materi Sistem Basis Data bersama para penulis lainnya dari berbagai Institusi. Semoga tulisan ini dapat menambah wawasan pembaca dan menjadi manfaat bagi kita semua. Terima kasih.

Yuyun Khairunisa, M.Kom merupakan Dosen Program Studi Teknologi Permianan di Politeknik Negeri Media Kreatif.

Warna Agung Cahyono Penulis merupakan dosen Rekayasa Perangkat Lunak di Universitas Anwar Medika. Selama masih menjadi mahasiswa S1 ITS hingga menjadi dosen juga praktisi develop project instansi pemerintah maupun swasta.

Prof. Dr. H. Jufriadif Na'am, S.Kom., M.Kom., CIRR. Penulis Lahir di Padang, 03 Februari 1967, telah S3 Teknologi Informasi pada Tahun 2017. Saat ini penulis berprofesi sebagai dosen tetap dan Guru Besar di Universitas Nusa Mandiri. Saat ini menjabat sebagai Kepala Kelompok Bidang Keahlian Computer Vision & Image Processing Universitas Nusa Mandiri. Jika ingin menghubungi bisa melalui email jufradifnaam@gmail.com atau bisa kontak whatsApp +6287895670026. Untuk Mengetahui lebih lanjut bisa ke https://orcid.org/0000-0002-9586-7031 .

SISTEM
BASIS DATA

Buku Sistem Basis ini ditujukan untuk memberikan pemahaman tentang dunia database. Buku ini akan membahas tentang pengenalan basis data, jenis-jenis basis data, basis data relational, defenisi sistem dan analisis kebutuhan, pemodelan relasi entitas, normalisasi basis data, enhanced entiti relationship, desain basis data, data definition languange, data entry languange, data manipulation languange, data control languange dan administrasi basis data.

Selain itu juga membahas tentang implementasi database dengan menggunakan DBMS(Database Management System),

Serta dilengkapi pemahaman tentang SQL(Structured Query Languange). Dan buku ini dilengkapi dengan kasus-kasus serta pembahasannya untuk membantu pemahaman pembaca.

Dengan mempelajari buku sistem basis data ini pembaca akan memperoleh pemahaman tentang database dan bagaimana mengelola data secara efesien. Sistem basis data adalah salah satu dasar ilmu yang sangat penting dalam dunia teknologi informasi. Oleh sebab itu buku ini dapat dijadikan sebagai sumber bacaan bagi mahasiswa/i dan siapa pun yang ingin mempelajari sistem basis data dengan baik untuk tujuan akademis maupun profesional.

ISBN 978-623-09-6245-5

9 786230 962455

PT Penerbit Penamuda Media
Godean, Yogyakarta
085700592256
@penamuda_media
penamuda.com