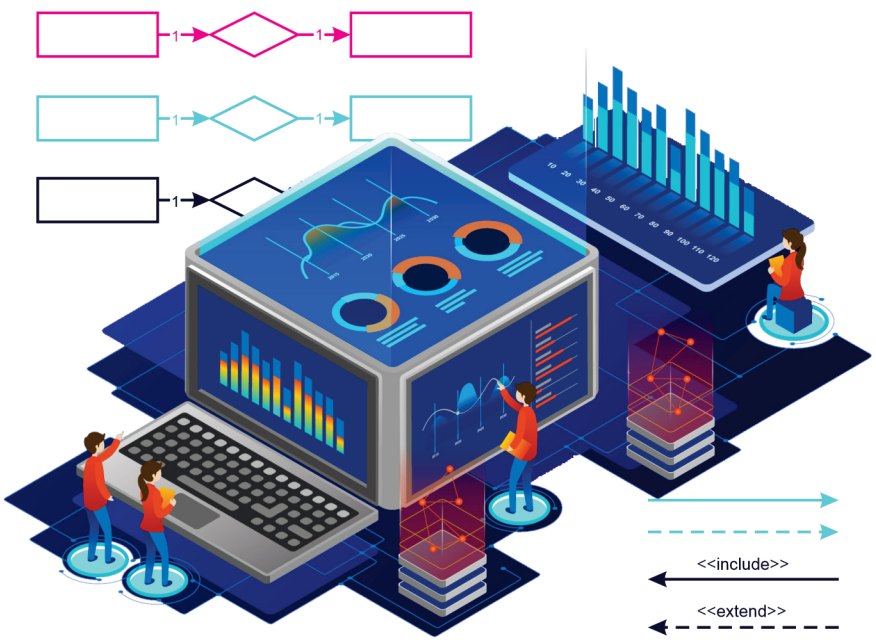
Penerbit litnus.

Pengantar
BASIS DATA



Arie Gunawan, S.Kom., M.M.S.I.
Sari Ningsih, S.Si., M.M.
Dhieka Avrilia Lantana, S.Kom., M.Kom.

PENGANTAR BASIS DATA

Ditulis oleh :

Arie Gunawan, S.Kom., M.M.S.I.

Sari Ningsih, S.Si., M.M.

Dhieka Avrilia Lantana, S.Kom., M.Kom.

Diterbitkan, dicetak, dan didistribusikan oleh
PT. Literasi Nusantara Abadi Grup
Perumahan Puncak Joyo Agung Residence Kav. B11 Merjosari
Kecamatan Lowokwaru Kota Malang 65144
Telp : +6285887254603, +6285841411519
Email: literasinusantaraofficial@gmail.com
Web: www.penerbitlitnus.co.id

Cetakan I, Mei 2023

Perancang sampul: Noufal Fahriza
Penata letak: Noufal Fahriza

ISBN : 978-623-8246-60-1
xii + 136 hlm. ; 15,5x23 cm.

KATA PENGANTAR

Puji dan syukur kita panjatkan kepada Allah Subhanahu wa ta'ala (الله سبحانه وتعالى) yang telah memberi rahmat dan hidayah-Nya sehingga penyusunan buku Pengantar Basis Data ini akhirnya bisa diselesaikan.

Basis data adalah komponen penting dalam dunia teknologi informasi, termasuk dalam pengembangan aplikasi dan sistem informasi kesehatan. Basis data digunakan untuk menyimpan dan mengelola informasi dalam jumlah besar dan kompleks, yang memungkinkan pengambilan keputusan yang lebih baik dalam berbagai bidang, termasuk bidang kesehatan. Oleh karena itu, pemahaman yang baik tentang basis data dan kemampuan untuk mengelolanya menjadi keterampilan penting bagi para profesional kesehatan.

Buku "Pengantar Basis Data" ini ditulis untuk memberikan pemahaman dasar tentang basis data dan cara mengelolanya secara efektif. Buku ini dirancang untuk para pembaca yang ingin mempelajari konsep-konsep dasar dalam basis data, termasuk jenis-jenis basis data, struktur data, model relasional, dan bahasa SQL. Selain itu, buku ini juga membahas topik-topik terkait seperti normalisasi, integritas referensial, dan transaksi.

Dalam penulisan buku ini, kami berusaha menyajikan materi dengan cara yang mudah dipahami, baik bagi pembaca yang belum pernah belajar tentang basis data maupun bagi pembaca yang ingin memperdalam pemahaman mereka tentang konsep-konsep dasar. Kami juga menyajikan berbagai contoh dan latihan untuk membantu pembaca mempraktikkan dan mengasah keterampilan mereka dalam mengelola basis data.

Kami berharap buku ini dapat membantu para pembaca untuk memahami dasar-dasar basis data dan memberikan dasar yang kuat bagi mereka untuk mempelajari lebih lanjut tentang topik ini.

Penulis menyadari masih banyak ketidaksempurnaan pada penulisan buku ini, baik isi maupun redaksinya, oleh karenanya kritik dan saran yang membangun diharapkan dapat memperbaiki buku ajar ini untuk tahun-tahun berikutnya.

Terakhir, saya ingin mengucapkan terima kasih kepada seluruh pihak yang telah membantu dalam pembuatan buku ini, dan semoga buku ini dapat memberikan manfaat bagi pembaca dalam meningkatkan kualitas pelayanan kesehatan di Indonesia.

Jakarta, Mei 2023

Tim Penyusun

PRAKATA

Selamat datang pada buku «Pengantar Basis Data». Basis data adalah suatu kumpulan data yang terorganisir dan tersimpan di dalam sistem komputer. Dalam era digital saat ini, penggunaan basis data sangat penting untuk mengelola informasi secara efektif dan efisien. Basis data memungkinkan pengguna untuk menyimpan, mengakses, dan memanipulasi data dengan mudah. Oleh karena itu, pengetahuan tentang basis data menjadi sangat penting, terutama bagi para profesional di bidang teknologi informasi dan komputer.

Buku ini ditujukan untuk memberikan pemahaman dasar tentang basis data. Buku ini dimulai dengan memperkenalkan konsep dasar basis data dan struktur basis data. Selain itu, buku ini juga membahas tentang bahasa query, pemrograman basis data, dan penggunaan basis data pada aplikasi. Buku ini juga mencakup topik-topik terkait seperti pemeliharaan basis data dan keamanan basis data.

Saya berharap buku ini akan menjadi sumber referensi yang berguna bagi para pembaca yang ingin mempelajari dasar-dasar basis data. Buku ini juga dapat digunakan sebagai bahan ajar bagi mahasiswa di bidang teknologi informasi dan komputer. Saya berterima kasih kepada semua yang telah memberikan dukungan dalam penulisan

buku ini, dan saya berharap buku ini dapat memberikan manfaat bagi para pembaca.

DAFTAR ISI

BAB 2

BAB 3

BAB 4

BAB 5

BAB 8

DATABASE MANAGEMENT SYSTEM 75

BAB 9

BAHASA BASIS DATA 81

BAB 10

SQL SERVER 89

BAB 11

DATA DEFINITION LANGUAGE (DDL) 109

BAB 12

DATA MANIPULATION LANGUAGE (DML) 117

BAB 1

PENDAHULUAN

Tujuan Pembelajaran

Mahasiswa memahami konsep dasar Basis Data.

Data dan Informasi

Data adalah kumpulan fakta mentah yang tidak memiliki arti atau konteks. Sedangkan informasi adalah data yang telah diolah atau diproses sehingga memiliki arti atau konteks yang bermanfaat bagi pengguna (Davis et al., 1985), (Stair & Reynolds, 2013), (Laudon & P., 2014). Contoh data adalah angka atau huruf, sedangkan informasi bisa berupa laporan atau analisis data.

Komponen Sistem Informasi

Sistem informasi terdiri dari beberapa komponen, yaitu input (masukan), proses, output (keluaran), perangkat keras, perangkat lunak, serta manusia atau pengguna sistem (O'Brien & Marakas, 2018), (Ralph M. Stair & Reynolds, 2019). Input adalah data yang dimasukkan ke dalam sistem, proses adalah pengolahan data menjadi informasi, output adalah hasil dari proses yang ditampilkan atau

dicetak, perangkat keras dan perangkat lunak adalah komponen teknis sistem, dan manusia adalah pengguna atau pelaku dalam sistem.

Basis Data

Basis Data adalah kumpulan data yang terorganisir dengan baik dan terstruktur di dalam suatu sistem komputer. Basis Data terdiri dari beberapa tabel yang terhubung dengan relasi atau hubungan tertentu. Basis Data digunakan untuk menyimpan, mengelola, dan mengakses data dengan efisien (Connolly & Begg, 2014), (Kroenke & Auer, 2016).

Sejarah Basis Data

Pada awal tahun komputer, "kartu punch" digunakan untuk input, output, dan penyimpanan data. Kartu punch menawarkan cara cepat untuk memasukkan data dan mengambilnya. Basis data komputer dimulai pada tahun 1960-an ketika penggunaan komputer menjadi pilihan yang lebih hemat biaya untuk organisasi swasta. Seiring dengan turunnya harga, menjadi lebih mudah untuk memindahkan penyimpanan data dan basis data ke komputer. Ada dua model data populer pada dekade ini: model jaringan dan model hierarki.

Model basis data relasional diperkenalkan pada tahun 1970 oleh E.F. Codd di IBM. Model ini didasarkan pada teori himpunan matematika dan dirancang untuk lebih fleksibel daripada model sebelumnya. Ini memungkinkan untuk kueri yang lebih kompleks dan membuatnya lebih mudah untuk mengelola jumlah data yang besar. Dalam dekade 1980-an terjadi perkembangan basis data berorientasi objek yang memungkinkan jenis data yang lebih kompleks seperti gambar dan video.

Pada tahun 1990-an, basis data menjadi lebih banyak digunakan dengan munculnya internet. Meningkatnya e-commerce berarti bahwa basis data digunakan untuk menyimpan informasi pelanggan dan data transaksi. Dalam beberapa tahun terakhir, terjadi pergeseran

ke basis data NoSQL yang dirancang untuk menangani jumlah data yang besar dan tidak terstruktur.

Kenapa Basis Data Diperlukan

Basis data diperlukan karena memungkinkan organisasi untuk menyimpan dan mengelola data dalam satu tempat. Basis data memungkinkan pengguna untuk mengakses informasi dengan lebih mudah dan cepat. Basis data juga memungkinkan pengguna untuk mengelola data dengan lebih efisien dan efektif. Dalam sistem informasi, basis data digunakan untuk menyimpan data yang diperlukan oleh organisasi. Basis data juga memungkinkan organisasi untuk melakukan analisis data dan membuat keputusan yang lebih baik.

Keuntungan Menggunakan Basis Data

Basis Data memberikan banyak keuntungan bagi pengguna dan organisasi, antara lain:

1. **Penghematan Waktu dan Biaya**

   Dengan Basis Data, informasi dapat diakses dan dikelola dengan cepat dan efisien, sehingga menghemat waktu dan biaya dalam memproses dan mengelola data. Basis Data juga memungkinkan pengguna untuk mengekstrak informasi dengan mudah dan cepat, sehingga memudahkan pengambilan keputusan.

2. **Konsistensi Data**

   Basis Data memastikan konsistensi data, sehingga informasi yang disimpan dalam Basis Data memiliki kualitas yang tinggi dan dapat diandalkan. Hal ini memudahkan pengguna untuk memahami dan menggunakan informasi yang tersimpan dalam Basis Data.

3. **Menghindari Duplikasi Data**

   Basis Data memastikan bahwa setiap data hanya tersimpan sekali dalam Basis Data, sehingga menghindari duplikasi data yang tidak perlu. Hal ini meminimalkan risiko kesalahan dan memudahkan pengguna untuk mengakses informasi yang akurat dan up-to-date.

4. **Memudahkan Akses dan Pembaruan Data**

   Basis Data memudahkan akses dan pembaruan data, sehingga pengguna dapat dengan mudah mengakses, memperbarui, dan mengelola informasi dalam Basis Data dari mana saja dan kapan saja. Hal ini memudahkan pengguna untuk bekerja secara fleksibel dan efisien.

5. **Menjaga Keamanan dan Privasi Data**

   Basis Data dapat memberikan tingkat keamanan dan privasi yang tinggi bagi data yang tersimpan di dalamnya. Hal ini memastikan bahwa informasi penting dan rahasia tetap aman dan terlindungi dari akses yang tidak sah.

6. **Analisis Data yang Lebih Baik**

   Basis Data memungkinkan pengguna untuk melakukan analisis data yang lebih baik dan mendalam, sehingga dapat menghasilkan wawasan bisnis yang lebih baik dan pengambilan keputusan yang lebih tepat. Analisis data dalam Basis Data juga memungkinkan pengguna untuk mengidentifikasi tren, pola, dan peluang bisnis yang mungkin terlewatkan sebelumnya.

7. **Mendukung Integrasi Sistem**

   Basis Data memudahkan integrasi dengan sistem lain, sehingga memudahkan pengguna untuk berbagi data dengan sistem lain yang mungkin digunakan oleh organisasi atau bisnis. Hal ini memungkinkan integrasi sistem yang lebih mudah dan efektif.

Kelemahan Menggunakan Basis Data

Meskipun Basis Data memberikan banyak keuntungan, namun ada juga beberapa kelemahan atau kekurangan yang perlu dipertimbangkan dalam penggunaannya, antara lain:

1. **Biaya yang Mahal**

   Pembuatan dan pemeliharaan Basis Data membutuhkan biaya yang cukup besar, terutama jika Basis Data digunakan dalam organisasi yang besar dengan banyak data yang harus diolah dan disimpan.

2. **Risiko Kehilangan Data**

   Meskipun Basis Data dirancang untuk melindungi data dari kerusakan atau kehilangan, namun ada risiko bahwa Basis Data dapat mengalami kerusakan atau kehilangan data secara tidak terduga, misalnya karena serangan hacker atau kegagalan sistem.

3. **Kerumitan dan Kesulitan dalam Implementasi**

   Implementasi Basis Data membutuhkan kemampuan teknis dan keahlian khusus untuk merancang, mengelola, dan memelihara Basis Data. Selain itu, Basis Data yang kompleks dapat membingungkan pengguna dan memerlukan waktu untuk menguasainya.

4. **Keterbatasan pada Skala Besar**

   Meskipun Basis Data dapat menangani banyak data, namun pada skala yang sangat besar, Basis Data dapat mengalami kinerja yang buruk atau bahkan crash. Hal ini memerlukan perencanaan dan desain Basis Data yang cermat dan hati-hati agar dapat menangani data dalam jumlah yang besar.

5. **Risiko Kebocoran Data atau Pelanggaran Keamanan**

   Basis Data yang diakses melalui internet atau jaringan dapat menjadi target serangan dan kebocoran data oleh pihak yang tidak bertanggung jawab. Kebocoran data dapat mengakibatkan

kerugian besar bagi organisasi atau bisnis, termasuk kerugian finansial dan reputasi.

6. **Kurangnya Fleksibilitas dalam Mengubah Skema Data**

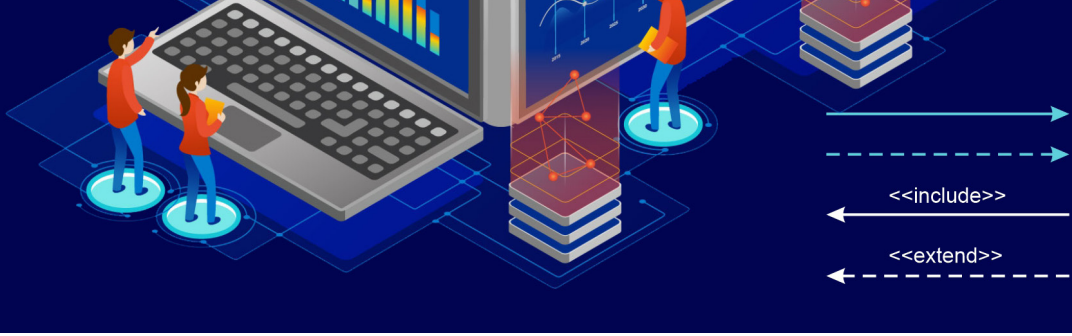
   Skema Basis Data yang sudah dibuat dapat sulit diubah atau diadaptasi, sehingga memerlukan perencanaan dan desain Basis Data yang matang agar dapat menyesuaikan dengan perubahan bisnis atau kebutuhan data di masa depan.

7. **Risiko Ketergantungan pada Teknologi**

   Basis Data bergantung pada teknologi, dan jika teknologi yang digunakan tidak mendukung lagi, maka Basis Data juga dapat menjadi usang. Hal ini memerlukan pembaruan dan upgrade Basis Data secara teratur untuk menjaga agar Basis Data selalu relevan dan mendukung kebutuhan bisnis atau organisasi.

Evaluasi / Soal Latihan

Pilihlah salah satu jenis basis data dan jelaskan kelebihan dan kekurangannya.



BAB 2

LINGKUNGAN BASIS DATA

Tujuan Pembelajaran

Mahasiswa akan memperoleh keterampilan dan pengetahuan yang dibutuhkan untuk menjadi profesional dalam bidang Basis Data dan teknologi informasi. Mahasiswa juga akan memahami pentingnya Basis Data dalam kehidupan modern dan dapat memanfaatkannya secara efektif untuk memecahkan masalah dan mencapai tujuan bisnis atau organisasi.

Arsitektur Basis Data

Arsitektur Basis Data mengacu pada desain dan struktur organisasi dari sebuah Basis Data yang mencakup komponen-komponen seperti basis data, aplikasi basis data, pengguna basis data, serta perangkat keras dan perangkat lunak yang terkait. Arsitektur Basis Data memberikan panduan tentang cara penggunaan dan pengelolaan Basis Data yang efektif dan efisien.

Komponen-komponen dari Arsitektur Basis Data meliputi:

1. **Basis Data**

   Basis Data merupakan kumpulan data yang terorganisir dengan cara tertentu dan terintegrasi di dalam sistem basis data. Basis Data terdiri dari satu atau beberapa tabel yang terdiri dari baris dan kolom yang merepresentasikan data secara terstruktur.

2. **Aplikasi Basis Data**

   Aplikasi Basis Data merupakan program atau aplikasi yang menggunakan Basis Data sebagai sumber daya informasi. Aplikasi Basis Data ini dapat digunakan untuk melakukan operasi seperti pencarian data, pembaruan data, dan penghapusan data dari Basis Data.

3. **Pengguna Basis Data**

   Pengguna Basis Data adalah individu atau kelompok yang menggunakan Basis Data untuk tujuan tertentu, seperti pengambilan keputusan atau pemrosesan data. Pengguna Basis Data dibagi menjadi beberapa kategori, seperti pengguna akhir, administrator basis data, dan pengembang aplikasi.

4. **Perangkat Keras dan Perangkat Lunak**

   Perangkat Keras dan Perangkat Lunak yang digunakan dalam Arsitektur Basis Data adalah perangkat keras seperti server, storage, dan perangkat lunak seperti sistem manajemen basis data (DBMS), bahasa query, dan aplikasi yang terkait dengan Basis Data.

Arsitektur Basis Data memainkan peran penting dalam menyediakan akses dan pengelolaan data yang aman dan terstruktur untuk organisasi atau bisnis. Arsitektur Basis Data yang baik dapat memastikan konsistensi dan integritas data, menghindari duplikasi data yang tidak perlu, serta memungkinkan integrasi dan interoperabilitas yang baik antara aplikasi dan sistem yang berbeda.

Selain itu, Arsitektur Basis Data juga membantu dalam memenuhi kebutuhan keamanan dan privasi data, serta meningkatkan performa Basis Data dan penggunaan sumber daya IT.

Klasifikasi Model Data

Model data adalah suatu representasi struktur dan sifat-sifat data dalam suatu sistem. Klasifikasi model data meliputi 3 kategori, yaitu model data konseptual, model data fisik, dan model data logikal.

1. **Model Data Konseptual**

   Model data konseptual adalah model yang mendefinisikan struktur data dalam suatu organisasi secara independen dari teknologi penyimpanan data dan perangkat lunak pengelola Basis Data yang digunakan. Model data konseptual memberikan gambaran besar mengenai struktur data pada suatu organisasi.

   Contoh model data konseptual adalah Entity-Relationship (ER) model, Dimensi Relasional (Relational Dimensional) model, dan Object Oriented model.

2. **Model Data Fisik**

   Model data fisik adalah model yang mendefinisikan struktur data pada tingkat teknis dan operasional dalam suatu organisasi. Model data fisik berhubungan dengan teknologi penyimpanan data dan perangkat lunak pengelola Basis Data yang digunakan.

   Contoh model data fisik adalah model data Hierarki, model data Jaringan, dan model data Relasional.

3. **Model Data Logikal**

   Model data logikal adalah model yang menyajikan struktur data dari suatu organisasi dalam suatu cara yang dapat dimengerti oleh pengguna. Model data logikal berhubungan dengan cara pengguna berinteraksi dengan data.

   Contoh model data logikal adalah model data Hierarki, model data Jaringan, model data Relasional.

Dalam Basis Data relasional, model data terdiri dari tabel dan hubungan di antara tabel. Kategori ini meliputi model data fisik, logikal dan konseptual. Contoh model data relasional termasuk model data Entity-Relationship (ER), model data Relational Dimensional, dan model data Object Oriented. Model-model ini memberikan pendekatan yang berbeda dalam menyelesaikan masalah-masalah Basis Data relasional dan dapat membantu para pengembang dalam merancang Basis Data yang sesuai dengan kebutuhan organisasi.

Bahasa Basis Data

Bahasa basis data (database language) adalah bahasa yang digunakan untuk mengelola dan memanipulasi basis data. Bahasa basis data dibagi menjadi beberapa jenis, yaitu Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), dan Data Query Language (DQL).

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

   DDL digunakan untuk mengatur struktur basis data, seperti membuat tabel, menghapus tabel, menambahkan kolom, dan sebagainya. Contoh perintah DDL adalah CREATE, ALTER, dan DROP.

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

   DML digunakan untuk memanipulasi data dalam basis data, seperti menambah, mengubah, dan menghapus data. Contoh perintah DML adalah SELECT, INSERT, UPDATE, dan DELETE.

3. **Data Control Language (DCL)**

   DCL digunakan untuk mengatur hak akses pengguna ke basis data, seperti memberikan hak akses dan mencabut hak akses. Contoh perintah DCL adalah GRANT dan REVOKE.

4. **Data Query Language (DQL)**

   DQL digunakan untuk mengekstraksi informasi dari basis data, seperti mencari data yang spesifik, menghitung jumlah data, dan sebagainya. Contoh perintah DQL adalah SELECT.

Bahasa basis data sangat penting dalam pengelolaan basis data karena dapat membantu pengguna untuk mengelola data secara efektif dan efisien. Selain itu, bahasa basis data juga dapat membantu pengguna untuk mengekstraksi informasi yang dibutuhkan dari basis data dan menjawab pertanyaan yang muncul terkait data.

Pengguna Basis Data

Pengguna basis data adalah individu atau organisasi yang menggunakan atau mengakses basis data. Berikut ini adalah rincian tentang pengguna basis data:

1. **Administrator Basis Data**

   Administrator Basis Data adalah orang yang bertanggung jawab untuk merancang, mengelola, memantau, dan memelihara sistem basis data. Tugas-tugas ini meliputi memastikan integritas dan keamanan data, mengoptimalkan kinerja sistem, dan memastikan bahwa data tersedia untuk pengguna yang berwenang.

2. **Pengembang Aplikasi**

   Pengembang Aplikasi adalah orang yang merancang dan mengembangkan aplikasi yang menggunakan data dari basis data. Mereka bertanggung jawab untuk memahami struktur data, melakukan pengambilan data, dan membuat aplikasi yang efektif dan efisien.

3. **Pengguna Akhir**

   Pengguna Akhir adalah orang yang menggunakan aplikasi yang dibangun dengan data dari basis data. Mereka dapat melakukan

tindakan seperti memasukkan data baru ke dalam sistem, mencari dan memperbarui data yang ada, dan menghasilkan laporan.

4. **Analis Data**

   Analis Data adalah orang yang bertanggung jawab untuk mengumpulkan dan menganalisis data dari basis data. Mereka menggunakan perangkat lunak analisis data untuk membantu memahami tren dan pola dalam data.

5. **Manajer**

   Manajer adalah orang yang menggunakan data dari basis data untuk membuat keputusan bisnis. Mereka membutuhkan data yang akurat dan terkini untuk memastikan keputusan yang tepat.

6. **Pemilik Bisnis**

   Pemilik Bisnis adalah orang yang memiliki kepentingan dalam data yang disimpan dalam basis data. Mereka dapat menggunakan data untuk memantau kinerja bisnis, mengidentifikasi peluang baru, dan membuat keputusan strategis.

Ketika pengguna basis data bekerja sama, mereka dapat memastikan bahwa data yang tersimpan dalam basis data akurat, terbaru, dan tersedia untuk pengguna yang membutuhkannya. Oleh karena itu, penting bagi pengguna basis data untuk bekerja sama dan mengikuti kebijakan dan prosedur yang ditetapkan oleh administrator basis data.

Evaluasi / Soal Latihan

1. Apa perbedaan antara basis data relasional dan basis data non-relasional? Jelaskan masing-masing kelebihan dan kekurangan.
2. Apa itu arsitektur basis data dan mengapa penting untuk memahaminya dalam lingkungan basis data?

BAB 3

PERMODELAN DATA

Tujuan Pembelajaran

Tujuan pembelajaran pemodelan data bagi mahasiswa adalah untuk mempelajari konsep, teknik, dan metode dalam merancang suatu sistem basis data. Dalam pembelajaran pemodelan data, mahasiswa diharapkan mampu memahami dan menerapkan konsep relasional, serta dapat merancang dan membangun database yang efisien, handal, dan mudah digunakan.

Model Data

Model data adalah representasi dari objek, konsep, atau informasi dalam bentuk struktur data, aturan, dan relasi yang disusun dalam format tertentu. Model data memungkinkan pengguna untuk memahami, mengorganisasi, dan mengelola data dengan lebih efisien dan efektif, terutama pada skala besar. Model data juga berfungsi sebagai panduan dalam mengembangkan aplikasi dan sistem yang menggunakan data. Model data bisa berupa diagram, tabel, atau objek yang menggambarkan hubungan antara data dan entitas yang terkait. Dalam pengembangan perangkat lunak atau aplikasi, model

data merupakan dasar untuk merancang dan membangun database dan sistem informasi.

Dalam konteks pengembangan perangkat lunak atau aplikasi, model data sangat penting karena berfungsi sebagai panduan untuk merancang, membangun, dan mengelola database. Model data juga membantu memastikan bahwa data yang dimasukkan ke dalam sistem benar-benar akurat, konsisten, dan dapat diandalkan. Dalam hal ini, model data digunakan sebagai alat untuk menyusun rencana bisnis, analisis persyaratan, dan desain sistem.

Ada berbagai jenis model data, masing-masing memiliki kegunaan dan kelebihan yang berbeda-beda. Beberapa jenis model data yang umum digunakan antara lain:

1. Model Hierarki: Model ini menggambarkan data dalam bentuk pohon, di mana setiap node dihubungkan dengan satu node lain. Contoh penggunaan model ini adalah untuk mengorganisasi data dalam sistem informasi manajemen keuangan.
2. Model Jaringan: Model ini menggambarkan data dalam bentuk jaringan, di mana setiap node dapat terhubung dengan beberapa node lain. Contoh penggunaan model ini adalah untuk mengorganisasi data dalam sistem informasi manajemen inventaris.
3. Model Relasional: Model ini menggunakan tabel untuk menggambarkan hubungan antara data. Setiap tabel mewakili suatu entitas, dengan setiap baris di tabel mewakili instance dari entitas tersebut. Contoh penggunaan model ini adalah untuk mengorganisasi data dalam sistem manajemen inventaris dan sistem manajemen pelanggan.
4. Model E-R (Entity-Relationship): Model ini menggunakan diagram untuk menggambarkan entitas, atribut, dan hubungan antara entitas. Contoh penggunaan model ini adalah untuk mengorganisasi data dalam sistem manajemen pelanggan.

Dalam pembelajaran permodelan data, mahasiswa akan mempelajari bagaimana merancang, membangun, dan mengelola database dengan menggunakan model-model data tersebut. Mahasiswa juga akan mempelajari cara menggunakan perangkat lunak khusus untuk membangun dan mengelola database, seperti Microsoft Access, MySQL dan Microsoft SQL Server. Tujuan dari pembelajaran ini adalah untuk memastikan bahwa mahasiswa memiliki pemahaman yang kuat tentang model data dan kemampuan praktis dalam merancang, membangun, dan mengelola database.

Model Data Berbasis Objek

Model Data Berbasis Objek (Object-Based Data Model) adalah model data yang didasarkan pada konsep-konsep pemrograman berorientasi objek, yang digunakan untuk merepresentasikan data dalam bentuk objek-objek. Model data ini menggunakan konsep inheritance, encapsulation, dan polymorphism, yang merupakan fitur-fitur kunci dalam pemrograman berorientasi objek.

Objek dalam model data ini terdiri dari atribut dan metode. Atribut adalah data yang terkait dengan objek tersebut, sedangkan metode adalah operasi atau fungsi yang dapat dilakukan pada objek tersebut. Contoh objek dalam model data ini adalah objek mahasiswa, yang memiliki atribut seperti nama, alamat, dan nomor telepon, serta metode seperti mengambil dan mengubah data mahasiswa.

Keuntungan dari penggunaan model data berbasis objek antara lain:

1. Dapat merepresentasikan dunia nyata dengan lebih baik, karena objek-objek yang direpresentasikan dalam model ini mirip dengan objek-objek dalam dunia nyata.
2. Dapat meningkatkan fleksibilitas dan skalabilitas dari sistem basis data, karena model ini dapat memperbolehkan adanya perubahan struktur data tanpa mengganggu aplikasi yang menggunakan data tersebut.

3. Dapat meningkatkan kinerja dan efisiensi dari sistem basis data, karena objek-objek yang terkait dapat diproses secara bersamaan.

Namun, model data berbasis objek juga memiliki kelemahan, seperti kompleksitas model yang tinggi, serta keterbatasan dalam interoperabilitas dengan model data yang lain. Oleh karena itu, pemilihan model data yang tepat harus didasarkan pada kebutuhan bisnis dan karakteristik data yang akan diolah.

Model Data Berbasis Record

Model Data Berbasis Record (Record-Based Data Model) adalah suatu model data yang menyimpan data dalam bentuk record atau kumpulan data terstruktur yang saling terkait. Model data ini sangat umum digunakan dalam sistem manajemen basis data relasional. Dalam model data ini, data disimpan dalam tabel yang terdiri dari baris dan kolom.

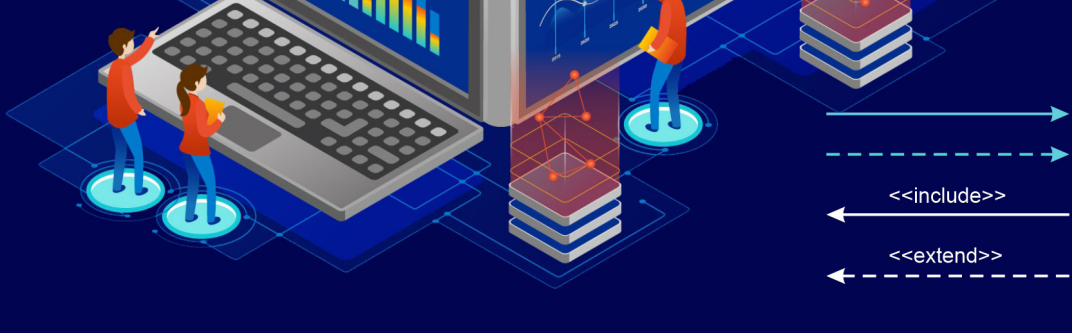
Setiap tabel dalam model data berbasis record memiliki sebuah kunci atau primary key yang membedakan setiap record dalam tabel tersebut. Setiap kolom dalam tabel mewakili sebuah atribut atau karakteristik dari entitas atau objek yang direpresentasikan dalam tabel tersebut. Contohnya, dalam sebuah tabel pelanggan, setiap record mewakili sebuah pelanggan dengan atribut seperti nama, alamat, nomor telepon, dan sebagainya.

Model data berbasis record sangat mudah dipahami dan digunakan oleh para pengguna yang terbiasa dengan spreadsheet atau aplikasi tabel seperti Microsoft Excel. Namun, model data ini terbatas dalam kemampuannya untuk merepresentasikan hubungan yang kompleks antara entitas atau objek dalam basis data.

Beberapa contoh dari sistem manajemen basis data yang menggunakan model data berbasis record antara lain MySQL, PostgreSQL, dan Microsoft SQL Server.

Evaluasi / Soal Latihan

Jelaskan perbedaan model data berbasis objek dengan model data berbasis record. Berikan contohnya!



BAB 4

ENTITY RELATIONSHIP DIAGRAM

Tujuan Pembelajaran

Mahasiswa mampu memahami konsep ERD dan diharapkan dapat menggambarkan ERD.

Pengertian

Entity Relationship Diagram (ERD) adalah sebuah model data yang digunakan untuk menggambarkan hubungan antar entitas (objek) dalam sebuah sistem informasi. ERD dapat membantu pengembang perangkat lunak untuk memahami, merancang, dan mengimplementasikan struktur database.

Menurut beberapa ahli, pengertian Entity Relationship Diagram adalah sebagai berikut:

1. "ERD adalah suatu model diagramatis yang dibuat untuk memvisualisasikan struktur suatu basis data melalui penggunaan simbol-simbol dan hubungan antar entitas" (Connolly & Begg, 2014).
2. "ERD adalah suatu model data diagramatis yang menggambarkan struktur dari sebuah basis data dengan menggunakan entitas, atribut, keterhubungan, dan kardinalitas" (Elmasri & Navathe, 2016).

3. "ERD adalah suatu model data konseptual yang digunakan untuk mendokumentasikan secara visual hubungan antar data di dalam suatu organisasi" (Kendall & Kendall, 2014).

Adapun fungsi dari ERD adalah sebagai berikut:

1. Memberikan kemudahan dalam menganalisis sebuah basis data (database) dengan cara yang cepat serta murah.
2. Menjalankan hubungan antar data yang memiliki keterkaitan berdasarkan objek yang dihubungkan dengan suatu relasi.
3. Mendokumentasikan data yang ada dalam sebuah basis data dengan cara menganalisis serta mengidentifikasi setiap objek atau entitas dan relasinya.
4. Melakukan pengujian model yang telah dibuat.

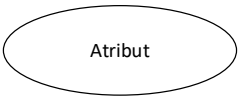
Komponen-Komponen ERD

ERD terdiri dari tiga komponen utama yaitu Entitas, Atribut, Relasi, dan Garis.

1. Entitas

Entitas adalah objek atau konsep yang diidentifikasi dan memiliki nilai yang ingin disimpan dalam database. Entity direpresentasikan oleh kotak persegi panjang dan diberi label nama.

2. Atribut



Atribut adalah properti dari suatu entity yang mendeskripsikan karakteristik dari entity tersebut. Atribut direpresentasikan oleh

oval kecil dan diberi label nama. Contoh atribut pada entity Mahasiswa bisa berupa Nama, NIM, Alamat, dll.

Berikut merupakan beberapa jenis atribut yang sering digunakan:

a. Atribut Kunci

   Merupakan atribut yang digunakan untuk menentukan data yang bersifat unik. Pada umumnya, data dari atribut key berbentuk angka. Contohnya NIM (Nomor Induk Mahasiswa), No. KTP, SIM, NPWP, dan lain sebagainya.

b. Atribut Simpel

   Yaitu atribut yang tidak dapat dipecah lagi atau atomic dan bernilai tunggal. Contohnya adalah alamat rumah, kantor, nama penerbit, tahun terbit jurnal, dan lain – lain.

c. Atribut Multinilai (Multivalue)

   Merupakan atribut yang memiliki sekelompok nilai untuk setiap entitas -nya. Contoh dari atribut multivalue adalah kumpulan nama pengarang dalam sebuah novel.

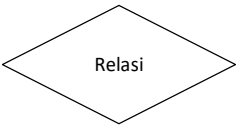
d. Atribut Gabungan (Composite)

   Yaitu atribut yang berasal dari susunan atribut yang lebih kecil dalam artian tertentu. Contohnya adalah data terkait nama lengkap, yang terdiri dari nama depan, tengah, dan belakang.

e. Atribut Derivatif

   Merupakan atribut yang berasal dari atribut lain dan tidak bersifat wajib untuk ditulis pada ERD. Contohnya adalah usia, selisih waktu, kelas atau ruang, dan lain sebagainya.

3. Relasi



Relasi adalah keterkaitan atau koneksi antara dua atau lebih entitas dalam sebuah database. Relasi direpresentasikan oleh garis dan diberi label nama. Ada tiga jenis hubungan, yaitu One-to-One, One-to-Many, dan Many-to-Many.

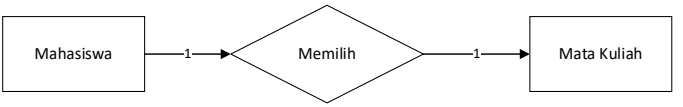
Terdapat tiga jenis relasi yang digunakan dalam ERD, diantaranya adalah sebagai berikut:

a. One to One (1:1)

Yang berarti, setiap entitas hanya boleh memiliki relasi dengan satu entitas yang lain. Contohnya adalah data mahasiswa dengan data NIM.

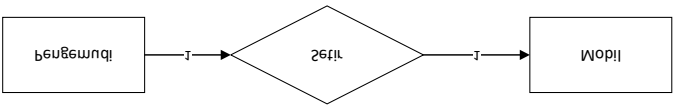
Contohnya:

- Seorang mahasiswa hanya dapat memilih satu mata kuliah, atau satu mata kuliah hanya dapat dipilih seorang mahasiswa



Gambar 1 .**4** One to One

- Seorang pengemudi hanya menyetir satu mobil, atau satu mobil disetir oleh seorang pengemudi.



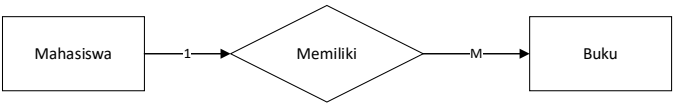
Gambar 2 .**4** One to One

b. One to Many (1:M)

Merupakan hubungan antara satu entitas dengan beberapa entitas, dan begitu pula sebaliknya. Contoh implementasinya adalah data terkait guru dengan siswa Sekolah Dasar (SD).

Contohnya:

- Seorang mahasiswa memiliki beberapa buku, atau beberapa buku dimiliki satu orang mahasiswa



Gambar 3 .**4** One to Many

- Seorang dosen mengajar beberapa mahasiswa, atau beberapa mahasiswa diajar oleh satu orang dosen.

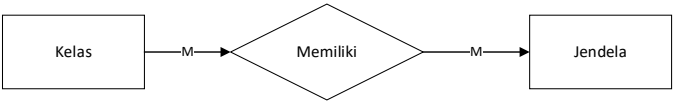
Gambar 4 .**4** One to Many

c. Many to Many (M:M)

Merupakan hubungan antara beberapa entitas yang memiliki lebih dari satu relasi. Contohnya adalah siswa SMP dengan data terkait ekstrakurikuler yang tersedia.

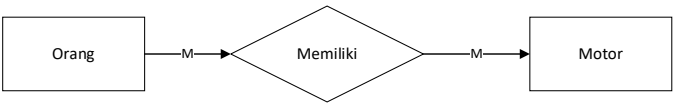
Contohnya:

- Beberapa kelas memiliki beberapa jendela, atau beberapa jendela dimiliki oleh beberapa kelas.



Gambar 5 .**4** Many to Many

- Beberapa orang memiliki beberapa motor, atau beberapa motor dimiliki oleh beberapa orang.



Gambar 6 .**4** Many to Many

4. Garis

Garis digunakan sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

Jenis-Jenis ERD

Terdapat tiga jenis ERD yaitu ERD Konseptual, ERD Logis, dan ERD Fisik.

1. ERD Konseptual

ERD Konseptual adalah jenis ERD yang menjelaskan hubungan antara entitas secara konseptual tanpa memperhatikan implementasi teknisnya. ERD konseptual digunakan untuk merancang konsep database.

2. ERD Logis

ERD Logis adalah jenis ERD yang menjelaskan hubungan antara entitas secara teknis dan terkait dengan database. ERD logis digunakan untuk merancang model database secara logis.

3. ERD Fisik

ERD Fisik adalah jenis ERD yang menjelaskan implementasi teknis dari ERD logis. ERD fisik digunakan untuk membuat database secara fisik.

Langkah-Langkah Membuat ERD

Berikut adalah langkah-langkah dalam membuat Entity Relationship Diagram (ERD):

1. Identifikasi entitas-entitas: Identifikasi entitas utama yang ingin dicatat dalam basis data. Entitas-entitas ini dapat berupa objek, orang, tempat, atau konsep lain yang terkait dengan sistem yang akan dibangun.
2. Identifikasi relasi antar entitas: Identifikasi hubungan antara entitas-entitas yang telah diidentifikasi. Terdapat tiga jenis relasi yang dapat didefinisikan dalam ERD, yaitu relasi satu-satu (one-to-one), relasi satu-banyak (one-to-many), dan relasi banyak-banyak (many-to-many).
3. Definisikan atribut: Setelah entitas dan relasi antara entitas-entitas diidentifikasi, tentukan atribut yang diperlukan untuk setiap entitas. Atribut adalah karakteristik yang digunakan untuk mendefinisikan atau menggambarkan entitas. Setiap entitas harus memiliki atribut yang menggambarkan entitas tersebut dengan baik.
4. Buat diagram ERD: Setelah semua elemen telah diidentifikasi dan didefinisikan, buatlah diagram ERD dengan memasukkan setiap entitas sebagai kotak dan relasi sebagai garis yang menghubungkan kotak-kotak tersebut. Atribut juga dapat ditambahkan ke dalam kotak entitas.
5. Normalisasi: Normalisasi adalah proses memastikan bahwa ERD telah dibuat dengan cara yang efisien dan efektif, sehingga tidak ada duplikasi data dan konsistensi data dapat dijaga dengan baik.
6. Review dan validasi: Terakhir, review dan validasi ERD untuk memastikan bahwa diagram tersebut memenuhi kebutuhan dan persyaratan yang telah ditetapkan. Review juga membantu memastikan bahwa ERD tidak memiliki kekurangan atau kesalahan yang dapat memengaruhi kinerja basis data.

Saat ini, terdapat banyak sekali cara untuk membuat diagram ER secara cepat, cukup dengan menghubungkan perangkat komputer anda dengan jaringan internet. Anda dapat membuat ERD dengan mengakses aplikasi berbasis web yang berupa tools online ataupun yang berbayar yang bisa digunakan secara offline.

1. **Draw.io**

   Tool ini cukup casual dan sangat mudah untuk diguanakan. Bentuk penyimpanannya berupa berbasis cloud dapat dapat digunakan untuk membuat flowchart.

2. **Dbdiagram.io**

   Dbdiagram.io merupakan tool yang dapat digunakan untuk membuat diagram ER dan mendesain database secara cepat. Tool ini juga menggunakan bahasa yang mudah dan bersifat open source.

3. **Lucidchart**

   Lucidchart biasanya digunakan oleh para desainer profesional untuk memudahkan pekerjaannya dalam merancang model ERD. Lucidchart menawarkan tampilan interface yang baik, namun berbayar. Anda jangan khawatir, karena Lucidchart juga menyediakan free version untuk anda yang baru belajar untuk membuat diagram.

4. **QuickDBD**

   Tool ini berbasis teks dan sangat cepat untuk menggambar diagram yang diperlukan. QuickDBD juga menyediakan file export dalam berbagai format, seperti PDF, SQL, maupun Word.

5. **SQLDBM**

   Dan tool yang terakhir yang bisa digunakan secara online dan gratis adalah SQLDBM yang digunakan untuk menjalankan database berbasis MySQL. Kelebihan utama dari SQLDBM

adalah dapat bekerja pada browser apapun dan tidak memerlukan database engine tambahan.

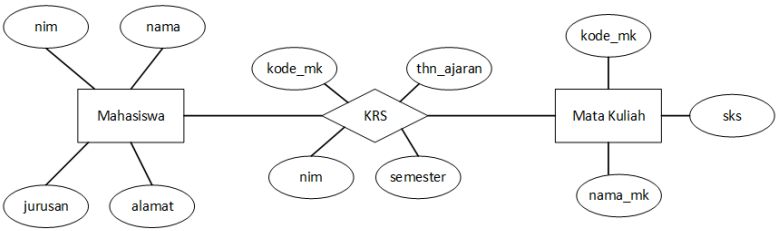
6. **Microsoft Visio**

   Untuk Microsoft Visio merupakan tools berbayar ataupun berlangganan. Microsoft Visio adalah sebuah program aplikasi komputer yang sering digunakan untuk membuat diagram, diagram alir, brainstorm, dan skema jaringan yang dirilis oleh Microsoft Corporation.

Contoh ERD

Berikut ini adalah beberapa contoh ERD berdasarkan studi kasus disertasi penjelasan.

1. **ERD Satu**



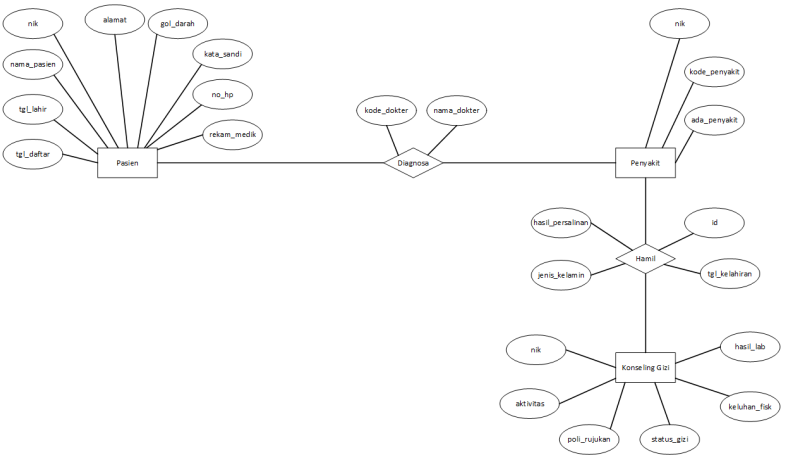
Gambar 4. 7 ERD satu

Entitas	Atribut
Mahasiswa Mata Kuliah	nim, nama, jurusan, alamat, kode_mk, nama_mk, sks

Gambar ERD satu di atas memiliki 2 entitas dengan nama mahasiswa dan mata kuliah, serta satu proses yang menghubungkan keduanya. Entitas mahasiswa memiliki atribut seperti nim, nama, jurusan dan alamat. Sedangkan mata kuliah memiliki kode_mk, nama_mk_mk dan sks. Proses krs memiliki masukan berupa nim, kode_mk, thn_ajaran dan semester. Dalam

proses di atas setiap mahasiswa yang ingin mengontrak mata kuliah diharuskan memberikan atribut yang terdapat dalam proses.

2. ERD dua



Gambar 4. 8 ERD Dua

Entitas	Atribut
Pasien Penyakit Konseling Gizi	Nik, nama_pasien, tgl_lahir, tgl_daftar, alamat, gol_darah, kata_sandi, no_hp, rekam_medik, kode_dokter, nama_dokter, kode_penyakit, ada_penyakit, aktivitas, poli_rujukan, status_gizi, hasil_lab, keluhan_fisik

Gambar ERD dua di atas memiliki tiga entitas, yaitu pasien, penyakit dan konseling gizi. Sedangkan untuk prosesnya memiliki dua proses.

a. Proses pertama yaitu diagnosa yang menghubungkan antara pasien dengan penyakit.
b. Proses kedua yaitu hamil yang menghubungkan antara penyakit dengan konseling gizi.

Evaluasi / Soal Latihan

1. Apa itu Entity Relationship Diagram (ERD) dan bagaimana cara membuatnya? Jelaskan langkah-langkahnya.
2. Buatlah sebuah ERD untuk sebuah sistem informasi peminjaman buku di perpustakaan.

BAB 5

DATA FLOW DIAGRAM (DFD)

Tujuan Pembelajaran

Mahasiswa mampu menghasilkan desain basis data yang baik, efisien, dan tidak redundan. Dengan memahami normalisasi, mahasiswa akan mampu mengoptimalkan struktur basis data, menghindari masalah yang mungkin terjadi pada data, dan meningkatkan kinerja sistem basis data. Selain itu, dengan pemahaman yang baik tentang normalisasi, mahasiswa juga akan mampu menganalisis struktur basis data yang sudah ada dan melakukan perbaikan jika diperlukan.

Pengertian DFD

Data Flow Diagram (DFD) adalah suatu teknik pemodelan yang digunakan untuk menggambarkan aliran data dan proses dalam suatu sistem. DFD digunakan untuk memvisualisasikan data yang bergerak di dalam sistem dari satu proses ke proses lainnya, serta memberikan gambaran mengenai bagaimana data diproses dalam sistem tersebut. DFD merupakan alat yang berguna dalam pengembangan sistem, karena memungkinkan para analis sistem untuk memahami sistem

secara keseluruhan, serta memudahkan dalam identifikasi kelemahan dan masalah sistem yang mungkin terjadi.

Data Flow Diagram (DFD) pertama kali diperkenalkan oleh Larry Constantine pada tahun 1970-an. Larry Constantine adalah seorang ahli dalam bidang software engineering dan juga penulis buku «Structured Design». DFD awalnya dikenal sebagai «flow-maps», yang digunakan untuk menggambarkan aliran informasi dalam sistem informasi bisnis pada saat itu. Dalam bukunya yang berjudul «Structured Design», Constantine memperkenalkan DFD sebagai alat untuk pemodelan sistem yang dapat membantu para analis sistem dalam mengidentifikasi kebutuhan pengguna dan dalam merancang sistem yang efektif dan efisien.

Pada tahun 1980-an, Gane dan Sarson mengembangkan DFD sebagai bagian dari metodologi analisis dan desain sistem mereka yang dikenal sebagai «Structured Systems Analysis and Design Method (SSADM)». SSADM merupakan salah satu metodologi yang paling populer pada saat itu untuk pengembangan sistem informasi.

Pada tahun 1990-an, James Martin mengembangkan DFD sebagai bagian dari metodologi analisis dan desain sistem yang dikenal sebagai «Information Engineering». Information Engineering merupakan salah satu metodologi yang paling populer pada saat itu untuk pengembangan sistem informasi.

Sejak itu, DFD telah menjadi alat yang sangat penting dalam pengembangan sistem informasi. DFD digunakan oleh para analis sistem untuk memahami aliran informasi dalam sistem, memodelkan sistem, serta merancang sistem yang efektif dan efisien. DFD juga digunakan oleh para pengembang perangkat lunak untuk merancang aplikasi yang dapat mengakomodasi kebutuhan pengguna dan mengoptimalkan penggunaan sumber daya.

Seiring dengan perkembangan teknologi informasi, DFD juga mengalami beberapa pengembangan dan modifikasi. Saat ini, DFD masih digunakan oleh banyak organisasi dan perusahaan dalam pengembangan sistem informasi, serta menjadi salah satu teknik

pemodelan yang paling populer dan efektif dalam pengembangan sistem informasi.

Tujuan dari pembuatan Data Flow Diagram (DFD) adalah untuk menggambarkan aliran informasi yang terjadi dalam sistem informasi bisnis. DFD dapat membantu para analis sistem untuk memahami proses bisnis, mengidentifikasi kebutuhan pengguna, merancang dan mengoptimalkan sistem informasi.

Berikut adalah tujuan dari pembuatan DFD secara lebih detail:

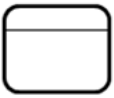
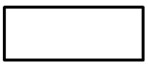
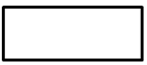
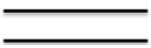
1. Memahami proses bisnis: DFD dapat membantu para analis sistem untuk memahami bagaimana suatu bisnis atau organisasi berjalan. Dengan melihat aliran informasi dalam sistem, analis sistem dapat mengetahui bagaimana suatu proses bisnis dijalankan, siapa yang terlibat dalam proses tersebut, dan apa yang menjadi output dari proses tersebut.
2. Mengidentifikasi kebutuhan pengguna: DFD juga dapat membantu para analis sistem untuk mengidentifikasi kebutuhan pengguna dan menentukan fitur-fitur yang harus ada dalam sistem. DFD dapat membantu dalam menentukan informasi apa yang diperlukan oleh pengguna, bagaimana informasi tersebut dihasilkan, dan bagaimana informasi tersebut digunakan.
3. Merancang sistem informasi: DFD dapat membantu para analis sistem dalam merancang sistem informasi yang efektif dan efisien. Dengan melihat aliran informasi dalam sistem, para analis sistem dapat menentukan bagaimana proses bisnis dapat dioptimalkan, apa yang harus dilakukan untuk menghindari kesalahan atau redundansi, dan bagaimana informasi dapat diproses dengan cepat dan akurat.
4. Menjelaskan sistem informasi: DFD dapat membantu dalam menjelaskan sistem informasi kepada pengguna atau pemangku kepentingan lainnya. DFD dapat digunakan sebagai alat komunikasi yang efektif untuk menjelaskan aliran informasi dalam sistem, sehingga semua pihak dapat memahami sistem dengan mudah.

5. Mengidentifikasi masalah dan kelemahan sistem: DFD juga dapat membantu dalam mengidentifikasi masalah dan kelemahan dalam sistem informasi. DFD dapat membantu para analis sistem untuk melihat di mana terjadi kesalahan atau redundansi, di mana proses bisnis dapat dioptimalkan, dan bagaimana informasi dapat diproses dengan lebih baik.

Dengan demikian, pembuatan DFD sangat penting dalam pengembangan sistem informasi, karena DFD dapat membantu para analis sistem untuk memahami, merancang, dan mengoptimalkan sistem informasi yang efektif dan efisien.

Notasi DFD

Notasi-notasi yang digunakan dalam DFD (Data Flow Diagram) memiliki tujuan untuk memberikan representasi visual dari aliran data dan proses yang terjadi dalam sebuah sistem. Berikut adalah penjelasan notasi-notasi yang umum digunakan dalam DFD:

Keterangan	DeMarco & Yourdan Simbol	Gane & Sarson Simbol
Proses		
Aliran Data		
Entitas		
Penyimpanan Data		

1. Proses

Proses dalam DFD direpresentasikan sebagai sebuah lingkaran atau kotak dengan nama proses di dalamnya. Proses ini merupakan aktivitas atau tindakan yang dilakukan pada data dalam sistem. Lingkaran atau kotak proses biasanya diberi nomor identifikasi yang unik dan deskripsi singkat tentang apa yang dilakukan oleh proses tersebut.

2. **Aliran Data**

   Aliran data dalam DFD direpresentasikan sebagai panah yang menghubungkan kotak proses dan entitas penyimpan data. Panah tersebut menggambarkan aliran data dari satu entitas ke entitas lain atau dari satu proses ke proses lain. Setiap panah biasanya diberi label untuk menjelaskan jenis data yang dihasilkan atau diproses.

3. **Entitas**

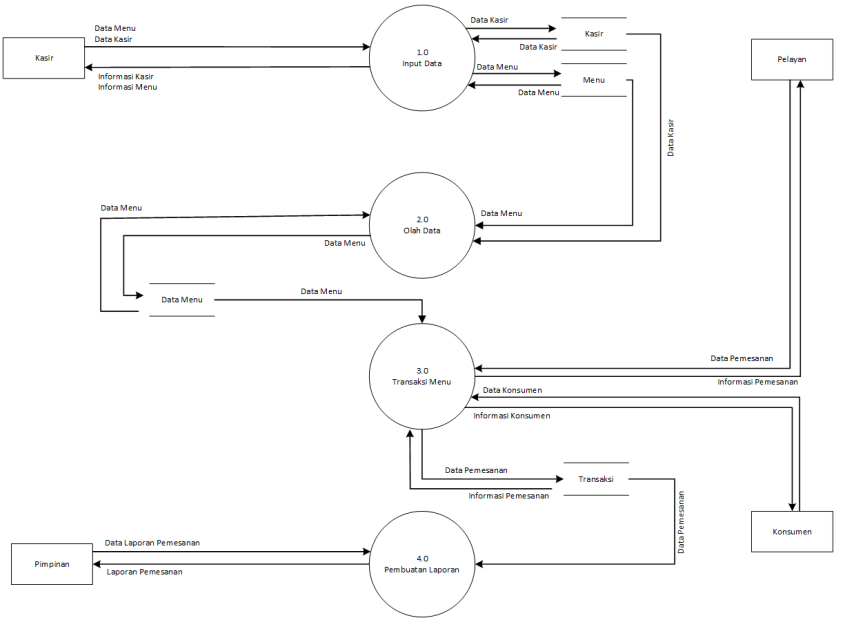
   Entitas dalam DFD direpresentasikan sebagai persegi panjang yang merepresentasikan penyimpan data dalam sistem. Entitas bisa berupa file, database, tabel atau dokumen lainnya yang menyimpan data yang digunakan dalam sistem. Entitas biasanya diberi nama dan nomor identifikasi unik.

4. **Penyimpanan Data**

   Penyimpanan data dalam DFD direpresentasikan sebagai sebuah entitas dengan nama file atau tabel yang menyimpan data. Entitas ini biasanya diberi label untuk menjelaskan jenis data yang disimpan, seperti data pelanggan atau data produk. Terdapat dua jenis penyimpanan data dalam DFD, yaitu penyimpanan data tetap dan penyimpanan data sementara.

Notasi-notasi dalam DFD memiliki tujuan untuk memberikan representasi visual yang mudah dipahami tentang aliran data dan proses dalam sistem. Hal ini memudahkan analis sistem dalam memahami dan merancang sistem dengan lebih efektif, serta membantu tim pengembang dalam mengimplementasikan sistem secara efektif dan efisien.

Berikut ini adalah contoh penggunaan notasi-notasi dalam DFD pada sebuah sistem pemesanan makanan di sebuah restoran:



1. Proses

Proses dalam sistem pemesanan makanan dapat berupa:

a. Input Data: Proses input data menu dan data kasir sehingga menghasilkan informasi.
b. Olah Data: Database menu dan kasir di olah datanya menjadi sebuah data menu.
c. Transaksi Menu: Proses yang mengolah data konsumen dan data pemesanan menjadi sebuah informasi.
d. Pembuatan Laporan: Proses yang mengolah data menu, data kasir, data konsumen, data pemesanan serta data laporan pemesanan.

2. Aliran Data

Aliran data dalam sistem pemesanan makanan dapat berupa:

a. Data Menu: Data yang yang diinput oleh kasir berupa menu makanan.
b. Data Kasir: Data yang diinput oleh kasir.

c. Informasi Menu: Data yang dikirimkan oleh sistem berupa informasi menu.
d. Informasi Kasir: Data yang dikirimkan oleh sistem berupa informasi kasir.
e. Data Konsumen: Data yang diinput oleh konsumen.
f. Informasi Konsumen: Data yang dikirimkan oleh sistem berupa informasi konsumen.
g. Data Pemesanan: Data yang diinput oleh pelayan ke dalam sistem berupa pesanan menu konsumen.
h. Informasi Pemesanan: Data yang dikirimkan oleh sistem berupa informasi pemesanan
i. Data Laporan Pemesanan: Data yang diinput oleh pimpinan ke dalam sistem berupa data laporan pemesanan.
j. Laporan Pemesanan: Data yang dikirimkan oleh sistem berupa laporan pemesanan

3. Entitas

Entitas dalam sistem pemesanan makanan dapat berupa:

a. Kasir: Entitas yang merepresentasikan kasir yang melakukan input menu dan data kasir.
b. Pelayan: Entitas yang merepresentasikan pelayan yang mencatat data pesanan makanan konsumen.
c. Konsumen: Entitas yang merepresentasikan konsumen yang melakukan pemesanan makanan.
d. Pimpinan: Entitas yang merepresentasikan data mengenai pemilik restoran.

4. Penyimpanan Data

Penyimpanan data dalam sistem pemesanan makanan dapat berupa:

a. Database Kasir: Database yang menyimpan data mengenai kasir.
b. Database Menu: Database yang menyimpan data mengenai menu.

c. Database Data Menu: Database yang menyimpan data mengenai menu yang telah diolah.
d. Database Transaksi: Database yang menyimpan data mengenai transaksi pemesanan makanan.

Level DFD

DFD terdiri dari beberapa level, yang masing-masing level menggambarkan tingkat detail yang berbeda dari aliran data dalam sistem. Berikut ini adalah penjelasan tentang masing-masing level DFD:

1. **Level 0**

   Level 0 DFD adalah gambaran awal dari sebuah sistem informasi, yang menggambarkan hubungan antara input, proses, dan output secara global. Pada level ini, sistem hanya digambarkan sebagai sebuah kotak tunggal yang menerima input dan menghasilkan output. Level 0 DFD ini biasanya digunakan untuk memberikan gambaran umum tentang sistem yang akan dikembangkan.

2. **Level 1**

   Level 1 DFD adalah gambaran yang lebih detail tentang aliran data dalam sistem. Pada level ini, sistem dibagi menjadi beberapa proses utama dan input/output yang berbeda. Setiap proses utama dijelaskan secara terpisah dan terdapat deskripsi tentang input dan output yang dihasilkan oleh setiap proses tersebut.

3. **Level 2**

   Level 2 DFD adalah gambaran yang lebih terperinci lagi tentang aliran data dalam sistem. Pada level ini, setiap proses pada level 1 dibagi lagi menjadi beberapa sub-proses yang lebih kecil dan detail. Setiap sub-proses dijelaskan dengan lebih rinci dan deskripsi tentang input dan output yang dihasilkan oleh setiap sub-proses tersebut.

4. **Level selanjutnya**

   Level selanjutnya dari DFD dapat dibuat jika diperlukan untuk lebih mendetailkan aliran data dalam sistem. Level-level berikutnya terdiri dari sub-proses yang lebih kecil lagi dan lebih detail tentang input dan output yang dihasilkan. Level-level yang lebih tinggi biasanya digunakan untuk mendeskripsikan sistem secara keseluruhan, sedangkan level yang lebih rendah digunakan untuk menggambarkan detail yang lebih spesifik tentang sistem.

Dalam membuat DFD, penting untuk menentukan level yang sesuai dengan kebutuhan analisis. Jika level terlalu tinggi, maka analisis detail tentang sistem tidak akan terlihat, tetapi jika level terlalu rendah maka akan terlalu rumit dan sulit dipahami. Dengan menentukan level DFD yang tepat, maka dapat memudahkan analis sistem untuk memahami aliran data dalam sistem dan merancang sistem informasi yang efektif dan efisien.

Perbedaan antara level DFD terletak pada tingkat detail informasi yang disajikan. Semakin tinggi level DFD, semakin rinci informasi yang diberikan tentang aliran data dalam sistem. Setiap level DFD dapat digunakan untuk tujuan yang berbeda. Level 0 DFD dapat digunakan untuk memberikan gambaran umum tentang sistem, sedangkan level-level yang lebih rendah dapat digunakan untuk menggambarkan detail yang lebih spesifik tentang sistem. Oleh karena itu, dalam membuat DFD, penting untuk menentukan level yang sesuai dengan kebutuhan analisis.

Untuk memberikan contoh penggunaan level DFD dalam analisis sistem, kita akan mengambil contoh sebuah sistem pembayaran online untuk toko retail.

1. **Level 0 DFD**

   Pada level 0 DFD, sistem pembayaran online direpresentasikan sebagai sebuah kotak tunggal yang menerima input dan menghasilkan output. Input yang diterima adalah data transaksi pembayaran dari pelanggan, sedangkan output yang dihasilkan

adalah proses pembayaran yang berhasil atau tidak berhasil. Pada level ini, tidak ada proses detail yang dijelaskan.

2. **Level 1 DFD**

   Pada level 1 DFD, sistem pembayaran online dibagi menjadi beberapa proses utama dan input/output yang berbeda. Setiap proses utama dijelaskan secara terpisah dan terdapat deskripsi tentang input dan output yang dihasilkan oleh setiap proses tersebut. Contoh proses pada level 1 DFD dapat meliputi proses verifikasi data pelanggan, proses pemrosesan pembayaran, dan proses pengiriman hasil pembayaran.

3. **Level 2 DFD**

   Pada level 2 DFD, setiap proses pada level 1 dibagi lagi menjadi beberapa sub-proses yang lebih kecil dan detail. Contoh sub-proses pada level 2 DFD dapat meliputi proses validasi nomor kartu kredit, proses pengecekan saldo pelanggan, dan proses pencatatan transaksi ke database.

4. **Level selanjutnya**

   Jika diperlukan, dapat dibuat level-level DFD yang lebih detail lagi. Contoh level-level selanjutnya dapat meliputi proses pengecekan keaslian kartu kredit, proses pencatatan detail barang yang dibeli, atau proses integrasi dengan sistem inventaris.

Dalam analisis sistem, level-level DFD digunakan untuk memberikan gambaran yang lebih detail tentang aliran data dalam sistem. Dengan DFD, dapat diidentifikasi proses-proses utama, input/output yang dihasilkan, serta hubungan antar proses dan entitas dalam sistem. Hal ini membantu dalam merancang sistem yang lebih efektif, memperbaiki kelemahan dalam sistem yang sudah ada, serta membuat dokumen dokumentasi sistem yang lebih terperinci.

Prosedur Pembuatan DFD

Berikut adalah prosedur umum untuk membuat DFD:

1. **Identifikasi tujuan dari DFD**

   Langkah pertama dalam membuat DFD adalah mengidentifikasi tujuan dari DFD yang akan dibuat. Tujuan dapat berupa pemodelan sistem yang ada atau perancangan sistem baru.

2. **Identifikasi proses utama dalam sistem**

   Proses utama dalam sistem harus diidentifikasi dan dicatat. Proses-proses ini harus dianalisis secara rinci dan dicatat dalam notasi proses yang sesuai dalam DFD.

3. **Identifikasi aliran data dalam sistem**

   Aliran data yang diperlukan dalam sistem harus diidentifikasi. Aliran data ini harus diberi label yang jelas dan harus dicatat dalam notasi aliran data yang sesuai dalam DFD.

4. **Identifikasi sumber data dan tujuan data**

   Sumber data dan tujuan data harus diidentifikasi dalam sistem. Sumber data adalah entitas yang menghasilkan data, seperti pelanggan atau sistem lain, sementara tujuan data adalah entitas yang menerima data, seperti sistem atau pelanggan lain.

5. **Identifikasi entitas terkait**

   Entitas yang terkait dengan sistem, seperti pelanggan atau pemasok, harus diidentifikasi dan dicatat dalam DFD.

6. **Identifikasi batasan sistem**

   Batasan sistem harus diidentifikasi dan dicatat dalam DFD. Batasan sistem dapat berupa input, output, atau lingkungan eksternal.

7. **Gambar DFD**

   Setelah identifikasi lengkap dilakukan, gambarlah DFD menggunakan notasi yang benar untuk setiap elemen dalam diagram.

8. **Verifikasi DFD**

   Periksa DFD yang dibuat untuk memastikan bahwa tidak ada informasi yang hilang atau salah dicatat. Jika diperlukan, konsultasikan dengan tim atau ahli lain untuk memastikan bahwa DFD sesuai dengan kebutuhan dan spesifikasi sistem.

9. **Update DFD jika diperlukan**

   Jika terjadi perubahan pada sistem atau spesifikasi, perlu untuk memperbarui DFD untuk memastikan bahwa model tetap akurat.

Ada beberapa hal yang perlu diperhatikan saat membuat DFD, antara lain:

1. **Identifikasi tujuan DFD dengan jelas**

   Tujuan DFD harus diidentifikasi dengan jelas dan spesifik. Ini akan membantu memastikan bahwa DFD yang dibuat dapat memenuhi kebutuhan pengguna dan tujuan sistem.

2. **Gunakan notasi yang benar**

   Notasi yang benar harus digunakan dalam DFD agar diagram mudah dipahami oleh semua pihak yang terlibat dalam proyek. Notasi ini termasuk notasi proses, notasi aliran data, dan notasi entitas.

3. **Pastikan setiap elemen terdokumentasi dengan baik**

   Setiap elemen dalam DFD harus terdokumentasi dengan baik untuk memastikan bahwa tidak ada informasi yang hilang atau salah dicatat. Ini termasuk label yang jelas untuk setiap elemen dan deskripsi singkat yang menggambarkan peran masing-masing elemen dalam sistem.

4. **Identifikasi tingkat DFD yang tepat**

   Tingkat DFD yang tepat harus diidentifikasi dan digunakan dalam DFD. Tingkat ini harus mencerminkan tingkat rincian yang diperlukan untuk memenuhi tujuan DFD dan memastikan bahwa diagram tidak menjadi terlalu kompleks.

5. **Pertimbangkan keterkaitan antara elemen dalam sistem**

   Keterkaitan antara elemen dalam sistem harus dipertimbangkan dan dicatat dalam DFD. Ini akan membantu memastikan bahwa diagram mencerminkan hubungan yang sebenarnya antara elemen dalam sistem.

6. **Jelaskan batasan sistem dengan jelas**

   Batasan sistem harus dijelaskan dengan jelas dalam DFD untuk memastikan bahwa tidak ada asumsi yang salah tentang apa yang masuk ke dalam sistem dan apa yang keluar dari sistem.

7. **Pertimbangkan pandangan pengguna pada DFD**

   Pandangan pengguna harus dipertimbangkan saat membuat DFD. Ini akan membantu memastikan bahwa DFD yang dibuat mudah dipahami dan digunakan oleh pengguna.

8. **Berikan nomor pada setiap proses dan aliran data**

   Setiap proses dan aliran data harus diberi nomor untuk memudahkan penggunaan dan identifikasi dalam proses perancangan sistem.

9. **Periksa kembali dan verifikasi DFD yang dibuat**

   DFD yang dibuat harus diperiksa kembali dan diverifikasi untuk memastikan bahwa tidak ada kesalahan atau ketidakakuratan yang terjadi. Jika diperlukan, DFD harus diperbarui untuk memenuhi persyaratan sistem.

Dalam membuat DFD, penting untuk memperhatikan semua faktor yang telah disebutkan agar DFD yang dibuat dapat menggambarkan sistem dengan tepat dan memenuhi kebutuhan pengguna.

Keuntungan DFD

Terdapat beberapa manfaat yang dapat diperoleh dari penggunaan DFD dalam analisis sistem. Pertama, DFD dapat membantu memperjelas pemahaman tentang sistem secara keseluruhan. Ini karena DFD memungkinkan pengguna untuk melihat hubungan dan interaksi antara elemen yang berbeda dalam sistem, sehingga dapat menghindari kebingungan atau kesalahpahaman dalam pemahaman tentang sistem.

Kedua, DFD juga dapat menggambarkan aliran data secara visual sehingga mudah dimengerti. Dengan begitu, pengguna dapat melihat bagaimana data bergerak melalui sistem, dari mana data berasal, dan kemana data akan digunakan. Dalam hal ini, DFD dapat membantu pengguna untuk lebih memahami bagaimana sistem bekerja.

Ketiga, DFD juga membantu dalam mengidentifikasi masalah dalam sistem. Hal ini karena DFD memperlihatkan bagaimana elemen-elemen dalam sistem saling terkait dan berinteraksi. Sehingga, pengguna dapat dengan mudah mengidentifikasi daerah di mana perbaikan atau peningkatan diperlukan untuk meningkatkan efisiensi dan efektivitas sistem.

Keempat, DFD juga dapat membantu dalam perancangan sistem yang lebih baik. Hal ini karena DFD memungkinkan pengguna untuk melihat bagaimana elemen-elemen dalam sistem bekerja bersama. Dalam hal ini, DFD dapat membantu dalam mengidentifikasi area di mana sistem dapat ditingkatkan dan membantu dalam mengembangkan solusi yang lebih efektif.

Kelima, DFD juga dapat mempercepat proses pengembangan sistem. Hal ini karena DFD memungkinkan pengguna untuk melihat bagaimana elemen-elemen dalam sistem terkait satu sama lain dan memperlihatkan proses yang terjadi di dalam sistem. Dengan begitu, DFD dapat membantu pengguna untuk mengembangkan solusi yang lebih efektif dan mengurangi waktu yang dibutuhkan untuk mengembangkan sistem.

Dalam analisis sistem, DFD adalah alat yang sangat berguna untuk membantu memahami bagaimana sistem bekerja, mengidentifikasi masalah dalam sistem, dan mengembangkan solusi yang lebih baik dan lebih efektif. DFD juga dapat membantu mempercepat proses pengembangan sistem dan meningkatkan efisiensi dan efektivitas sistem secara keseluruhan.

DFD atau Data Flow Diagram menjadi penting dalam pengembangan sistem karena memungkinkan pengguna untuk memahami sistem secara keseluruhan dan mengidentifikasi bagian-bagian sistem yang mungkin perlu diperbaiki atau ditingkatkan. Selain itu, DFD juga dapat membantu dalam merancang sistem yang lebih efektif dan efisien.

Beberapa alasan mengapa DFD menjadi penting dalam pengembangan sistem adalah sebagai berikut:

1. Memperjelas Pemahaman tentang Sistem: DFD dapat membantu pengguna untuk memperjelas pemahaman tentang sistem secara keseluruhan. Dengan DFD, pengguna dapat melihat hubungan dan interaksi antara elemen yang berbeda dalam sistem, sehingga dapat menghindari kebingungan atau kesalahpahaman dalam pemahaman tentang sistem.
2. Mengidentifikasi Masalah dalam Sistem: DFD juga dapat membantu dalam mengidentifikasi masalah dalam sistem. Dengan DFD, pengguna dapat melihat bagaimana elemen-elemen dalam sistem saling terkait dan berinteraksi. Sehingga, pengguna dapat dengan mudah mengidentifikasi daerah di mana perbaikan atau peningkatan diperlukan untuk meningkatkan efisiensi dan efektivitas sistem.
3. Merancang Sistem yang Lebih Baik: DFD juga dapat membantu dalam merancang sistem yang lebih baik. DFD memungkinkan pengguna untuk melihat bagaimana elemen-elemen dalam sistem bekerja bersama. Dalam hal ini, DFD dapat membantu dalam mengidentifikasi area di mana sistem dapat ditingkatkan dan membantu dalam mengembangkan solusi yang lebih efektif.

4. Menghemat Waktu dan Biaya: DFD juga dapat membantu dalam menghemat waktu dan biaya dalam pengembangan sistem. DFD memperlihatkan alur kerja sistem dan interaksi antara elemen-elemen dalam sistem. Dengan memahami alur kerja dan interaksi ini, pengguna dapat mengembangkan solusi yang lebih efektif dan mengurangi waktu dan biaya yang dibutuhkan untuk mengembangkan sistem.
5. Mengurangi Risiko: DFD juga dapat membantu dalam mengurangi risiko dalam pengembangan sistem. Dengan DFD, pengguna dapat mengidentifikasi daerah-daerah di mana risiko kemungkinan terjadi dan mengembangkan solusi untuk mengurangi risiko tersebut.

Dengan demikian, DFD menjadi penting dalam pengembangan sistem karena membantu pengguna untuk memahami sistem secara keseluruhan, mengidentifikasi masalah dalam sistem, merancang sistem yang lebih baik, menghemat waktu dan biaya, serta mengurangi risiko dalam pengembangan sistem.

Keterbatasan DFD

Meskipun DFD sangat bermanfaat dalam analisis sistem, namun terdapat beberapa keterbatasan dalam penggunaannya, di antaranya:

1. Tidak menunjukkan aspek temporal: DFD tidak menunjukkan urutan waktu atau jadwal dalam sistem, sehingga tidak dapat digunakan untuk menganalisis aspek temporal atau waktu dalam sistem.
2. Tidak menyediakan detail implementasi: DFD tidak menyediakan detail implementasi teknis dalam sistem, melainkan hanya memberikan gambaran umum tentang aliran data dan proses dalam sistem. Oleh karena itu, DFD tidak dapat digunakan sebagai panduan rinci dalam pengembangan sistem.
3. Memerlukan keterampilan analisis yang cukup: Pembuatan DFD memerlukan keterampilan analisis yang cukup untuk

mengidentifikasi proses, aliran data, dan entitas dalam sistem dengan tepat. Jika analisis tidak dilakukan dengan benar, maka DFD yang dihasilkan tidak akan akurat dan dapat mengakibatkan kesalahan dalam pengembangan sistem.

4. Tidak memperhitungkan faktor manusia: DFD tidak memperhitungkan faktor manusia dalam sistem, seperti keterampilan dan pengetahuan pengguna sistem. Oleh karena itu, pengembang harus mempertimbangkan faktor manusia secara terpisah.
5. Tidak dapat digunakan untuk sistem yang kompleks: DFD tidak dapat digunakan untuk menganalisis sistem yang sangat kompleks, seperti sistem yang memiliki banyak interaksi dan proses yang terintegrasi secara kompleks.

Dalam penggunaannya, penting bagi pengembang sistem untuk mempertimbangkan keterbatasan-keterbatasan tersebut agar DFD dapat digunakan dengan efektif dan efisien dalam analisis sistem.

Untuk mengatasi keterbatasan-keterbatasan dalam penggunaan DFD, pengembang sistem dapat menggunakan beberapa cara, seperti menggunakan metode analisis lain yang memperhitungkan faktor temporal dan manusia dalam sistem, menggunakan notasi yang lebih detail, melakukan kajian kebutuhan sistem secara rinci, menggabungkan DFD dengan model lain, dan menggunakan software bantu. Penting untuk memilih metode yang sesuai dan melakukan kajian kebutuhan sistem secara menyeluruh untuk menghasilkan diagram DFD yang akurat dan efektif dalam menganalisis sistem.

Penggunaan metode analisis lain dapat membantu memperhitungkan faktor-faktor seperti penggunaan waktu, interaksi manusia dengan sistem, dan kompleksitas proses dalam sistem. Metode-metode tersebut dapat mencakup pemodelan proses bisnis (business process modeling), pemodelan aliran kerja (workflow modeling), atau pemodelan data (data modeling). Dengan

menggunakan metode-metode ini, pengembang sistem dapat menghasilkan model sistem yang lebih rinci dan akurat.

Selain itu, pengembang sistem juga dapat menggunakan notasi yang lebih detail dalam pembuatan DFD, seperti menggunakan notasi Gane-Sarson atau Yourdon-Coad yang memiliki simbol-simbol yang lebih banyak. Hal ini dapat membantu menjelaskan proses dan interaksi dalam sistem dengan lebih detail.

Melakukan kajian kebutuhan sistem secara rinci juga sangat penting dalam mengatasi keterbatasan DFD. Kajian ini dapat membantu mengidentifikasi proses dan interaksi dalam sistem secara lebih detail, sehingga diagram DFD yang dihasilkan lebih akurat dan sesuai dengan kebutuhan sistem.

Pengembang sistem juga dapat menggabungkan DFD dengan model lain, seperti diagram use case atau diagram kelas dalam UML, untuk mendapatkan gambaran yang lebih lengkap tentang sistem. Dengan menggabungkan model-model tersebut, pengembang sistem dapat memperoleh pemahaman yang lebih komprehensif tentang sistem dan memastikan bahwa DFD yang dihasilkan mencerminkan sistem secara keseluruhan.

Terakhir, penggunaan software bantu dapat membantu pengembang sistem dalam pembuatan dan analisis DFD. Software-software tersebut dapat membantu mempercepat proses pembuatan DFD dan memudahkan pengembang sistem dalam melakukan analisis terhadap diagram DFD yang sudah dibuat.

Dalam mengatasi keterbatasan-keterbatasan tersebut, perlu diingat bahwa DFD bukanlah satu-satunya metode yang dapat digunakan dalam analisis sistem. Oleh karena itu, pengembang sistem perlu mempertimbangkan faktor-faktor seperti kebutuhan sistem, keahlian dan pengalaman tim pengembang, dan sumber daya yang tersedia sebelum memilih metode analisis yang tepat untuk sistem yang sedang dikembangkan.

Evaluasi / Soal Latihan

Buatlah DFD mengenai sistem informasi perpustakaan, dimulai dari level 0 sampai dengan level 2!

BAB 6

UNIFIED MODELING LANGUAGE (UML)

Tujuan Pembelajaran

Mahasiswa mampu memahami konsep permodelan UML termasuk representasi visual dari elemen-elemen yang terlibat dalam sistem perangkat lunak atau proyek pengembangan serta mengajarkan konsep-konsep sehingga mahasiswa dapat memahami bagaimana merancang, mengembangkan, dan memelihara sistem perangkat lunak dengan baik.

Pengertian UML

Unified Modeling Language (UML) adalah sebuah bahasa pemodelan grafis yang digunakan untuk merancang dan mendokumentasikan sistem perangkat lunak. UML sering digunakan oleh para pengembang perangkat lunak untuk memvisualisasikan, mendokumentasikan, dan memahami rancangan sistem perangkat lunak yang akan dibangun.

UML pertama kali diperkenalkan pada tahun 1997 oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson, yang kemudian dikenal sebagai «The Three Amigos». Sebelumnya, Booch, Rumbaugh, dan Jacobson telah mengembangkan bahasa pemodelan perangkat lunak

masing-masing, yaitu Booch OOD, OMT, dan OOSE. Pada tahun 1994, mereka bergabung untuk menggabungkan kekuatan bahasa-bahasa mereka dan mengembangkan sebuah bahasa pemodelan perangkat lunak yang lebih kuat.

Pada awalnya, UML hanya terdiri dari beberapa jenis diagram, yaitu diagram kelas, diagram objek, diagram urutan, diagram aktivitas, dan diagram statechart. Namun, UML terus berkembang dan pada versi-versi berikutnya, lebih banyak jenis diagram ditambahkan, seperti diagram komponen, diagram penempatan, diagram paket, dan lain-lain.

UML kemudian menjadi sangat populer dan digunakan secara luas di seluruh dunia. Pada tahun 1997, Object Management Group (OMG), sebuah organisasi internasional yang terdiri dari perusahaan-perusahaan teknologi terkemuka, mengadopsi UML sebagai standar bahasa pemodelan perangkat lunak mereka. OMG terus mengembangkan UML dan pada tahun 2005, UML menjadi standar ISO.

Sejak saat itu, UML terus digunakan oleh para pengembang perangkat lunak di seluruh dunia dan telah menjadi alat yang sangat penting dalam pengembangan perangkat lunak. Dengan UML, pengembang perangkat lunak dapat dengan mudah merancang dan mendokumentasikan rancangan sistem perangkat lunak mereka, serta berkomunikasi dengan stakeholder lainnya.

Tujuan utama dari penggunaan UML adalah untuk membantu pengembang perangkat lunak dalam memvisualisasikan, mendokumentasikan, dan memahami rancangan sistem perangkat lunak yang akan dibangun. Selain itu, UML juga dapat membantu dalam komunikasi antara pengembang perangkat lunak dan stakeholder yang terlibat dalam proyek perangkat lunak, sehingga dapat memperbaiki kualitas dan efisiensi pengembangan perangkat lunak.

Unified Modeling Language (UML) memiliki beberapa konsep dasar yang membentuk landasan dasar dalam pengembangan model

dan dokumentasi perangkat lunak. Beberapa konsep dasar UML yang paling penting antara lain sebagai berikut:

1. Objek: Objek adalah entitas dalam perangkat lunak yang memiliki sifat dan perilaku tertentu. Objek digambarkan sebagai sebuah persegi panjang dengan nama objek di dalamnya.
2. Kelas: Kelas adalah sebuah blueprint atau cetak biru dari objek yang digunakan untuk membuat objek-objek serupa. Kelas dapat memiliki atribut atau variabel, metode atau perilaku, dan hubungan dengan kelas lain dalam bentuk asosiasi, agregasi, atau komposisi. Kelas digambarkan sebagai sebuah persegi panjang dengan nama kelas di dalamnya.
3. Atribut: Atribut adalah variabel atau properti dari objek atau kelas yang dapat digunakan untuk menyimpan data. Atribut digambarkan sebagai sebuah persegi kecil di dalam persegi panjang yang mewakili objek atau kelas.
4. Metode: Metode adalah perilaku atau fungsi yang dapat dilakukan oleh objek atau kelas untuk memanipulasi data atau menjalankan tugas tertentu. Metode digambarkan sebagai sebuah lingkaran di dalam persegi panjang yang mewakili objek atau kelas.
5. Asosiasi: Asosiasi adalah hubungan antara dua kelas yang menunjukkan bahwa objek dari kelas satu dapat terkait dengan objek dari kelas lain. Asosiasi digambarkan sebagai sebuah garis yang menghubungkan dua persegi panjang kelas dengan nama asosiasi di atas garis.
6. Agregasi: Agregasi adalah hubungan antara dua kelas yang menunjukkan bahwa satu kelas memiliki kelas lain sebagai bagian darinya. Agregasi digambarkan sebagai sebuah garis dengan sebuah diamond yang menghubungkan dua persegi panjang kelas dengan nama agregasi di atas garis.
7. Komposisi: Komposisi adalah hubungan antara dua kelas yang menunjukkan bahwa satu kelas adalah bagian penting dari kelas lainnya dan tidak dapat hidup tanpa kelas lainnya. Komposisi digambarkan sebagai sebuah garis dengan sebuah diamond yang

memiliki garis penuh di dalamnya yang menghubungkan dua persegi panjang kelas dengan nama komposisi di atas garis.

8. Inheritance: Inheritance atau pewarisan adalah konsep yang memungkinkan kelas untuk mewarisi sifat dan perilaku dari kelas lainnya. Kelas yang mewarisi disebut kelas anak atau subclass, sedangkan kelas yang memberikan warisan disebut kelas induk atau superclass. Inheritance digambarkan sebagai sebuah panah dengan ujung segitiga kecil yang mengarah dari kelas anak ke kelas induk.

Keuntungan Menggunakan UML

Berikut adalah beberapa keuntungan menggunakan UML dalam pengembangan perangkat lunak:

1. Komunikasi yang lebih baik antara tim pengembang: UML adalah bahasa model visual standar yang memungkinkan tim pengembang berkomunikasi dan berbagi informasi dengan lebih jelas dan efektif. Hal ini meminimalkan kesalahan pemahaman dan meningkatkan kualitas komunikasi antara anggota tim.
2. Analisis dan desain yang lebih baik: UML memungkinkan analisis dan desain yang lebih baik karena model UML memperlihatkan struktur perangkat lunak secara visual, sehingga lebih mudah dipahami dan dianalisis.
3. Pemeliharaan dan pengembangan yang lebih mudah: UML membantu mengurangi kesalahan dalam proses pengembangan dan membuat pemeliharaan dan pengembangan lebih mudah karena model UML bisa dianalisis dan dimodifikasi dengan mudah.
4. Dokumentasi yang lebih baik: UML dapat digunakan untuk membuat dokumentasi perangkat lunak yang jelas dan mudah dipahami. Hal ini memudahkan pengembang untuk mengkomunikasikan desain dan implementasi sistem kepada pengguna dan pemangku kepentingan lainnya.

5. Reusabilitas yang lebih baik: Dalam UML, model komponen dapat digunakan kembali pada proyek-proyek lain dan diintegrasikan dengan mudah dengan sistem yang sudah ada.
6. Pengujian yang lebih baik: UML dapat membantu meningkatkan efisiensi pengujian dan meminimalkan kesalahan pada saat pengujian. Model UML dapat digunakan untuk mengidentifikasi kemungkinan masalah sebelum implementasi, sehingga pengujian dapat dilakukan dengan lebih efektif dan efisien.

Diagram Use Case

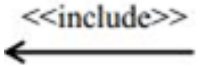
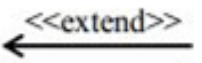
Diagram use case adalah salah satu jenis diagram yang digunakan dalam UML (Unified Modeling Language) untuk merepresentasikan interaksi antara sistem dan pengguna. Tujuan dari diagram use case adalah untuk menggambarkan fitur-fitur utama yang diperlukan oleh pengguna dalam menggunakan sistem.

Secara umum, ada tiga langkah yang harus dilakukan dalam membuat diagram use case menggunakan UML:

1. Identifikasi aktor dan use case: Langkah pertama adalah mengidentifikasi aktor dan use case dalam sistem. Aktor adalah pengguna atau sistem lain yang berinteraksi dengan sistem, sedangkan use case adalah aksi atau fungsi yang dilakukan oleh sistem untuk memenuhi kebutuhan pengguna atau sistem lain. Identifikasi ini biasanya dilakukan melalui wawancara dengan pengguna atau pemangku kepentingan lain.
2. Menggambarkan hubungan antara aktor dan use case: Setelah aktor dan use case diidentifikasi, langkah berikutnya adalah menggambarkan hubungan antara keduanya. Hubungan ini dapat diwakili oleh panah yang menghubungkan aktor dengan use case yang sesuai.
3. Menambahkan deskripsi tambahan: Akhirnya, deskripsi tambahan dapat ditambahkan ke diagram use case, seperti

deskripsi singkat dari setiap use case, atau deskripsi singkat dari aliran kasus penggunaan yang mungkin terjadi.

Dalam membuat diagram use case, ada beberapa elemen penting yang harus diperhatikan, yaitu:

Simbol	Elemen	Keterangan
	Aktor	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan dengan use case.
	Use Case	Abstraksi dan interaksi antara sistem dan aktor.
	Assocation	Abstraksi dari penghubung antara aktor dengan use case.
	Generalisasi	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan use case.
	Include	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya.
	Extend	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi.

Dengan menggunakan diagram use case, pengembang sistem dapat dengan mudah memahami kebutuhan pengguna dan memastikan bahwa fitur-fitur yang dibangun dalam sistem memenuhi kebutuhan pengguna dengan baik. Selain itu, diagram use case juga membantu pengembang sistem untuk berkomunikasi dengan pemangku kepentingan dan tim pengembang lainnya dengan lebih efektif. Contoh diagram use case.

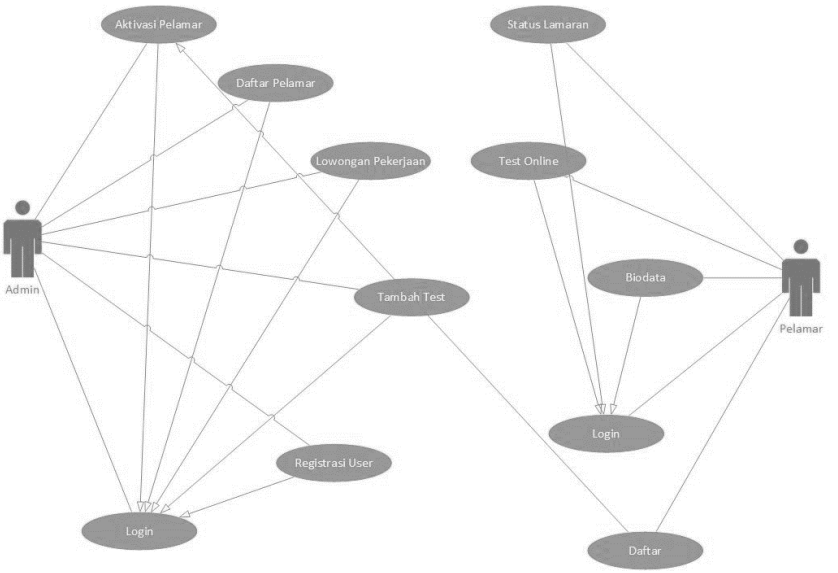


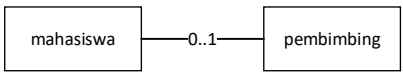
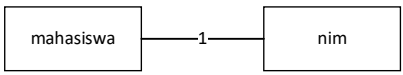
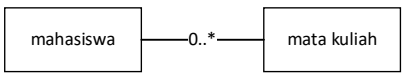
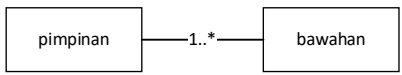
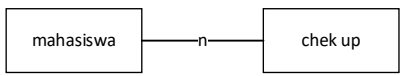
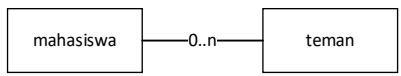
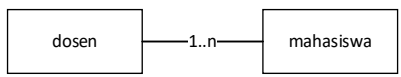
Diagram Class

Diagram class merupakan salah satu jenis diagram yang digunakan dalam UML untuk merepresentasikan struktur kelas dan hubungan antar kelas dalam suatu sistem. Tujuan dari diagram class adalah untuk memberikan gambaran mengenai objek-objek yang ada dalam suatu sistem dan hubungan antara objek-objek tersebut. Diagram class terdiri dari beberapa elemen, yaitu:

Simbol	Elemen	Keterangan
	Garis lurus (Generalization)	Menunjukan hubungan objek anak (descendent) dan induk (ancestor) dalam hal berbagai perilaku dan struktur datanya.
	Nary Association	Suatu upaya untuk menghindari asosiasi yang melebihi 2 objek.
	Class	Suatu himpunan dari objek-objek dalam sistem, yang kemudian berbagi atribut dan operasi yang persis sama.

	Collaboration	Berupa urutan aksi-aksi dalam sistem agar menghasilkan sebuah hasil yang terukur.
	Realization	Sebuah operasi yang benar-benar dilakukan oleh objek dalam sistem.
	Dependency	Suatu hubungan pada perubahan yang terjadi dalam independent yang mempengaruhi elemen yang tidak mandiri.
	Association	Bagian yang menghubungkan objek yang satu dengan yang lainnya.

Sama halnya dengan ERD, class diagram memiliki kardinalitas, berikut kardinalitas dari class diagram.

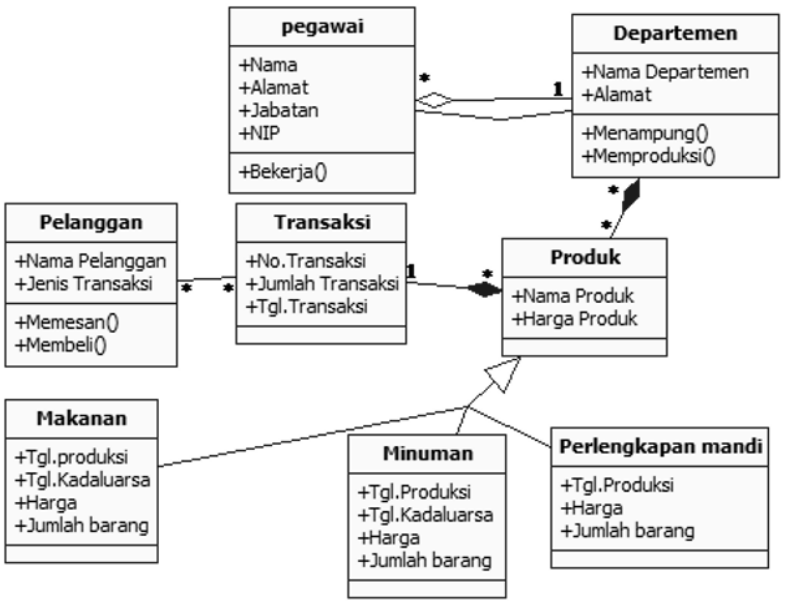
Nilai Kardinalitas	Arti	Contoh
0..1	Nol atau satu	
1	Hanya satu	
0..*	Nol atau lebih	
1..*	Satu atau lebih	
n	Hanya n (dengan n>1)	
0..n	Nol sampai n (dengan n>1)	
1..n	Satu sampai n (dengan n>1)	

Cara membuat diagram class menggunakan UML adalah sebagai berikut:

1. Identifikasi kelas-kelas yang akan dimodelkan dalam sistem.
2. Tentukan atribut dan metode dari setiap kelas.
3. Gambarkan kelas-kelas tersebut dalam diagram class.
4. Tentukan relasi antar kelas dan gambarkan dalam diagram class. Beberapa jenis relasi antar kelas yang dapat digunakan adalah inheritance, aggregation, dan association.
5. Berikan label pada setiap kelas, atribut, metode, dan relasi yang digambarkan dalam diagram class.
6. Berikan penjelasan singkat mengenai diagram class yang telah dibuat.
7. Revisi dan perbaiki diagram class jika diperlukan.

Dalam pembuatan diagram class, sebaiknya dilakukan secara terstruktur dan sistematis agar dapat memberikan gambaran yang jelas dan mudah dipahami mengenai struktur kelas dan hubungan antar kelas dalam suatu sistem.

Contoh diagram class penjualan.



Untuk keterangan diagram class di atas, yaitu:

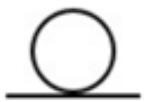
1. Class atau table departemen mempunya agresi dengan class atau table pegawai karena departemen ini dapat berdiri sendiri. Kemudian banyak pegawai dapat bekerja dalam satu departemen, jadi many to 1.
2. Class atau table transaksi tidak dapat berdiri sendiri, sebab ia harus ada table produk. Hal ini berlaku terhadap table produk, sebab membutuhkan table departemen.
3. Banyak pelanggan yang bisa melakukan banyak transaksi.
4. Satu transaksi bisa mencakup banyak produk.

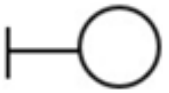
Diagram Sequence

Diagram Sequence atau diagram urutan adalah jenis diagram UML yang menggambarkan interaksi antara objek-objek dalam suatu sistem secara berurutan. Diagram ini biasanya digunakan untuk menggambarkan aliran logika dari sebuah fitur atau fungsi pada suatu sistem.

Tujuan dari diagram sequence adalah untuk menggambarkan urutan pesan atau pemanggilan objek antara objek-objek dalam suatu sistem dan memperlihatkan bagaimana objek-objek tersebut berinteraksi dalam waktu yang berbeda. Dalam diagram ini, urutan pesan ditampilkan sebagai garis panah yang menghubungkan objek-objek dan diurutkan sesuai dengan waktu.

Diagram sequence terdiri dari beberapa elemen, yaitu:

Simbol	Elemen	Keterangan
	Actor	Menggambar orang yang sedang berinteraksi dengan sistem.
	Entity Class	Menggambarkan hubungan yang akan dilakukan.

	Boundary Class	Menangani komunikasi antar lingkungan sistem.
	Control Class	Menggambarkan penghubung antar boundary dengan tabel
	A Focus of Control & A Life Line	Menggambarkan tempat mulai dan berakhirnya message.
	A Message	Menggambarkan pengiriman pesan.

Untuk membuat diagram sequence menggunakan UML, langkah-langkah yang perlu dilakukan antara lain:

1. Identifikasi objek-objek yang terlibat dalam suatu interaksi atau fungsi sistem.
2. Tentukan urutan atau langkah-langkah yang terjadi dalam interaksi atau fungsi tersebut.
3. Buat garis vertikal yang merepresentasikan setiap objek dan letakkan garis tersebut sejajar satu sama lain.
4. Tuliskan pesan atau pemanggilan method pada garis panah yang menghubungkan dua objek.
5. Susun urutan garis panah sesuai dengan urutan waktu.
6. Tambahkan keterangan atau catatan pada diagram sequence jika diperlukan.

Untuk membatasi hak akses seorang pengguna, sebuah program biasanya dilengkapi dengan fitur login. Login dapat menjaga privasi user dan membuat program menjadi lebih aman. User akan mendapat izin akses masuk pada sebuah program bila memasukkan username dan password yang sesuai. Berikut adalah contoh Sequence Diagram proses login:

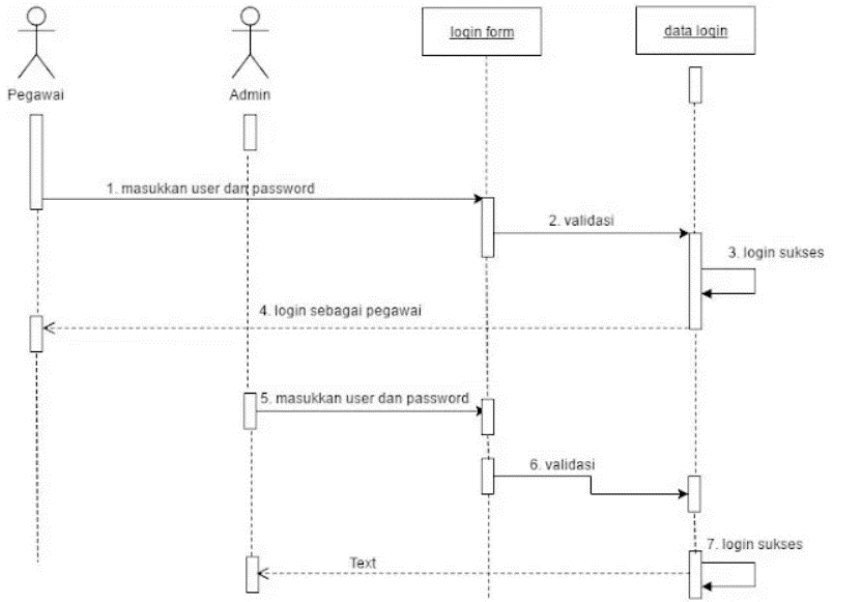


Diagram State

Diagram state adalah salah satu jenis diagram UML yang digunakan untuk menggambarkan perubahan keadaan suatu objek dalam suatu sistem. Diagram state menggambarkan serangkaian keadaan yang mungkin terjadi pada suatu objek dan transisi antara keadaan tersebut.

Tujuan dari diagram state adalah untuk memodelkan keadaan internal dari suatu objek dalam suatu sistem dan bagaimana objek tersebut berperilaku dalam setiap keadaan. Dengan diagram state, pengembang sistem dapat memvisualisasikan berbagai keadaan yang mungkin terjadi pada objek dan melihat bagaimana objek tersebut bereaksi terhadap transisi antar keadaan.

Untuk membuat diagram state menggunakan UML, terdapat beberapa langkah yang harus dilakukan, yaitu:

1. Identifikasi objek yang akan dimodelkan. Langkah pertama dalam membuat diagram state adalah mengidentifikasi objek atau kelas yang akan dimodelkan dalam diagram.

2. Tentukan keadaan-keadaan yang mungkin terjadi pada objek. Setelah mengidentifikasi objek yang akan dimodelkan, langkah selanjutnya adalah menentukan berbagai keadaan yang mungkin terjadi pada objek tersebut.
3. Tentukan transisi antara keadaan-keadaan. Setelah menentukan keadaan-keadaan yang mungkin terjadi pada objek, langkah selanjutnya adalah menentukan transisi yang mungkin terjadi antara keadaan-keadaan tersebut.
4. Gambarkan diagram state. Setelah menentukan keadaan-keadaan dan transisi antara keadaan-keadaan, langkah terakhir adalah menggambarkan diagram state menggunakan notasi UML yang sesuai.

Notasi yang digunakan dalam diagram state meliputi:

Simbol	Elemen	Keterangan
	State	Menunjukkan keadaan dari suatu objek.
	Initial Pseudo State	Menunjukkan keadaan awal suatu objek.
	Final State	Menunjukkan keadaan akhir suatu objek.
	Transition	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	Node	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan sumber daya komputasi.

Dalam diagram state, objek ditunjukkan sebagai kotak dengan nama objek pada bagian atas kotak. Keadaan-keadaan ditunjukkan sebagai lingkaran yang diletakkan di dalam kotak objek, sedangkan transisi antara keadaan-keadaan ditunjukkan dengan panah yang

menghubungkan lingkaran-lingkaran tersebut. Pada panah transisi dapat ditulis event yang memicu transisi atau action yang dilakukan pada objek ketika terjadi transisi.

Contoh diagram state.

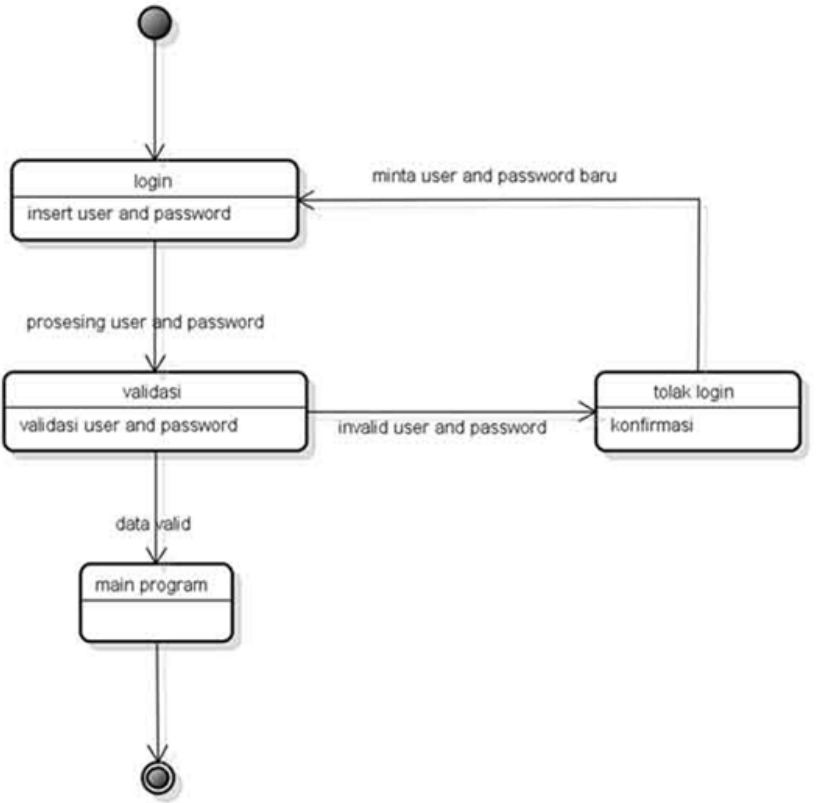
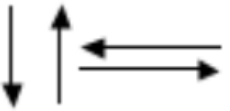


Diagram Activity

Diagram aktivitas (activity diagram) adalah salah satu jenis diagram UML yang digunakan untuk menggambarkan aktivitas atau alur kerja dalam sebuah sistem atau proses bisnis. Tujuan dari diagram aktivitas adalah untuk memvisualisasikan alur kerja atau aktivitas yang terjadi dalam sebuah sistem atau proses bisnis dan mengidentifikasi bagian-bagian yang dapat dioptimalkan untuk meningkatkan efisiensi.

Diagram aktivitas terdiri dari beberapa simbol, antara lain:

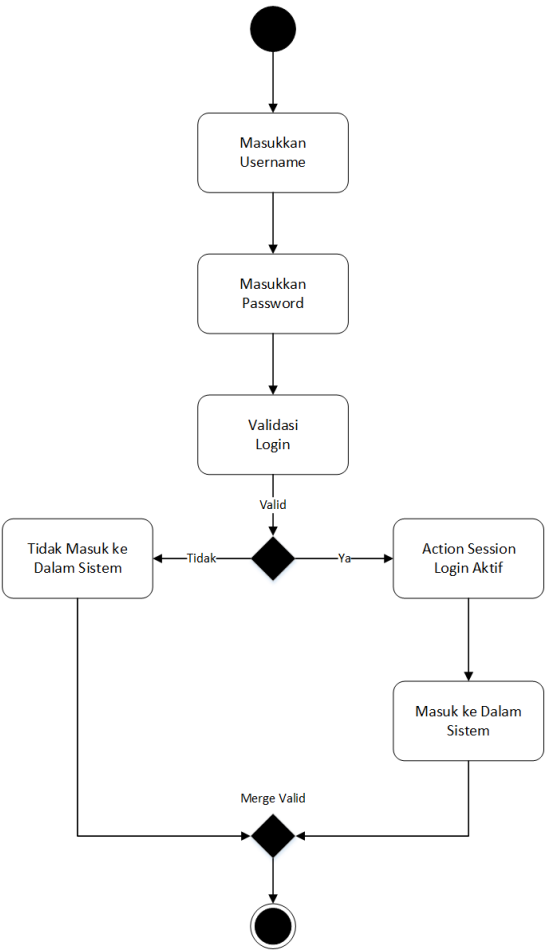
Simbol	Elemen	Keterangan
	Activity	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
	Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	Initial Node	Menunjukkan titik awal dari diagram aktivitas
	Activity Final Node	Menunjukkan titik akhir dari diagram aktivitas
	Decision	Digunakan untun menggambarkan suatu keputusan/tindakan yang harus diambil pada kondisi tertentu.
	Line Connector	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

Cara membuat diagram aktivitas menggunakan UML adalah sebagai berikut:

1. Tentukan aktivitas-aktivitas atau tindakan-tindakan yang akan dimasukkan dalam diagram aktivitas.
2. Buatlah node untuk setiap aktivitas tersebut, dan hubungkan node-node tersebut dengan edge.
3. Identifikasi cabang-cabang dalam alur kerja dengan menggunakan fork node, dan identifikasi juga titik-titik penggabungan dengan menggunakan join node.
4. Tentukan titik awal dan titik akhir dari diagram aktivitas dengan menggunakan initial node dan final node.
5. Beri label pada setiap node untuk menjelaskan aktivitas atau tindakan yang dilakukan.
6. Beri label pada setiap edge untuk menjelaskan urutan dari aktivitas atau tindakan tersebut.

Dengan menggunakan diagram aktivitas, kita dapat lebih mudah memahami alur kerja atau aktivitas dalam sebuah sistem atau proses bisnis, sehingga dapat membantu kita dalam mengidentifikasi bagian-bagian yang dapat dioptimalkan untuk meningkatkan efisiensi.

Contoh diagram aktivitas.



Evaluasi / Soal Latihan

Buatlah diagram use case, diagram class, diagram sequence, diagram state, dan diagram aktivitas dari sistem informasi perpustakaan !

BAB 7

NORMALISASI

Tujuan Pembelajaran

Mahasiswa mampu menghasilkan desain basis data yang baik, efisien, dan tidak redundan. Dengan memahami normalisasi, mahasiswa akan mampu mengoptimalkan struktur basis data, menghindari masalah yang mungkin terjadi pada data, dan meningkatkan kinerja sistem basis data. Selain itu, dengan pemahaman yang baik tentang normalisasi, mahasiswa juga akan mampu menganalisis struktur basis data yang sudah ada dan melakukan perbaikan jika diperlukan.

Pengertian Normalisasi

Normalisasi basis data adalah suatu proses untuk mengorganisir data dalam basis data relasional dengan tujuan menghilangkan redudansi (redundancy) dan menghindari anomali dalam pengolahan data. Normalisasi basis data bertujuan untuk memperbaiki struktur tabel dan hubungan antar tabel agar lebih efisien dan mudah dikelola. Normalisasi basis data dilakukan dengan membagi tabel besar menjadi beberapa tabel yang lebih kecil dan terkait erat sesuai dengan aturan normalisasi (Date, 2004), (Connolly & Begg, 2014).

Tujuan Normalisasi

Tujuan normalisasi basis data adalah untuk menghasilkan desain basis data yang optimal dan efisien. Dengan normalisasi, basis data diorganisir menjadi beberapa tabel, dan setiap tabel memiliki satu set atribut yang terkait dengan suatu entitas atau relasi tertentu. Tujuan normalisasi adalah untuk menghilangkan redudansi data dan meminimalkan ketidak-konsistenan data, sehingga mengurangi kemungkinan terjadinya anomali data seperti update, insert, dan delete.

Normalisasi membantu dalam meningkatkan kinerja basis data dan membuatnya lebih mudah dikelola dan dipelihara. Normalisasi juga membantu dalam mengurangi kebutuhan untuk melakukan pembaruan berulang pada setiap entitas atau relasi dalam basis data, karena hanya satu entitas atau relasi yang perlu diperbarui setiap kali ada perubahan data. Dalam jangka panjang, normalisasi juga membantu dalam mengurangi biaya pemeliharaan dan pengembangan basis data.

Manfaat Normalisasi

Berikut beberapa manfaat normalisasi basis data:

1. Mengurangi Redundansi Data: Normalisasi membantu mengurangi redundansi data dalam basis data, sehingga menghemat ruang penyimpanan dan meningkatkan konsistensi data. Hal ini memungkinkan perubahan data dilakukan dengan mudah dan meminimalkan kesalahan dan inkonsistensi yang mungkin terjadi akibat redundansi data.
2. Meningkatkan Konsistensi Data: Normalisasi membantu memastikan bahwa data yang disimpan dalam basis data adalah konsisten dan dapat diandalkan. Hal ini memungkinkan penggunaan data yang lebih akurat dan dapat diandalkan dalam aplikasi dan sistem.

3. Mempercepat Pencarian Data: Normalisasi membantu meningkatkan kecepatan pencarian data dalam basis data dengan mengurangi jumlah tabel dan kolom yang harus dicari. Hal ini meningkatkan kinerja basis data dan mempercepat waktu akses data.
4. Memfasilitasi Perubahan: Normalisasi membuat basis data lebih fleksibel dan memfasilitasi perubahan pada struktur data tanpa mempengaruhi keseluruhan basis data. Hal ini memungkinkan pengembangan dan pengelolaan basis data yang lebih mudah dan efisien.
5. Menghindari Duplikasi Data: Normalisasi memastikan bahwa data hanya disimpan di satu lokasi, sehingga menghindari duplikasi data dan meningkatkan integritas data. Hal ini juga meminimalkan kemungkinan kesalahan pada data dan memudahkan pemeliharaan basis data.

Kriteria Normalisasi

Kriteria normalisasi basis data yang umum digunakan adalah sebagai berikut:

1. **Setiap atribut dalam tabel hanya memiliki satu nilai**

   Setiap atribut pada tabel harus memiliki nilai tunggal (single-value) atau atomik, artinya tidak boleh memiliki kumpulan nilai atau nilai terpisah yang harus dikelompokkan bersama. Hal ini membantu memastikan bahwa data dapat dikelola dan dimanipulasi dengan mudah dan akurat.

2. **Setiap tabel harus memiliki sebuah primary key**

   Setiap tabel harus memiliki sebuah primary key yang unik dan tidak berubah-ubah. Primary key digunakan untuk mengidentifikasi setiap baris dalam tabel secara unik.

3. **Tidak ada atribut yang tergantung pada atribut non-key**

   Tabel harus disusun sedemikian rupa sehingga tidak ada atribut yang bergantung pada atribut non-key. Hal ini dapat dicapai melalui normalisasi tabel hingga mencapai minimal bentuk normal.

4. **Tidak ada ketergantungan siklik**

   Tidak ada ketergantungan siklik antar tabel. Artinya, tidak ada tabel yang saling tergantung satu sama lain sehingga tidak mungkin menentukan urutan tabel yang tepat saat mengakses data.

Dengan memenuhi kriteria di atas, normalisasi dapat membantu memperbaiki integritas data, meningkatkan efisiensi database, dan mengurangi redundansi data.

Level Normalisasi

Normalisasi dilakukan melalui serangkaian level atau tingkatan, yaitu:

1. **First Normal Form (1NF)**

   Pada level ini, setiap kolom dalam tabel hanya memiliki nilai atomik atau tidak dapat dibagi lagi. Setiap baris dalam tabel harus unik, dan tabel tersebut tidak boleh memiliki kolom-kolom yang dapat berubah-ubah.

2. **Second Normal Form (2NF)**

   Pada level ini, setiap kolom dalam tabel harus bergantung sepenuhnya pada kunci utama tabel. Dalam kata lain, setiap non-kunci kolom harus bergantung pada kunci utama tabel, dan tidak boleh bergantung pada kolom non-kunci lainnya.

3. **Third Normal Form (3NF)**

   Pada level ini, setiap kolom non-kunci dalam tabel harus bergantung pada kunci utama tabel. Selain itu, tidak boleh ada ketergantungan transitive antara kolom-kolom dalam tabel. Ketergantungan transitive terjadi ketika sebuah kolom bergantung pada kolom non-kunci lainnya.

4. **Boyce-Codd Normal Form (BCNF)**

   Pada level ini, setiap ketergantungan fungsional dalam tabel harus bersifat trivial atau trivial. Dalam kata lain, kunci utama tabel harus dapat menentukan semua kolom non-kunci dalam tabel.

5. **Fourth Normal Form (4NF)**

   Pada level ini, tabel harus bebas dari multi-nilai dependensi, yaitu ketergantungan fungsional yang melibatkan beberapa nilai dalam satu kolom.

6. **Fifth Normal Form (5NF) atau Normal Form Domain/ Projection-Join.**

   Pada level ini, setiap ketergantungan fungsional antara kunci utama dan non-kunci harus dijaga di seluruh relasi database. Relasi ini mencakup semua informasi yang relevan dan tidak ada informasi yang redundan atau ambigu.

Contoh Normalisasi

Berikut ini adalah contoh normalisasi database struk penjualan.

kode_faktur	tanggal	kode_barang	nama_barang	harga	qty
KF001	10/03/2023	KB001	Rinso Matic	60000	1
		KB002	Pepsodent	25000	1
		KB003	Tisue Paseo	15000	1
		KB004	Sikat Gigi	10000	1
KF002	12/03/2023	KB005	Minya Bimoli	40000	1
		KB006	Indome Kari	2900	10
KF003	15/03/2023	KB007	Nugget Fiesta	50000	2

Contoh data di atas merupakan data yang belum dinormalisasi, selanjutnya menuju tahap normalisasi 1NF.

1. **1NF**

   Suatu tabel dikatakan 1NF jika dan hanya jika setiap atribut dari data tersebut hanya memiliki nilai tunggal dalam satu baris.

Jadi, tabel yang belum dinormalisasi tadi perlu diubah, sehingga bentuk 1NF menjadi seperti ini:

kode_faktur	tanggal	kode_barang	nama_barang	harga	qty
KF001	10/03/2023	KB001	Rinso Matic	60000	1
KF001	10/03/2023	KB002	Pepsodent	25000	1
KF001	10/03/2023	KB003	Tisue Paseo	15000	1
KF001	10/03/2023	KB004	Sikat Gigi	10000	1
KF002	12/03/2023	KB005	Minya Bimoli	40000	1
KF002	12/03/2023	KB006	Indome Kari	2900	10
KF003	15/03/2023	KB007	Nugget Fiesta	50000	2

Inti dari normalisasi 1NF adalah tidak boleh ada grouping data ataupun duplikasi data. Sekarang lanjut pada tahap normalisasi 2NF.

2. 2NF

Syarat 2NF adalah tidak diperkenankan adanya partial "functional dependency" kepada primary key dalam sebuah tabel. Apa itu "functional dependency"? Functional dependency adalah setiap atribut yang bukan kunci (non key) bergantung secara fungsional terhadap primary key. Intinya adalah pada tahap normalisasi 2NF ini tabel tersebut harus dipecah berdasarkan primary key. Sehingga bentuk normalisasi 2NF dari tabel tersebut adalah sebagai berikut:

- Tabel Barang

kode_barang	nama_barang	harga
KB001	Rinso Matic	60000
KB002	Pepsodent	25000
KB003	Tisue Paseo	15000
KB004	Sikat Gigi	10000
KB005	Minya Bimoli	40000
KB006	Indome Kari	2900
KB007	Nugget Fiesta	50000

- Tabel Transaksi

kode_faktur	tanggal	kode_barang	qty
KF001	10/03/2023	KB001	1

KF001	10/03/2023	KB002	1
KF001	10/03/2023	KB003	1
KF001	10/03/2023	KB004	1
KF002	12/03/2023	KB005	1
KF002	12/03/2023	KB006	10
KF003	15/03/2023	KB007	2

3. 3NF

Pada 3NF tidak diperkenankan adanya partial "transitive dependency" dalam sebuah tabel. Apa itu "transitive dependency"? Transitive dependency biasanya terjadi pada tabel hasil relasi, atau kondisi dimana terdapat tiga atribut A, B, C. Kondisinya adalah A ⇒ B dan B ⇒ C. Maka C dikatakan sebagai transitive dependency terhadap A melalui B.

Intinya pada 3NF ini, jika terdapat suatu atribut yang tidak bergantung pada primary key tapi bergantung pada field yang lain maka atribut-atribut tersebut perlu dipisah ke tabel baru.

Contohnya ada pada atribut qty, kolom tersebut tidak bergantung langsung pada primary key kode_faktur melainkan bergantung pada kolom kode_barang. Jadi setelah dinormalisasi 3NF akan menghasilkan tabel berikut:

- Tabel Barang

kode_barang	nama_barang	harga
KB001	Rinso Matic	60000
KB002	Pepsodent	25000
KB003	Tisue Paseo	15000
KB004	Sikat Gigi	10000
KB005	Minya Bimoli	40000
KB006	Indome Kari	2900
KB007	Nugget Fiesta	50000

- Tabel Transaksi

kode_faktur	tanggal
KF001	10/03/2023
KF002	12/03/2023
KF003	15/03/2023

- Tabel Detail Barang

kode_faktur	kode_barang	qty	harga
KF001	KB001	1	60000
KF001	KB002	1	25000
KF001	KB003	1	15000
KF001	KB004	1	10000
KF002	KB005	1	40000
KF002	KB006	10	2900
KF003	KB007	2	50000

Dari tabel di atas dapat dilihat pada tahap normalisasi 3NF menghasilkan 1 tabel baru dari hasil pemecahan tabel transaksi yaitu tabel detail barang yang isinya menampung barang-barang yang dibeli.

Kenapa di tabel detail barang terdapat kolom harga lagi? padahal kolom harga sudah ada di tabel barang.

Kolom harga pada tabel detail barang digunakan untuk menyimpan harga barang pada saat proses transaksi. Jadi, meskipun kolom harga pada tabel barang berubah (naik/turun), harga barang yang ada pada tabel detail barang tidak ikut berubah (fixed). Bayangkan jika kita tidak menambahkan kolom harga pada pada tabel detail barang, maka yang terjadi total invoice dari transaksi akan berubah seiring berubahnya harga barang.

Evaluasi / Soal Latihan

1. Apa itu normalisasi basis data dan mengapa penting dalam desain basis data?
2. Jelaskan level-level normalisasi basis data!
3. Buatlah sebuah tabel pada basis data dan normalisasikan hingga mencapai level 3NF!

BAB 8

DATABASE MANAGEMENT SYSTEM

Tujuan Pembelajaran

Tujuan dari pembelajaran ini adalah agar mahasiswa memahami konsep dan arsitektur DBMS, mampu merancang dan mengimplementasikan basis data, mampu mengoptimalkan kinerja DBMS dan praktik pengelolaan basis data secara efektif dan efisien

Sejarah DBMS

Database Management System (DBMS) adalah sistem perangkat lunak yang dirancang untuk mengelola dan mengorganisir data dalam sebuah database. Sejarah DBMS dimulai pada tahun 1960-an ketika perusahaan-perusahaan besar mulai membutuhkan sistem yang lebih efisien dan terstruktur untuk mengelola data mereka. Pada saat itu, sistem pengolahan data manual masih banyak digunakan dan seringkali memakan waktu yang lama dan tidak efisien.

Salah satu pendahulu DBMS adalah Integrated Data Store (IDS) yang dikembangkan oleh Charles Bachman pada tahun 1960-an. IDS memperkenalkan konsep model data hierarkis yang kemudian

diadopsi oleh DBMS pertama yang populer, yaitu IBM›s Information Management System (IMS), yang diluncurkan pada tahun 1968.

Pada tahun 1970-an, Edgar Codd memperkenalkan model data relasional yang menjadi dasar untuk pengembangan DBMS relasional modern. Pada tahun yang sama, Oracle Corporation memperkenalkan produk DBMS pertama mereka yang berbasis SQL, yaitu Oracle RDBMS. Pada tahun 1980-an, IBM meluncurkan DB2, sebuah DBMS relasional yang sukses di pasar.

Pada tahun 1990-an, dengan semakin banyaknya data yang perlu diolah, DBMS mulai mengadopsi teknologi seperti pemrosesan transaksi online (OLTP) dan pemrosesan analitik online (OLAP) untuk mengelola data dengan lebih efisien. Selain itu, teknologi internet dan komputasi awan juga memungkinkan DBMS untuk mengelola data dengan skala yang lebih besar dan meningkatkan kinerja serta keamanan.

Saat ini, DBMS menjadi sangat penting dalam dunia bisnis dan teknologi informasi. DBMS modern memiliki berbagai fitur dan fungsionalitas yang kompleks dan terus berkembang untuk mengatasi tantangan dalam mengelola data yang semakin besar dan kompleks.

Definisi dan Fungsi DBMS

Definisi Database Management System (DBMS) adalah perangkat lunak yang digunakan untuk mengatur, menyimpan, mengelola, dan memanipulasi data dalam basis data. DBMS memberikan akses ke data dan memungkinkan pengguna untuk melakukan operasi seperti pencarian data, penyimpanan, penghapusan, dan pembaruan data.

Fungsi DBMS meliputi:

1. Pemeliharaan Data: DBMS memungkinkan pengguna untuk membuat, memperbarui, dan menghapus data dalam basis data.
2. Akses Data: DBMS memberikan akses ke data dalam basis data secara terorganisir dan terstruktur.

3. Manajemen Basis Data: DBMS memungkinkan pengguna untuk mengelola basis data, termasuk mengatur struktur basis data dan memperbarui penggunaan memori dan ruang penyimpanan.
4. Keamanan Data: DBMS menyediakan fitur keamanan untuk melindungi data dari akses yang tidak sah dan untuk memastikan integritas data.
5. Pemulihan Data: DBMS menyediakan fitur pemulihan data untuk memulihkan data dari kerusakan atau kehilangan.
6. Pengolahan Transaksi: DBMS mendukung pengolahan transaksi yang aman dan efisien pada basis data.
7. Integrasi Data: DBMS memungkinkan pengguna untuk mengintegrasikan data dari berbagai sumber dalam satu basis data.

Komponen DBMS

Database Management System (DBMS) terdiri dari beberapa komponen penting yang berfungsi untuk mengelola dan mengorganisir data dalam database. Beberapa komponen tersebut antara lain:

1. Hardware: hardware merupakan komponen fisik dari komputer yang berperan sebagai tempat penyimpanan data dalam database. Hardware yang digunakan dalam DBMS harus memiliki kapasitas penyimpanan yang cukup besar, performa yang tinggi, dan reliabilitas yang tinggi agar data dapat tersimpan dan diakses dengan cepat dan aman.
2. Software: software DBMS adalah program yang berfungsi untuk mengelola database dan melakukan operasi seperti penyimpanan, pengambilan, pembaruan, dan penghapusan data. Beberapa jenis software DBMS yang umum digunakan antara lain MySQL, Oracle, Microsoft SQL Server, dan PostgreSQL.

3. Data: data merupakan informasi yang disimpan dalam database. Data dalam DBMS terdiri dari beberapa jenis, seperti data teks, gambar, audio, dan video.
4. Procedure: procedure adalah kumpulan instruksi yang dibuat untuk memudahkan pengelolaan database, seperti operasi backup, restore, atau migrasi data. Procedure dapat dibuat secara manual atau otomatis dengan menggunakan fitur yang disediakan oleh DBMS.
5. User: user adalah orang atau sistem yang menggunakan database. Pengguna DBMS dapat dibagi menjadi beberapa jenis, seperti administrator database, pengembang aplikasi, dan pengguna akhir. Setiap jenis pengguna memiliki hak akses yang berbeda terhadap database, tergantung pada tugas dan wewenang yang dimilikinya.

Dengan adanya komponen-komponen tersebut, DBMS dapat bekerja dengan efisien dan dapat mengelola data dalam database dengan mudah dan aman.

Aristektur DBMS

Arsitektur Database Management System (DBMS) adalah cara di mana komponen-komponen DBMS bekerja bersama untuk menyediakan layanan yang diperlukan untuk mengakses dan memanipulasi data dalam database. Arsitektur DBMS terdiri dari tiga lapisan utama:

1. **Lapisan Konseptual (Conceptual Level)**

   Lapisan ini merupakan abstraksi tertinggi dalam arsitektur DBMS, yang menggambarkan bagaimana data disimpan, diorganisir, dan dihubungkan satu sama lain. Pada lapisan ini, database dijelaskan secara independen dari aplikasi yang menggunakannya. Lapisan konseptual juga dikenal sebagai skema konseptual.

2. **Lapisan Logika atau Taktis (Logical or Tactical Level)**

   Lapisan ini menggambarkan bagaimana data diorganisir secara logis dan bagaimana permintaan pengguna diinterpretasikan dan diterjemahkan menjadi operasi pada data. Pada lapisan ini, database dijelaskan dalam bahasa formal, seperti model data relasional atau model data berbasis objek.

3. **Lapisan Fisik (Physical Level)**

   Lapisan ini menggambarkan bagaimana data disimpan di media penyimpanan fisik, seperti hard disk, tape, atau memori utama. Pada lapisan ini, data dijelaskan dalam bentuk yang dapat diakses oleh perangkat keras.

Selain tiga lapisan utama tersebut, arsitektur DBMS juga dapat mencakup komponen-komponen tambahan, seperti sistem manajemen transaksi (transaction management system), sistem keamanan, sistem pemulihan bencana, dan sistem replikasi. Semua komponen ini bekerja bersama-sama untuk menyediakan layanan DBMS yang terintegrasi dan andal.

Evaluasi / Soal Latihan

1. Jelaskan definisi dari DBMS dan sebutkan tujuan utama dari penggunaannya.
2. Apa saja komponen utama dalam sistem manajemen basis data (DBMS)? Jelaskan fungsi dari masing-masing komponen.

BAB 9

BAHASA BASIS DATA

Tujuan Pembelajaran

Dengan mempelajari bahasa Basis Data, mahasiswa akan memiliki keterampilan dasar yang diperlukan untuk bekerja dengan Basis Data dan memanfaatkannya secara efektif dalam lingkungan bisnis dan industri. Selain itu, pembelajaran bahasa Basis Data juga dapat membuka peluang karir yang luas dalam bidang TI dan bisnis.

Pendahuluan

Bahasa Basis Data adalah bahasa yang digunakan untuk berkomunikasi dengan Basis Data. Bahasa Basis Data dapat dibagi menjadi empat jenis, yaitu Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), dan Data Query Language (DQL).

Bahasa Basis Data adalah bahasa yang digunakan untuk berinteraksi dengan Basis Data. Bahasa ini memungkinkan pengguna untuk membuat, mengubah, dan menghapus objek dalam Basis Data, serta melakukan kueri untuk mengambil data.

Data Definition Language (DDL)

Data Definition Language (DDL) adalah bahasa Basis Data yang digunakan untuk membuat, mengubah, dan menghapus objek dalam Basis Data, seperti tabel, indeks, dan tampilan. DDL memungkinkan pengguna untuk mendefinisikan struktur Basis Data, termasuk definisi tabel, kolom, tipe data, dan batasan integritas referensial. Contoh penggunaan DDL adalah sebagai berikut:

1. **Membuat tabel**

   Untuk membuat tabel, digunakan perintah CREATE TABLE. Berikut adalah contoh sintaksisnya:

```
CREATE TABLE pelanggan (
id INT PRIMARY KEY,
nama VARCHAR(50),
alamat VARCHAR(100),
kota VARCHAR(50),
kodepos VARCHAR(10)
);
```

   Perintah di atas akan membuat tabel baru bernama «pelanggan» dengan lima kolom: id, nama, alamat, kota, dan kodepos. Kolom id akan menjadi kunci utama atau primary key.

2. **Mengubah tabel**

   Untuk mengubah struktur tabel, digunakan perintah ALTER TABLE. Berikut adalah contoh sintaksisnya:

```
ALTER TABLE pelanggan ADD email VARCHAR(100);
```

   Perintah di atas akan menambahkan kolom baru bernama «email» ke tabel «pelanggan».

3. **Menghapus tabel**

Untuk menghapus tabel, digunakan perintah DROP TABLE. Berikut adalah contoh sintaksisnya:

```
DROP TABLE pelanggan;
```

Perintah di atas akan menghapus tabel «pelanggan» beserta seluruh data yang ada di dalamnya.

Dalam penggunaannya, DDL sangat berguna untuk mendefinisikan struktur Basis Data dan memastikan bahwa Basis Data tersebut memenuhi kebutuhan pengguna. Namun, perlu diingat bahwa perintah DDL yang salah dapat menyebabkan kerusakan pada Basis Data, oleh karena itu penggunaannya harus dilakukan dengan hati-hati dan teliti.

Data Manipulation Language (DML)

Data Manipulation Language (DML) adalah bahasa Basis Data yang digunakan untuk memasukkan, mengubah, dan menghapus data dalam Basis Data. DML memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada data dalam Basis Data.

DML berisi perintah-perintah seperti SELECT, INSERT, UPDATE, dan DELETE, yang digunakan untuk mengambil, menambahkan, memperbarui, dan menghapus data dalam Basis Data.

Berikut adalah contoh penggunaan perintah DML dalam SQL:

1. **SELECT**

Perintah SELECT digunakan untuk menampilkan data yang ada dalam Basis Data. Berikut adalah contoh sintaksisnya:

```
SELECT * FROM pelanggan;
```

Perintah di atas akan menampilkan semua data dalam tabel «pelanggan».

2. INSERT

Perintah INSERT digunakan untuk menambahkan data baru ke dalam Basis Data. Berikut adalah contoh sintaksisnya:

```
INSERT INTO pelanggan (nama, alamat, kota, kodepos)
VALUES (‹John Doe›, ‹Jl. Jendral Sudirman No. 123›,
‹Jakarta›, ‹12345›);
```

Perintah ini akan menambahkan data baru ke dalam tabel «pelanggan».

3. UPDATE

Perintah UPDATE digunakan untuk memperbarui data yang ada dalam Basis Data. Berikut adalah contoh sintaksisnya:

```
UPDATE pelanggan SET alamat = ‹Jl. MH Thamrin No.
456›, kota = ‹Bandung› WHERE id = 1;
```

Perintah di atas akan memperbarui alamat dan kota pelanggan dengan id=1.

4. DELETE

Perintah DELETE digunakan untuk menghapus data yang ada dalam Basis Data. Berikut adalah contoh sintaksisnya:

```
DELETE FROM pelanggan WHERE id = 1;
```

Perintah ini akan menghapus data pelanggan dengan id=1 dari tabel «pelanggan».

Dalam penggunaannya, DML sangat berguna untuk memanipulasi atau mengubah data dalam Basis Data. Namun, perlu diingat bahwa perintah DML yang salah dapat menyebabkan kerusakan pada Basis Data, oleh karena itu penggunaannya harus dilakukan dengan hati-hati dan teliti.

Data Control Language (DCL)

Data Control Language (DCL) adalah bahasa Basis Data yang digunakan untuk mengatur hak akses pengguna ke Basis Data. DCL memungkinkan administrator Basis Data untuk menentukan siapa yang dapat mengakses Basis Data, serta jenis akses apa yang dapat diberikan ke pengguna, seperti hak akses baca, tulis, atau hapus.

Beberapa perintah DCL yang umum digunakan di antaranya adalah:

1. GRANT: Perintah ini digunakan untuk memberikan hak akses tertentu kepada pengguna dalam Basis Data. Contoh penggunaan perintah GRANT adalah sebagai berikut:

   ```
   GRANT SELECT ON table_name TO user_name;
   ```

   Perintah di atas memberikan hak akses SELECT pada table_name ke user_name.

2. REVOKE: Perintah ini digunakan untuk mencabut hak akses yang telah diberikan kepada pengguna dalam Basis Data. Contoh penggunaan perintah REVOKE adalah sebagai berikut:

   ```
   REVOKE SELECT ON table_name FROM user_name;
   ```

   Perintah di atas mencabut hak akses SELECT pada table_name dari user_name.

3. DENY: Perintah ini digunakan untuk mencegah pengguna untuk melakukan aksi tertentu pada Basis Data. Contoh penggunaan perintah DENY adalah sebagai berikut:

   ```
   DENY INSERT ON table_name TO user_name;
   ```

   Perintah di atas mencegah user_name untuk melakukan operasi INSERT pada table_name.

Dengan menggunakan DCL, administrator Basis Data dapat memastikan bahwa pengguna hanya dapat mengakses data yang sesuai dengan perannya dan hak akses yang diberikan kepadanya.

Ini sangat penting dalam menjaga keamanan dan privasi data dalam Basis Data.

Data Query Language (DQL)

Data Query Language (DQL) adalah bahasa Basis Data yang digunakan untuk mengambil data dari Basis Data. DQL memungkinkan pengguna untuk melakukan kueri pada Basis Data, serta melakukan pengurutan, pengelompokan, dan penyeleksian data. DQL memungkinkan pengguna untuk mengakses data dalam Basis Data dan mengekstrak informasi dari Basis Data yang diinginkan.

Beberapa perintah DQL yang umum digunakan di antaranya adalah:

1. SELECT: Mengambil data dari satu atau lebih tabel dalam Basis Data. Contoh:

   SELECT * FROM customers;

   SELECT name, email FROM customers;

2. WHERE: Menentukan kriteria pencarian dalam Basis Data. Contoh:

   SELECT * FROM customers WHERE age > 30;

   SELECT * FROM customers WHERE city = ‹Jakarta›;

3. JOIN: Menggabungkan data dari dua atau lebih tabel dalam Basis Data. Contoh:

   SELECT * FROM orders JOIN customers ON orders. customer_id = customers.id;

   SELECT orders.order_number, customers.name FROM orders JOIN customers ON orders.customer_id = customers.id;

4. GROUP BY: Mengelompokkan data berdasarkan kolom tertentu dalam Basis Data. Contoh:

   SELECT city, COUNT(*) as total_customers FROM customers GROUP BY city;

   SELECT country, SUM(price) as total_sales FROM orders GROUP BY country;

5. ORDER BY: Mengurutkan data berdasarkan kolom tertentu dalam Basis Data. Contoh:

   SELECT * FROM customers ORDER BY age DESC;

   SELECT * FROM orders ORDER BY order_date ASC;

6. COUNT: Menghitung jumlah baris atau data dalam tabel. Contoh:

   SELECT COUNT(*) FROM customers;

   SELECT COUNT(DISTINCT city) FROM customers;

7. AVG: Menghitung rata-rata nilai dari kolom tertentu dalam tabel. Contoh:

   SELECT AVG(age) FROM customers;

   SELECT AVG(price) FROM orders WHERE country = ‹Indonesia›;

8. MAX dan MIN: Menampilkan nilai maksimum atau minimum dari kolom tertentu dalam tabel. Contoh:

   SELECT MAX(age) FROM customers;

   SELECT MIN(price) FROM orders WHERE country = ‹Indonesia›;

9. LIKE: Mencari data yang cocok dengan pola tertentu. Contoh:

   SELECT * FROM customers WHERE name LIKE ‹%ani%›;

   SELECT * FROM products WHERE product_name LIKE ‹M%›;

10. EXISTS: Mengecek apakah ada data yang sesuai dengan kriteria tertentu. Contoh:

    SELECT * FROM customers WHERE EXISTS (SELECT * FROM orders WHERE orders.customer_id = customers.id);

    SELECT * FROM orders WHERE EXISTS (SELECT * FROM products WHERE products.id = orders.product_id AND products.category = ‹Electronic›);

DQL sangat berguna untuk memfilter, mengurutkan, mengelompokkan, dan menggabungkan data dalam Basis Data. Dengan menggunakan perintah-perintah DQL yang tepat, pengguna dapat mengambil informasi yang spesifik dan relevan dari Basis Data.

Dengan menggunakan DQL, pengguna dapat mengambil informasi dari Basis Data dengan mudah dan cepat. DQL sangat penting dalam analisis data dan pengambilan keputusan bisnis karena memungkinkan pengguna untuk mengambil informasi yang spesifik dan relevan dari Basis Data.

Evaluasi / Soal Latihan

Jelaskan perbedaan antara DDL dan DML pada SQL serta berikan contoh dari masing-masing!

BAB 10

SQL SERVER

Tujuan Pembelajaran

Memperkenalkan dan mengajarkan konsep-konsep dasar serta keterampilan praktis dalam mengelola basis data menggunakan Microsoft SQL Server.

Pengertian SQL Server

SQL Server adalah sistem manajemen basis data relasional (RDBMS) yang dikembangkan oleh Microsoft. SQL Server digunakan untuk menyimpan, mengelola, dan memulihkan data untuk aplikasi perangkat lunak yang dapat digunakan oleh organisasi dan perusahaan untuk mengelola data mereka. SQL Server memungkinkan pengguna untuk membuat, mengelola, dan memodifikasi struktur basis data, dan melakukan berbagai operasi basis data, seperti pengambilan, penambahan, penghapusan, dan modifikasi data dalam basis data. SQL Server juga menyediakan fitur keamanan, integrasi data, dan analisis data. Dengan SQL Server, pengguna dapat membuat aplikasi basis data yang aman, andal, dan mudah diakses.

Dalam SQL Server, pengguna dapat membuat, memodifikasi, dan menghapus basis data, tabel, dan kolom dengan mudah menggunakan SQL Server Management Studio atau perintah SQL. Pengguna juga dapat membuat query SQL untuk mengambil data dari tabel dan mengelola data dengan menggunakan perintah DML seperti INSERT, UPDATE, dan DELETE.

Selain itu, SQL Server juga menyediakan fitur keamanan seperti autentikasi dan otorisasi pengguna, enkripsi data, dan audit log untuk melacak aktivitas pengguna dalam basis data. SQL Server juga mendukung teknologi keterhubungan dengan aplikasi dan layanan lain seperti .NET Framework, Web Services, dan XML.

Dengan banyaknya fitur dan dukungan dari Microsoft, SQL Server sangat populer di dunia bisnis dan industri. Mahasiswa yang mempelajari SQL Server dapat meningkatkan keterampilan dan pengetahuan dalam pengembangan dan pengelolaan basis data, serta dapat meningkatkan peluang karir di bidang teknologi informasi.

Fungsi SQL Server

SQL Server memiliki berbagai fungsi, antara lain:

1. Menyimpan data: SQL Server digunakan untuk menyimpan data dalam basis data dan menyediakan kemampuan untuk memanipulasi data tersebut.
2. Memproses data: SQL Server menyediakan berbagai fitur untuk memproses data, seperti pengelolaan transaksi, pembuatan tampilan data, pengelompokan data, dan pencarian data.
3. Mengamankan data: SQL Server dilengkapi dengan fitur keamanan yang memungkinkan pengguna untuk mengelola hak akses dan izin pengguna pada basis data.
4. Mengelola basis data: SQL Server menyediakan alat untuk mengelola basis data, termasuk alat pengelolaan basis data visual dan alat pemantauan kinerja.

5. Memfasilitasi aplikasi: SQL Server dapat digunakan untuk memfasilitasi aplikasi yang menggunakan basis data, seperti aplikasi web, aplikasi bisnis, dan aplikasi mobile.
6. Meningkatkan skalabilitas: SQL Server dapat diatur untuk meningkatkan skalabilitas, memungkinkan basis data untuk menangani volume data yang lebih besar dan lebih banyak pengguna secara efisien.
7. Meningkatkan ketersediaan: SQL Server menyediakan fitur untuk meningkatkan ketersediaan basis data, seperti replikasi data, pengelolaan backup dan restore, dan pengelolaan failover clustering.

Dengan fungsi-fungsi di atas, SQL Server dapat digunakan untuk memenuhi berbagai kebutuhan bisnis dan teknologi informasi, seperti pengolahan transaksi, analisis data, dan pengembangan aplikasi.

Kelebihan dan Kekurangan SQL Server

1. **Kelebihan SQL Server:**
    a. Terintegrasi dengan teknologi Microsoft: SQL Server berfungsi sebagai sistem manajemen basis data yang terintegrasi dengan teknologi Microsoft, seperti Visual Studio, .NET Framework, dan teknologi lainnya yang membuatnya mudah untuk digunakan.
    b. Skalabilitas yang tinggi: SQL Server dapat menangani sejumlah besar data dengan sangat efektif dan efisien, sehingga membuatnya sangat mudah untuk diintegrasikan dalam lingkungan bisnis yang besar.
    c. Keamanan: SQL Server menawarkan banyak opsi keamanan yang memungkinkan pengguna untuk mengontrol akses ke database, memperkuat enkripsi data, dan mengontrol izin pengguna.

d. Pengelolaan database yang mudah: SQL Server menyediakan berbagai fitur pengelolaan database yang mudah digunakan, termasuk backup dan pemulihan, pemeliharaan indeks, pemeliharaan statistik, dan pemeliharaan database secara umum.

2. Kekurangan SQL Server:

a. Biaya: SQL Server adalah sistem manajemen basis data yang berbayar, sehingga biaya lisensi dan perangkat keras untuk menjalankan SQL Server bisa menjadi mahal.
b. Konsumsi sumber daya yang tinggi: SQL Server bisa memakan banyak sumber daya pada server, yang dapat menyebabkan kinerja sistem yang lambat.
c. Kompleksitas: SQL Server menawarkan banyak fitur yang kompleks dan canggih, yang memerlukan pemahaman mendalam tentang teknologi database dan pengembangan aplikasi.
d. Tidak mendukung beberapa sistem operasi: SQL Server hanya dapat dijalankan di sistem operasi Microsoft Windows, sehingga tidak cocok untuk lingkungan yang menggunakan sistem operasi yang berbeda.

Aturan Perintah dalam SQL Server

Dalam SQL Server, perintah-perintah atau statement yang umum digunakan meliputi:

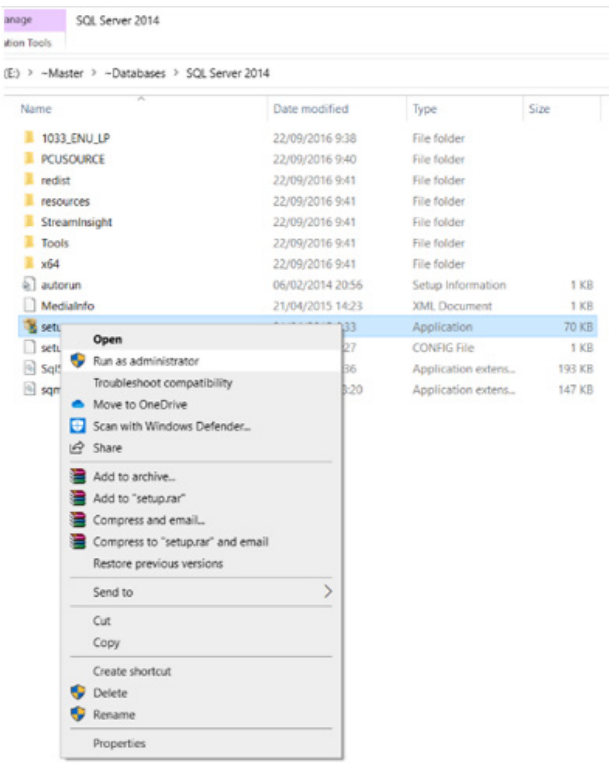
1. CREATE DATABASE: digunakan untuk membuat database baru.
2. CREATE TABLE: digunakan untuk membuat tabel baru dalam database.
3. ALTER TABLE: digunakan untuk mengubah struktur tabel yang sudah ada, seperti menambah atau menghapus kolom.
4. INSERT INTO: digunakan untuk memasukkan data ke dalam tabel.

5. SELECT: digunakan untuk melakukan query atau seleksi data dari tabel.
6. UPDATE: digunakan untuk memperbarui atau mengubah data yang sudah ada di dalam tabel.
7. DELETE: digunakan untuk menghapus data dari tabel.
8. JOIN: digunakan untuk menggabungkan data dari dua atau lebih tabel dalam satu query.
9. GROUP BY: digunakan untuk mengelompokkan data berdasarkan kolom tertentu.
10. ORDER BY: digunakan untuk mengurutkan data berdasarkan kolom tertentu.
11. WHERE: digunakan untuk memfilter data berdasarkan kriteria tertentu.
12. HAVING: digunakan untuk memfilter hasil query yang sudah di-generate melalui GROUP BY berdasarkan kriteria tertentu.
13. DISTINCT: digunakan untuk mengambil nilai unik dari suatu kolom.
14. TOP: digunakan untuk mengambil sejumlah data teratas dari hasil query.
15. BETWEEN: digunakan untuk memfilter data antara dua nilai tertentu.
16. IN: digunakan untuk memfilter data yang cocok dengan beberapa nilai tertentu.
17. LIKE: digunakan untuk memfilter data berdasarkan pola yang didefinisikan.
18. EXISTS: digunakan untuk memeriksa apakah data yang diinginkan ada dalam tabel lain atau tidak.
19. NOT: digunakan untuk membalikkan hasil operasi yang dilakukan.
20. UNION: digunakan untuk menggabungkan hasil query dari dua atau lebih query yang berbeda.

Instalasi SQL Server

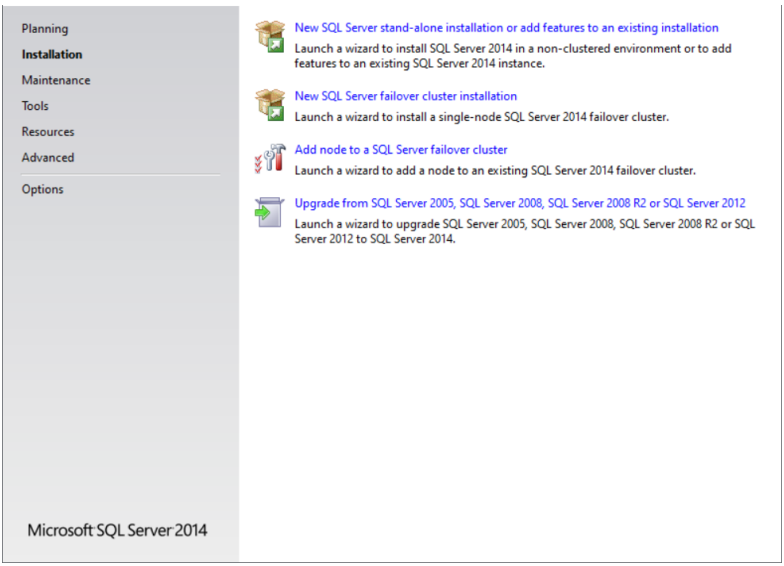
Berikut adalah langkah-langkah umum untuk melakukan instalasi SQL Server:

1. Pertama, download installer SQL Server dari situs resmi Microsoft atau media distribusi lainnya. Pastikan untuk memilih versi SQL Server yang sesuai dengan sistem operasi dan kebutuhan Anda.
2. Setelah file installer SQL Server berhasil diunduh, jalankan installer tersebut dengan mengklik kanan di setup.exe dan klik Run as administrator.



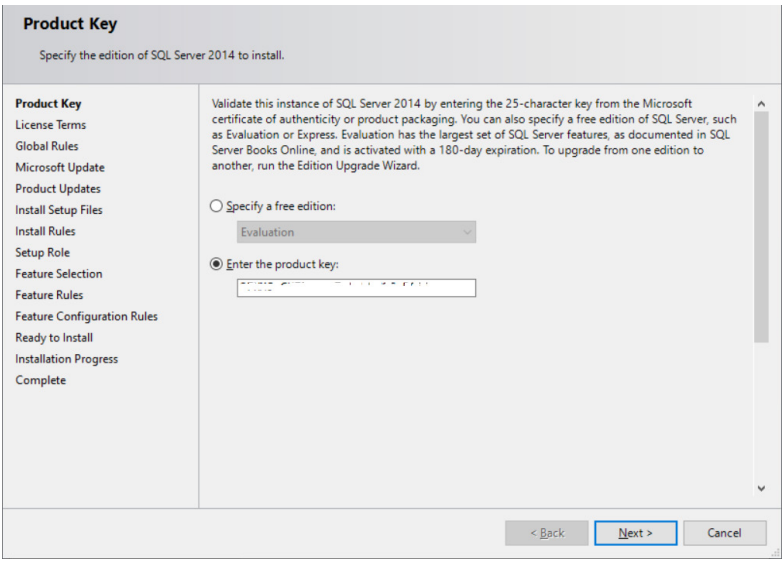
Gambar 8. 1 File Installer

3. Kemudian pilih menu Installation > New SQL Server stand-alone installation or add features to an existing installation.



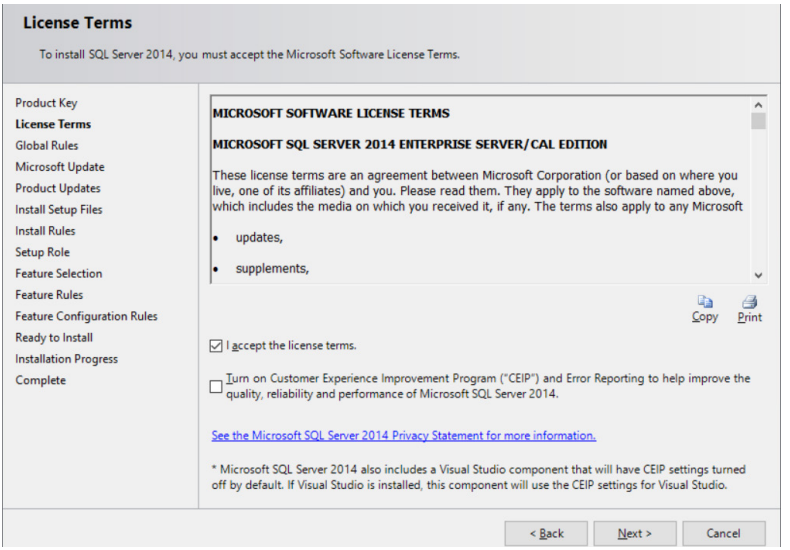
Gambar 8. 2 Installation

4. Masukkan Product Key lalu klik Next



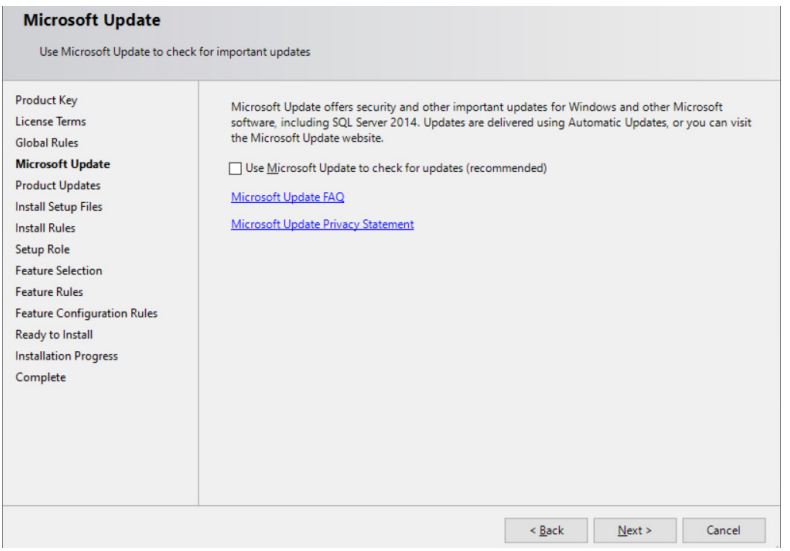
Gambar 8. 3 Product key

5. Pada menu Licence Terms, centang I accept the license terms.



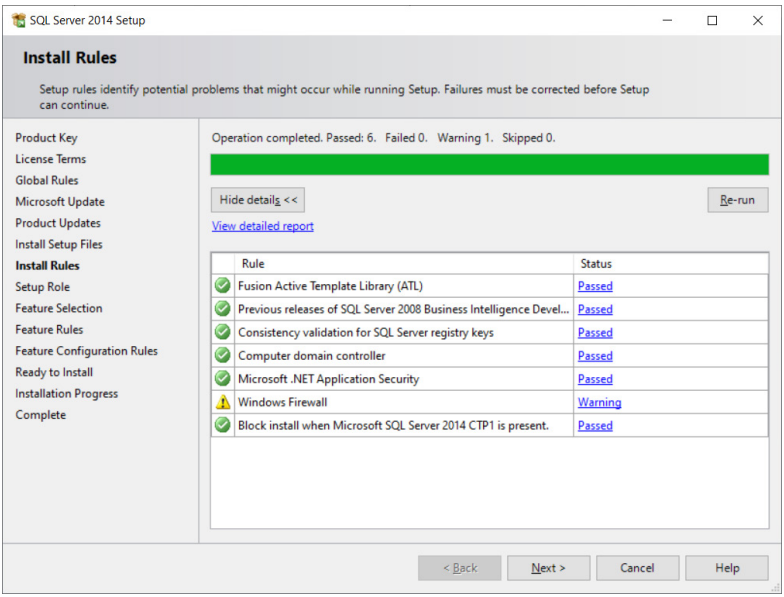
Gambar 8. 4 Licence Terms

6. Selanjutnya muncul proses Global Role, jika tidak ada masalah, maka lanjut pada bagian Microsoft Update, bisa dicentang atau tidak, saya pilih tidak dicentang, lalu next



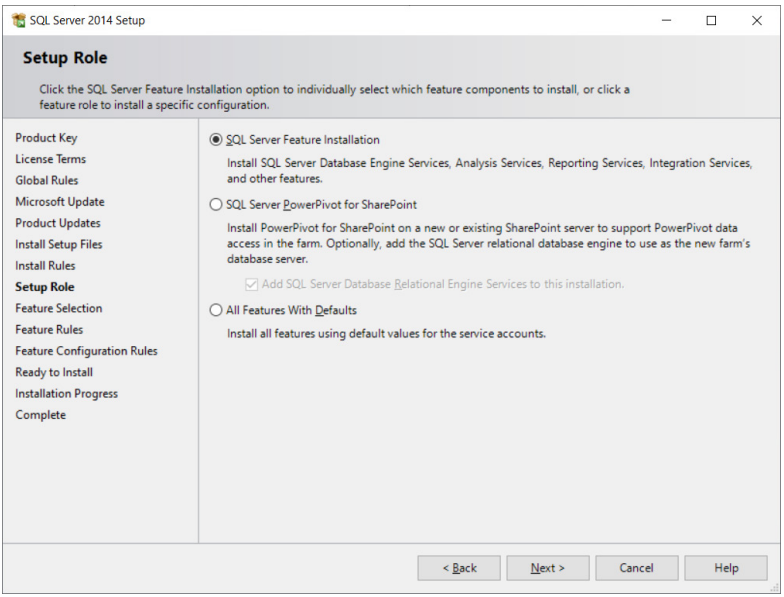
Gambar 8. 5 Microsoft Update

7. Muncul proses pengecheckan Instal Setup Files, jika tidak ada masalah maka lanjut ke proses selanjutnya.
8. Proses selanjutnya adalah pengecheckan Install Rules, jika tidak ada masalah atau hanya warning langsung aja Next.



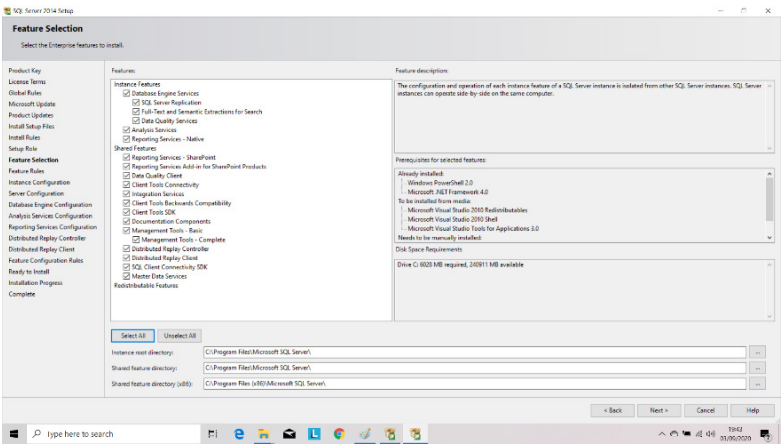
Gambar 8. 6 Install Rules

9. Proses selanjutnya adalah Setup Role, pilih SQL Server Feature Installation, lalu Next



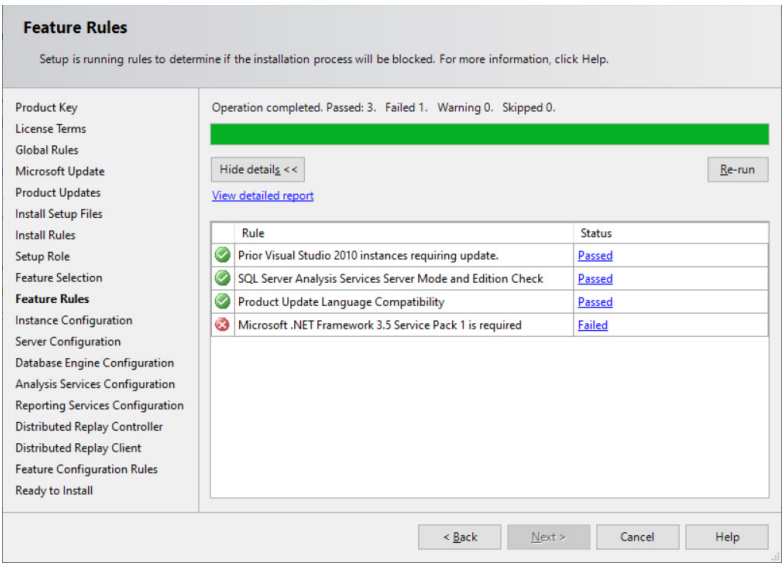
Gambar 8. 7 Setup Role

10. Selanjutnya proses Feature Selection. Untuk Features pilih Select All agar terpilih semua, lalu Next



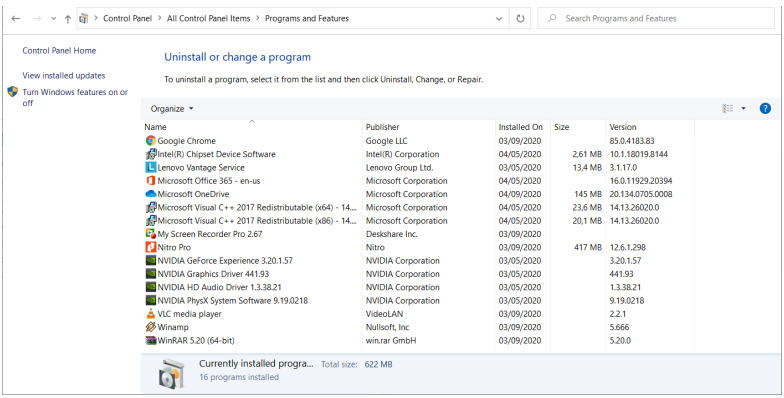
Gambar 8. 8 Feature Selection

11. Selanjutnya proses Feature Rules. Jika ada Rules yang statusnya Failed, maka harus diinstall terlebih dahulu. Jika tidak, maka tombol Next tidak aktif. Untuk Rules disini yang Failed adalah Microsoft .NET Framework 3.5 Service Pack 1 is required.



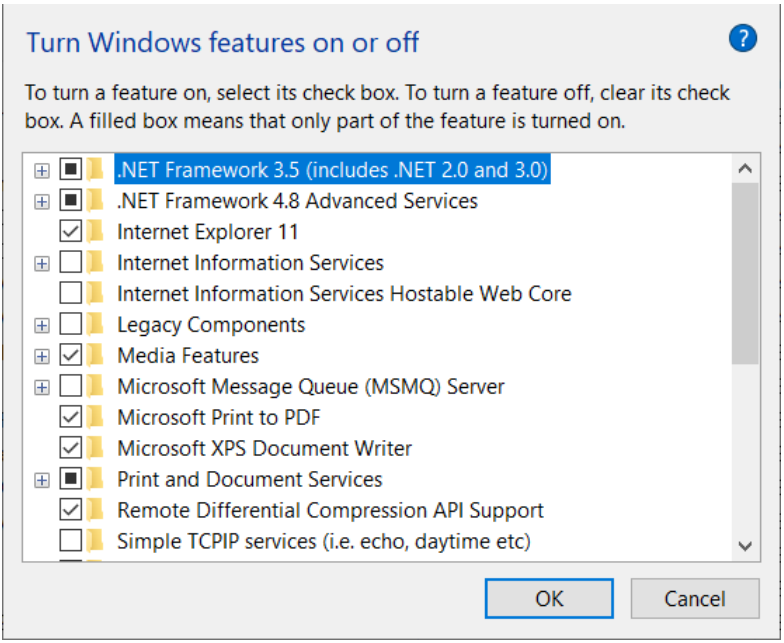
Gambar 8. 9 Feature Rules

12. Untuk melakukan instalasi, masuk ke Control Panel kemudian pilih Program and Features. Setelah itu, klik Turn Windows features on or off.



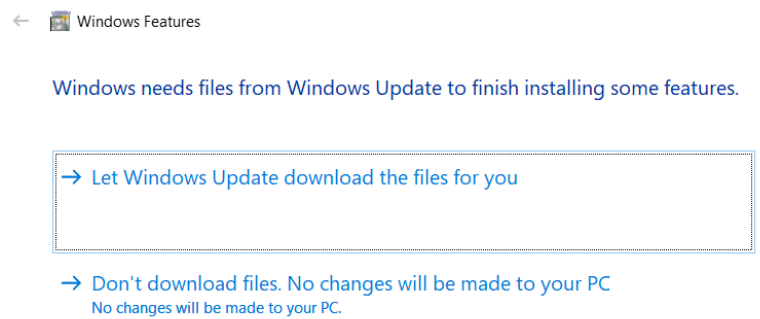
Gambar 8. 10 Program and Features

13. Maka akan tampil dialog box Windows Features. Centang .NET Framework 3.5 (includes .NET 2.0 and 3.0).



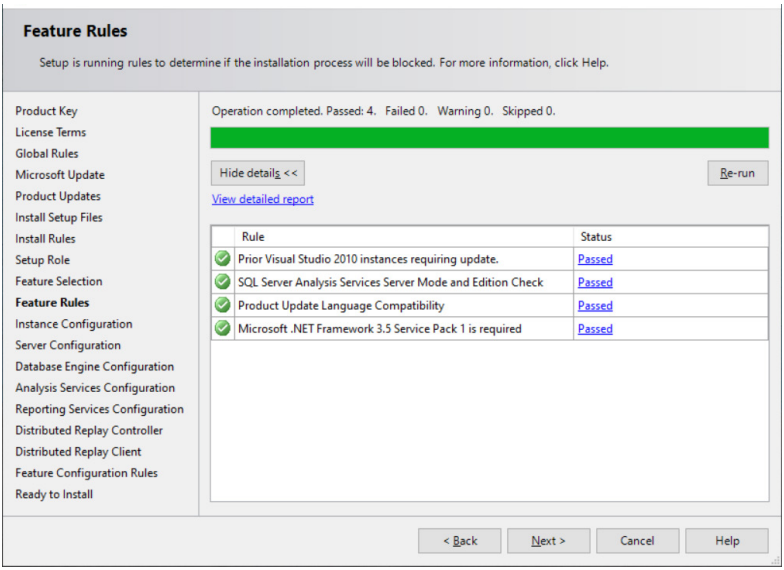
Gambar 8. 11 Windows Features

14. Pilih Let Windows Update download the files for you.



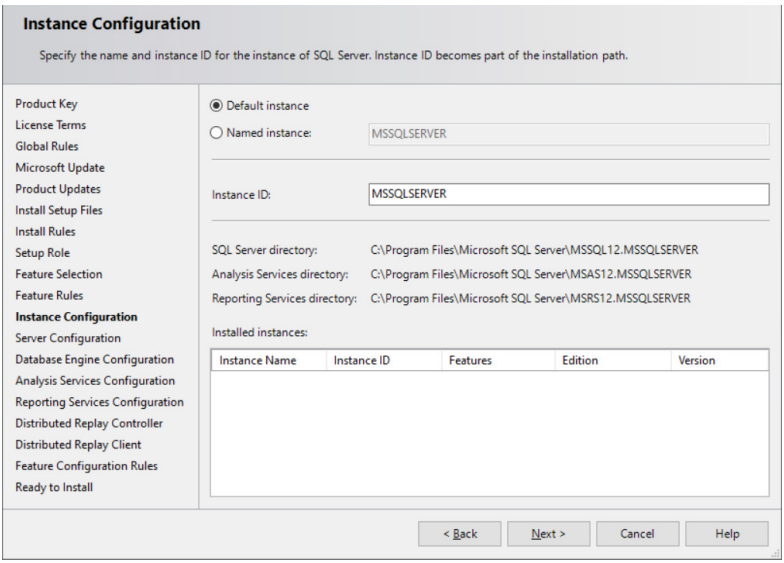
Gambar 8. 12 Windows Features

15. Tunggu hingga proses download dan instalasi selesai sehingga tampilan Feature Rules seperti gambar di bawah ini.



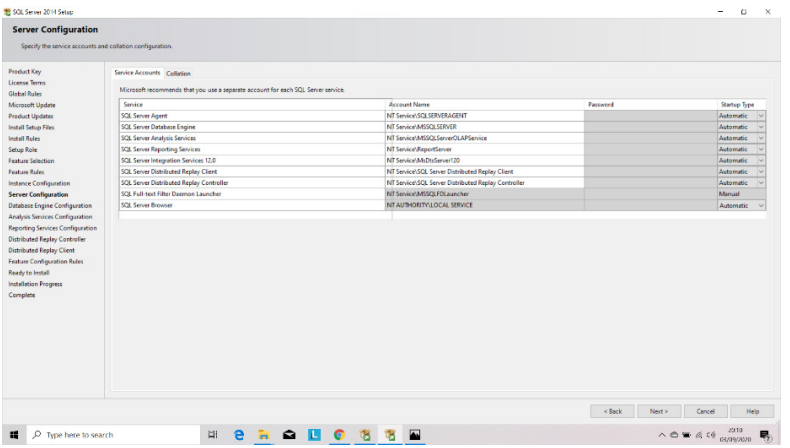
Gambar 8. 13 Feature Rules

16. Proses selanjutnya pilih Default Instance, lalu Next.



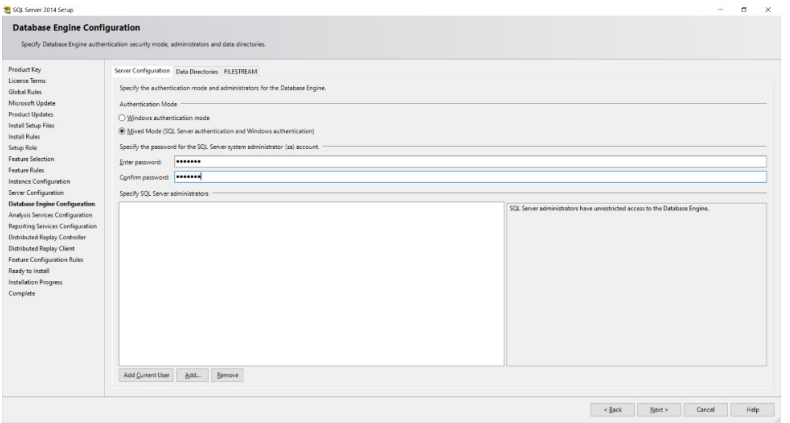
Gambar 8. 14 Instance Configuration

17. Proses selanjutnya Server Configuration. Untuk kolom Startup Type dirubah menjadi Automatic dan hanya satu saja yang Manual karena Default dari SQL Server nya.



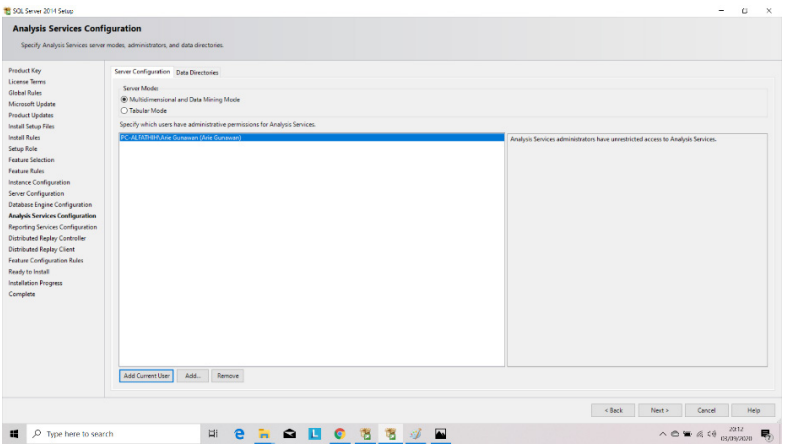
Gambar 8. 15 Server Configuration

18. Proses selanjutnya adalah Database Engine Configuration, pada Authentication Mode, Pilih Mixed Mode (SQL Server authentication and Windows authentication), lalu klik Add Curent User, maka akan keluar user pc anda. Lalu klik Next.



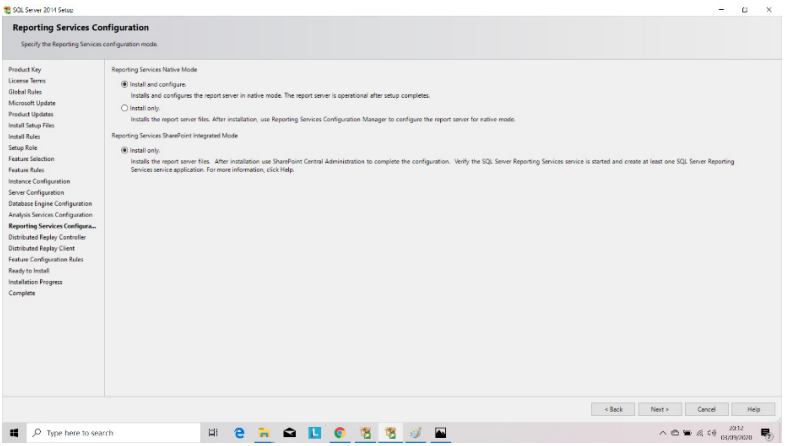
Gambar 8. 16 Database Engine Configuration

19. Proses selanjutnya klik Add Curent User, maka akan keluar user pc anda Lalu klik Next.



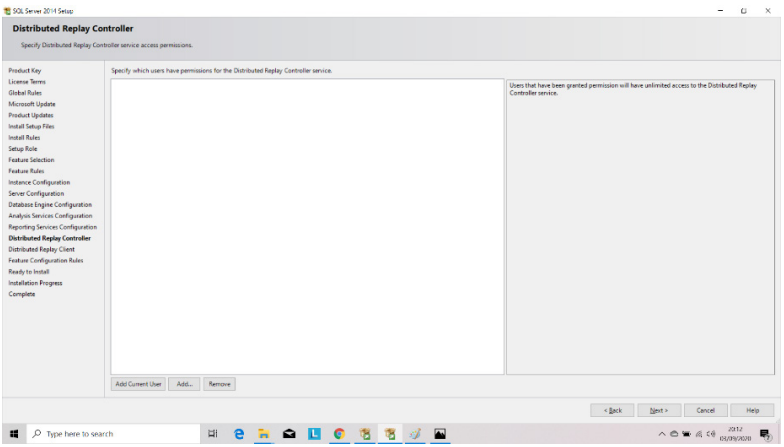
Gambar 8. 17 Analysis Services Configuration

20. Pada Proses Reporting Services Configuration, pilih Instal and configure, selanjutnya Next aja.



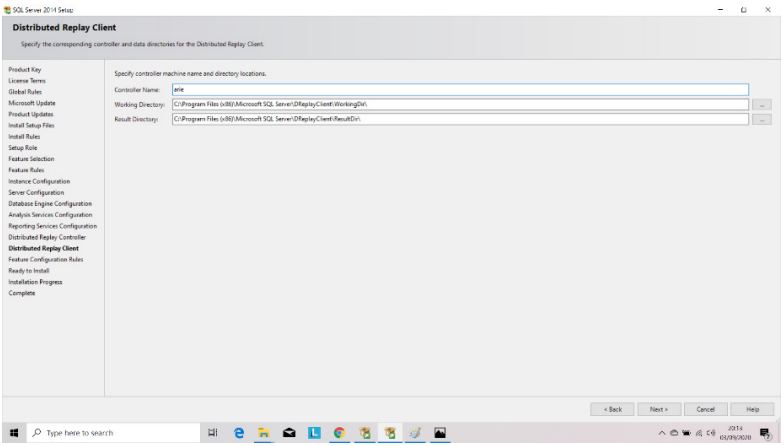
Gambar 8. 18 Reporting Services Configuration

21. Proses selanjutnya klik Add Curent User, maka akan keluar user pc anda. Lalu klik Next.



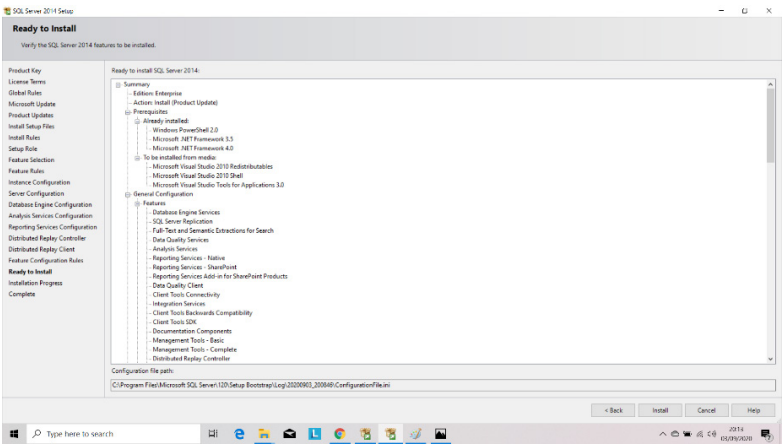
Gambar 8. 19 Distributed Replay Controller

22. Proses selanjutnya pada Control Name, ketik aja sesuai nama atau terserah anda, lalu Next.



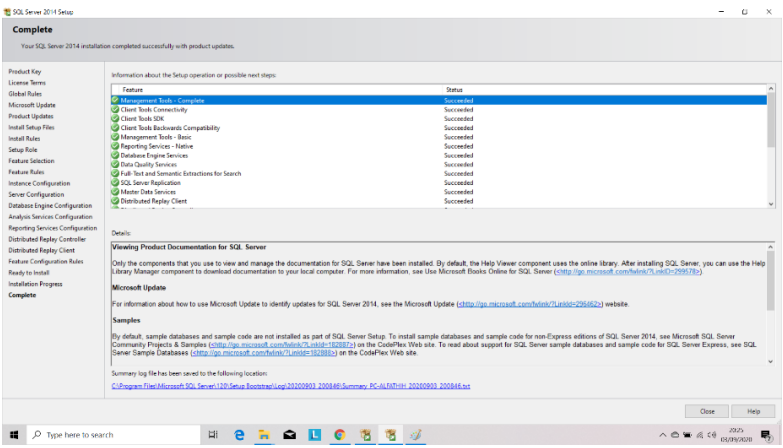
Gambar 8. 20 Distributed Replay Client

23. Setelah proses semua pengecheckan tidak ada yang eror, langsung saja klik Install.



Gambar 8. 21 Ready to Install

24. Setelah proses Installasi Microsoft SQL Server 2014 Berhasil, maka akan keluar jendela Complete, yang artinya Installasi Microsoft SQL Server 2014 anda telah berhasil.



Gambar 8. 22 Instalasi Complete

Tipe Data di SQL Server

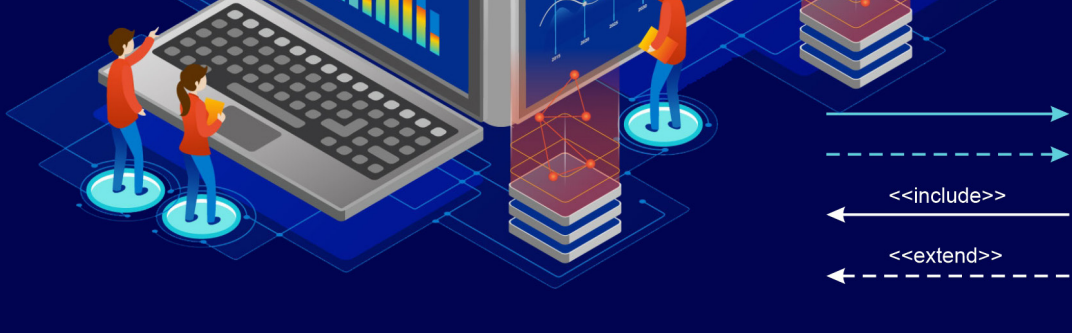
SQL Server memiliki beberapa tipe data untuk menyimpan berbagai jenis data, di antaranya:

1. Tipe Data Karakter (Character Data Types)
   a. CHAR(n): Menyimpan karakter dengan panjang tetap n.
   b. VARCHAR(n): Menyimpan karakter dengan panjang variabel maksimum n.
   c. TEXT: Menyimpan data karakter panjang variabel dengan batasan sekitar 2GB.
2. Tipe Data Bilangan (Numeric Data Types)
   a. INT: Menyimpan nilai bilangan bulat dengan rentang -2,147,483,648 hingga 2,147,483,647.
   b. BIGINT: Menyimpan nilai bilangan bulat yang lebih besar dengan rentang -9,223,372,036,854,775,808 hingga 9,223,372,036,854,775,807.
   c. FLOAT(n): Menyimpan bilangan pecahan dengan digit desimal hingga n.
3. Tipe Data Tanggal/Waktu (Date/Time Data Types)
   a. DATE: Menyimpan nilai tanggal dengan format YYYY-MM-DD.
   b. TIME: Menyimpan nilai waktu dengan format HH:MI:SS.
   c. DATETIME: Menyimpan nilai tanggal dan waktu dengan format YYYY-MM-DD HH:MI:SS.
4. Tipe Data Binary (Binary Data Types)
   a. BINARY(n): Menyimpan data biner dengan panjang tetap n.
   b. VARBINARY(n): Menyimpan data biner dengan panjang variabel maksimum n.
5. Tipe Data Lainnya (Other Data Types)
   a. BOOLEAN: Menyimpan nilai boolean (true/false).
   b. XML: Menyimpan data XML.
   c. JSON: Menyimpan data JSON.

Setiap tipe data memiliki aturan dan batasan masing-masing dalam penggunaannya di SQL Server.

Evaluasi / Soal Latihan

1. Bagaimana proses instalasi SQL Server? Jelaskan langkah-langkah yang perlu dilakukan.
2. Apa saja kelebihan dan kekurangan dari SQL Server?
3. Buatlah tabel tipe data yang ada di SQL Server!



BAB 11

DATA DEFINITION LANGUAGE (DDL)

Tujuan Pembelajaran

Mahasiswa diharapkan mampu memahami penggunaan perintah dasar DDL di SQL Server, mulai dari CREATE DATABASE, DROP DATABASE, CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, dan DROP INDEX.

Perintah Dasar DDL di SQL Server

Berikut adalah beberapa perintah dasar DDL (Data Definition Language) dalam SQL Server:

1. **CREATE DATABASE**

   Perintah ini digunakan untuk membuat sebuah database baru di SQL Server. Contohnya:

   ```
   CREATE DATABASE myDatabase;
   ```

2. **DROP DATABASE**

   Perintah ini digunakan untuk menghapus sebuah database di SQL Server. Contohnya:

   ```
   DROP DATABASE myDatabase;
   ```

3. CREATE TABLE

Perintah ini digunakan untuk membuat sebuah tabel baru di dalam database. Contohnya:

```
CREATE TABLE employees (
id INT PRIMARY KEY,
name VARCHAR(50),
age INT,
salary DECIMAL(10, 2)
);
```

Perintah di atas akan membuat sebuah tabel bernama «employees» dengan empat kolom yaitu «id», «name», «age», dan «salary».

4. ALTER TABLE

Perintah ini digunakan untuk mengubah sebuah tabel yang sudah ada di dalam database. Contohnya, jika ingin menambahkan kolom baru ke dalam tabel «employees»:

```
ALTER TABLE employees
ADD email VARCHAR(100);
```

5. DROP TABLE

Perintah ini digunakan untuk menghapus sebuah tabel di dalam database. Contohnya:

```
DROP TABLE employees;
```

6. CREATE INDEX

Perintah ini digunakan untuk membuat sebuah indeks pada sebuah tabel untuk mempercepat pencarian data. Contohnya:

```
CREATE INDEX idx_employee_name
ON employees (name);
```

Perintah di atas akan membuat sebuah indeks bernama «idx_employee_name» pada kolom «name» di dalam tabel «employees».

7. **DROP INDEX**

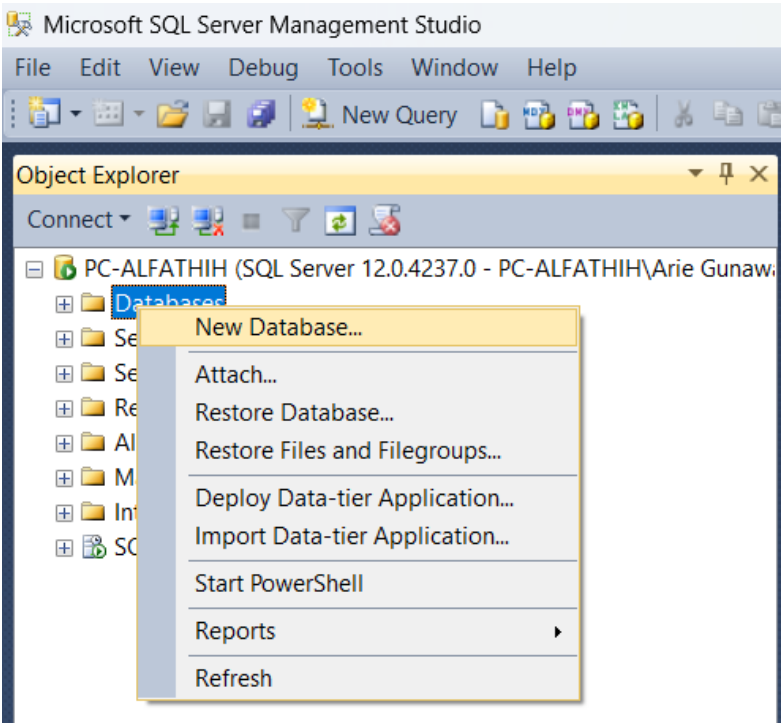
   Perintah ini digunakan untuk menghapus sebuah indeks dari sebuah tabel. Contohnya:

```
DROP INDEX idx_employee_name
ON employees;
```

Menciptakan Basis Data di SQL Server

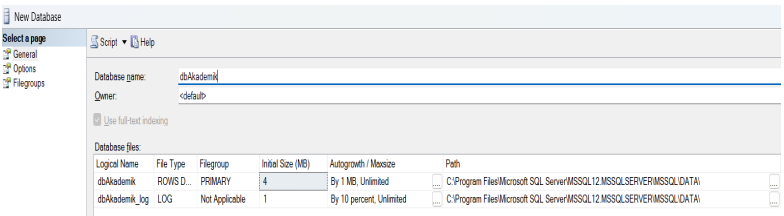
Berikut adalah langkah-langkah untuk menciptakan basis data di SQL Server:

1. Buka SQL Server Management Studio.
2. Login menggunakan akun yang memiliki hak akses sebagai administrator SQL Server.
3. Pilih folder "Databases" di Object Explorer.
4. Klik kanan pada folder "Databases" dan pilih "New Database".



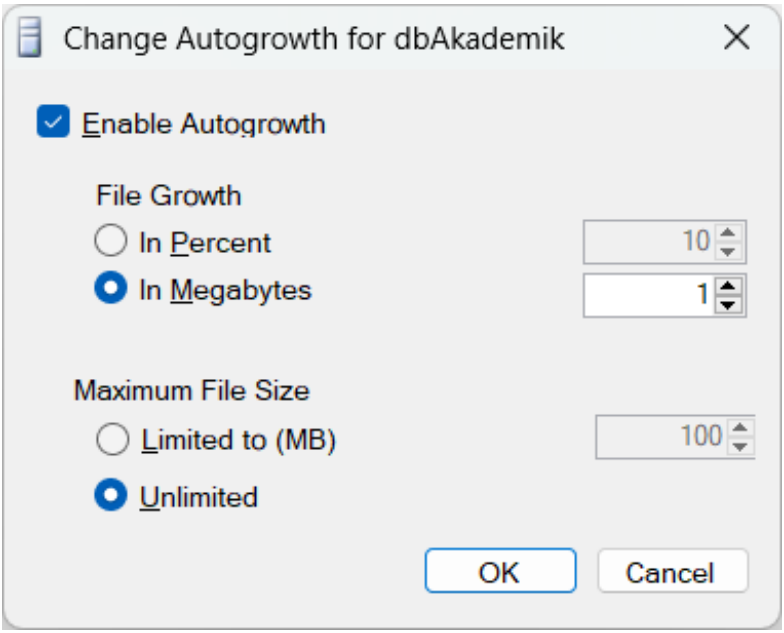
Gambar 9. 1 Create Database

5. Muncul jendela "New Database", pada tab "General" terdapat beberapa opsi yang harus diisi, di antaranya:
   a. Database name: Nama basis data yang akan dibuat.
   b. Owner: Nama pemilik basis data. Secara default, nama pemilik akan diisi dengan akun yang digunakan untuk login.
   c. Collation: Konfigurasi tata bahasa dan pengaturan karakter untuk basis data.



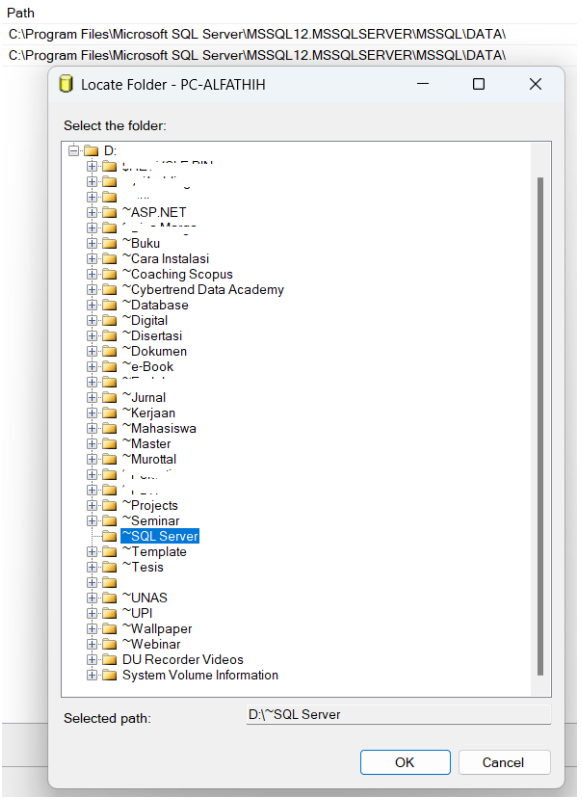
Gambar 9. 2 Create Database

6. Untuk kolom Autogrowth / Maxsize bisa disesuaikan dengan kebutuhan.



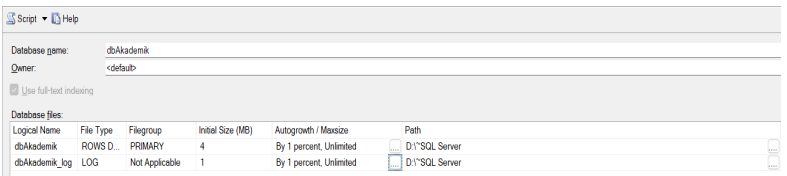
Gambar 9. 3 Autogrowth

7. Untuk kolom Path, diusahakan jangan disimpan di drive C, di contoh ini penulis menyarankan di drive D.



Gambar 9. 4 Path Database

8. Hasil akhir dari pengisian di atas adalah seperti berikut.



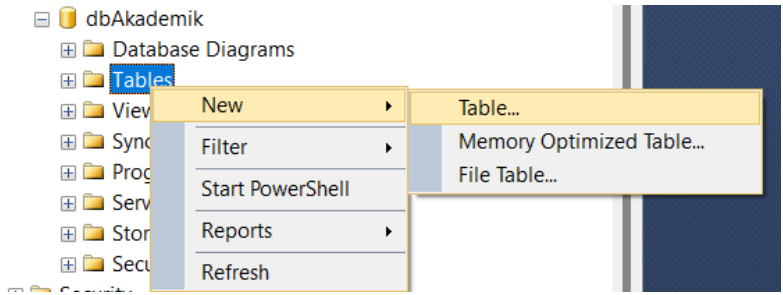
Gambar 9. 5 Create Datbase

9. Setelah semua opsi terisi, klik "OK" untuk membuat basis data baru.
10. Basis data yang baru dibuat akan muncul pada folder "Databases" di Object Explorer.

Menciptakan Tabel dalam Basis Data di SQL Server

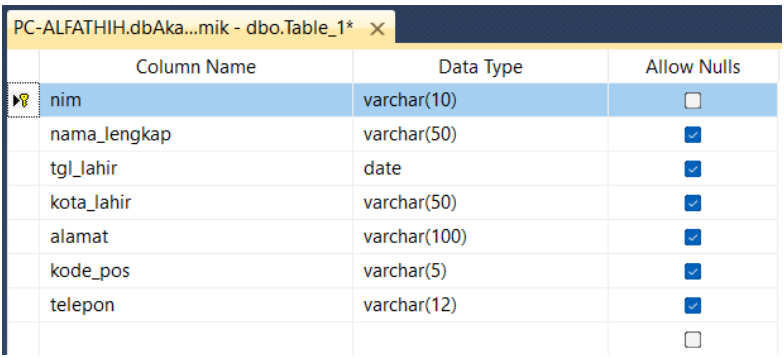
Berikut adalah langkah-langkah dalam membuat table di SQL Server:

1. Klik kanan di database yang telah dibuat, pilih Tables, kemudian New, dan klik Table.



Gambar 9. 6 Create Table

2. Akan muncul pembuatan table baru seperti di bawah ini. Kemudian isikan masing-masing kolom sesuai kebutuhan. Di sini penulis memberi contoh pembuatan table mahasiswa dengan Primary Key nya adalah NIM.

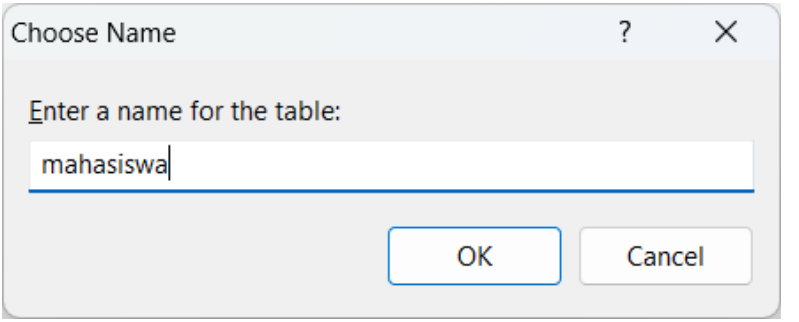


Column Name	Data Type	Allow Nulls
nim	varchar(10)	☐
nama_lengkap	varchar(50)	☑
tgl_lahir	date	☑
kota_lahir	varchar(50)	☑
alamat	varchar(100)	☑
kode_pos	varchar(5)	☑
telepon	varchar(12)	☑
		☐

Gambar 9. 7 Create Table

Untuk membuat Primary Key caranya adalah dengan memilih salah satu atau lebih kolom, kemudian tekan icon kunci .

3. Simpan table dengan cara tekan icon save hingga muncul tampilan seperti di bawah ini.



Gambar 9. 8 Create Table

Beri nama table sesuai kebutuhan, contoh di atas penulis beri nama tablenya adalah mahasiswa.

4. Untuk menambahkan table baru lagi, langkah yang dilakukan sama seperti di atas.

Evaluasi / Soal Latihan

1. Buatlah tabel mahasiswa dengan kolom-kolom berikut: id (integer), nama (varchar), jurusan (varchar), dan umur (integer).
2. Tambahkan kolom alamat (varchar) pada tabel mahasiswa.
3. Buatlah tabel dosen dengan kolom-kolom berikut: id (integer), nama (varchar), dan email (varchar).
4. Buatlah tabel mata_kuliah dengan kolom-kolom berikut: id (integer), nama (varchar), dan sks (integer).
5. Buatlah tabel kelas dengan kolom-kolom berikut: id (integer), nama (varchar), dan dosen_id (integer).

BAB 12

DATA MANIPULATION LANGUAGE (DML)

Tujuan Pembelajaran

Mahasiswa diharapkan mampu memahami penggunaan perintah dasar DML di SQL Server, mulai dari INSERT INTO, UPDATE, DELETE, dan SELECT.

Perintah Dasar DML di SQL Server

Berikut adalah beberapa perintah DML (Data Manipulation Language) di SQL Server:

1. **INSERT INTO**

   Perintah ini digunakan untuk memasukkan data baru ke dalam tabel. Contohnya:

   ```
   INSERT INTO nama_tabel (kolom1, kolom2, kolom3)
   VALUES (nilai1, nilai2, nilai3);
   ```

2. **UPDATE**

   Perintah ini digunakan untuk memperbarui data yang sudah ada dalam tabel. Contohnya:

   ```
   UPDATE nama_tabel

   SET kolom1 = nilai1, kolom2 = nilai2

   WHERE kondisi;
   ```

3. **DELETE**

   Perintah ini digunakan untuk menghapus data dari tabel. Contohnya:

   ```
   DELETE FROM nama_tabel

   WHERE kondisi;
   ```

4. **SELECT**

   Perintah ini digunakan untuk menampilkan data dari tabel. Contohnya:

   ```
   SELECT kolom1, kolom2, kolom3

   FROM nama_tabel

   WHERE kondisi;
   ```

- Keterangan:
  nama_tabel: Nama tabel yang ingin diproses
  kolom1, kolom2, kolom3: Nama kolom yang ingin diproses
  nilai1, nilai2, nilai3: Nilai yang ingin dimasukkan ke dalam kolom
  kondisi: Kondisi untuk memfilter data yang ingin diproses

Perlu diingat bahwa perintah DML harus dijalankan pada sebuah database yang sudah terhubung pada SQL Server.

Menambah Data di SQL Server

Untuk menambah data ke dalam tabel pada SQL Server, dapat menggunakan perintah '**INSERT INTO'**. Berikut adalah sintaks perintah dasar untuk menambahkan data ke dalam tabel:

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

VALUES (nilai_kolom1, nilai_kolom2, nilai_kolom3, ...);
```

Contoh penggunaan perintah '**INSERT INTO'** untuk menambahkan data ke dalam tabel produk:

```
INSERT INTO produk (id_produk, nama_produk, harga)

VALUES (1, ‹Sepatu›, 150000);
```

Perintah tersebut akan menambahkan data baru ke dalam tabel '**produk'** dengan nilai kolom '**id_produk'** adalah 1, '**nama_produk'** adalah ‹Sepatu›, dan '**harga**' adalah 150000.

Jika ingin menambahkan lebih dari satu baris data sekaligus, dapat menggunakan perintah '**INSERT INTO**' dengan multiple rows, seperti contoh berikut:

```
INSERT INTO produk (id_produk, nama_produk, harga)

VALUES

(1, ‹Sepatu›, 150000),

(2, ‹Baju›, 100000),

(3, ‹Celana›, 120000);
```

Perintah tersebut akan menambahkan tiga baris data ke dalam tabel '**produk'**.

Menampilkan Data di SQL Server

Untuk menampilkan data di SQL Server, Anda dapat menggunakan perintah SELECT. Berikut adalah contoh perintah SELECT:

```
SELECT column1, column2, ...

FROM table_name;
```

Di sini, '**column1**', '**column2**', ... adalah nama kolom yang ingin Anda tampilkan, sedangkan '**table_name**' adalah nama tabel dari mana Anda ingin menampilkan data.

Anda juga dapat menambahkan beberapa kriteria untuk memfilter data yang ditampilkan dengan menggunakan perintah WHERE. Berikut adalah contoh perintah SELECT dengan kriteria:

```
SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

Di sini, '**condition**' adalah kriteria yang harus dipenuhi oleh data yang ingin ditampilkan. Anda juga dapat mengurutkan hasil dengan menambahkan perintah ORDER BY. Berikut adalah contoh perintah SELECT dengan pengurutan:

```
SELECT column1, column2, ...

FROM table_name

WHERE condition

ORDER BY column_name ASC|DESC;
```

Di sini, '**column_name**' adalah nama kolom yang ingin Anda gunakan untuk mengurutkan hasil. Anda juga harus menentukan apakah Anda ingin mengurutkan secara ascending (ASC) atau descending (DESC).

Selain itu, Anda juga dapat menggunakan fungsi agregat seperti COUNT, SUM, AVG, MIN, dan MAX untuk melakukan operasi pada data. Berikut adalah contoh perintah SELECT dengan fungsi agregat:

```
SELECT COUNT(column_name)

FROM table_name;
```

Di sini, '**COUNT(column_name)'** akan menghitung jumlah nilai yang tidak null di kolom yang diberikan.

Mengubah Data di SQL Server

Untuk mengubah data pada tabel di SQL Server, digunakan perintah '**UPDATE'**. Berikut adalah sintaks dari perintah '**UPDATE'**:

```
UPDATE nama_tabel

SET kolom1=nilai1, kolom2=nilai2, ...

WHERE kondisi;
```

Penjelasan dari sintaks tersebut:

1. **'UPDATE'**: keyword untuk mengubah data pada tabel
2. **'nama_tabel'**: nama tabel yang ingin diubah datanya
3. **'SET'**: keyword untuk mengatur nilai pada kolom yang ingin diubah
4. **'kolom1'**, '**kolom2'**, ...: nama kolom yang ingin diubah nilainya
5. **'nilai1'**, '**nilai2'**, ...: nilai baru yang ingin diisikan pada kolom tersebut
6. **'WHERE'**: keyword untuk menentukan kondisi untuk mengubah data pada kolom tertentu saja
7. **'kondisi'**: kondisi yang harus terpenuhi agar data pada kolom tersebut dapat diubah

Berikut adalah contoh penggunaan perintah '**UPDATE'** untuk mengubah data pada tabel '**pegawai'** di database '**contoh_database'**:

```
UPDATE contoh_database.pegawai

SET nama=›Ani›, umur=28

WHERE id_pegawai=1;
```

Perintah tersebut akan mengubah data pada kolom '**nama'** dan '**umur'** pada baris dengan '**id_pegawai'** sama dengan 1 menjadi '**Ani'** dan '**28'**.

Menghapus Data di SQL Server

Untuk menghapus data pada SQL Server, dapat digunakan perintah DELETE. Berikut adalah sintaks perintah DELETE:

```
DELETE FROM nama_tabel

WHERE kondisi;
```

Keterangan:

1. **'DELETE FROM**' digunakan untuk menghapus data dari tabel tertentu
2. **'nama_tabel**' adalah nama tabel yang ingin dihapus datanya
3. **'WHERE'** digunakan untuk menentukan kondisi atau filter data yang akan dihapus
4. **'kondisi'** adalah syarat yang harus dipenuhi agar data bisa dihapus

Contoh penggunaan perintah DELETE: Menghapus data pada tabel Mahasiswa dengan ID=1234

```
DELETE FROM Mahasiswa

WHERE ID = 1234;
```

Perintah tersebut akan menghapus data pada tabel Mahasiswa yang memiliki ID = 1234. Jika tidak ada kondisi yang ditentukan, maka perintah DELETE akan menghapus seluruh data pada tabel

tersebut. Oleh karena itu, penggunaan perintah DELETE harus hati-hati dan perlu dipastikan kondisinya sudah benar.

Perintah SELECT di SQL Server

Berisi tentang apa yang ingin dicapai oleh mahasiswa setelah mempelajari bab ini. Dapat disajikan dalam bentuk paragraf atau poin (nomor).

Operator Relasi di SQL Server

Operator relasi dalam SQL Server digunakan untuk membandingkan nilai antara dua atau lebih data atau nilai. Berikut adalah beberapa operator relasi yang tersedia di SQL Server:

1. **Operator sama dengan (=)**

   Digunakan untuk membandingkan nilai antara dua ekspresi. Misalnya:

   SELECT * FROM customers WHERE city = ‹New York›

   Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom city sama dengan string «New York».

2. **Operator tidak sama dengan (<>)**

   Digunakan untuk membandingkan nilai antara dua ekspresi dan mengembalikan hasil ketika nilai tidak sama. Misalnya:

   SELECT * FROM customers WHERE city <› ‹New York›

   Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom city tidak sama dengan string «New York».

3. **Operator lebih besar dari (>)**

   Digunakan untuk membandingkan nilai dan mengembalikan hasil ketika nilai pertama lebih besar dari nilai kedua. Misalnya:

```
SELECT * FROM customers WHERE age > 25
```

Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom age lebih besar dari 25.

4. **Operator lebih kecil dari (<)**

   Digunakan untuk membandingkan nilai dan mengembalikan hasil ketika nilai pertama lebih kecil dari nilai kedua. Misalnya:

```
SELECT * FROM customers WHERE age < 25
```

   Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom age lebih kecil dari 25.

5. **Operator lebih besar atau sama dengan (>=)**

   Digunakan untuk membandingkan nilai dan mengembalikan hasil ketika nilai pertama lebih besar atau sama dengan nilai kedua. Misalnya:

```
SELECT * FROM customers WHERE age >= 25
```

   Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom age lebih besar atau sama dengan 25.

6. **Operator lebih kecil atau sama dengan (<=)**

   Digunakan untuk membandingkan nilai dan mengembalikan hasil ketika nilai pertama lebih kecil atau sama dengan nilai kedua. Misalnya:

```
SELECT * FROM customers WHERE age <= 25
```

   Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom age lebih kecil atau sama dengan 25.

7. **Operator BETWEEN**

   Digunakan untuk membandingkan nilai dan mengembalikan hasil ketika nilai berada di antara dua nilai. Misalnya:

SELECT * FROM customers WHERE age BETWEEN 20 AND 30

Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom age berada di antara 20 dan 30.

8. **Operator LIKE**

Digunakan untuk mencari nilai yang cocok dengan pola tertentu. Misalnya:

SELECT * FROM customers WHERE name LIKE ‹John%›

Perintah ini akan mengembalikan semua baris dari tabel customers di mana nilai pada kolom name dimulai dengan string «John». Tanda % digunakan sebagai wildcard, yang dapat mewakili satu atau lebih karakter.

Operator Logika di SQL Server

SQL Server mendukung operator logika standar seperti AND, OR, dan NOT. Operator logika digunakan dalam klausa WHERE dan dapat membantu Anda mengembangkan kondisi yang lebih kompleks dalam pernyataan SQL.

Berikut adalah beberapa contoh penggunaan operator logika di SQL Server:

1. **Operator AND:**

SELECT * FROM customers

WHERE city = ‹New York› AND state = ‹NY›;

Perintah di atas akan mengembalikan semua pelanggan yang berasal dari kota New York dan negara bagian New York.

2. **Operator OR:**

   SELECT * FROM orders

   WHERE status = ‹pending› OR status = ‹processing›;

   Perintah di atas akan mengembalikan semua pesanan yang memiliki status «pending» atau «processing».

3. **Operator NOT:**

   SELECT * FROM products

   WHERE NOT category = ‹electronics›;

   Perintah di atas akan mengembalikan semua produk yang tidak termasuk dalam kategori «electronics».

Dalam penggunaannya, operator logika biasanya digunakan bersamaan dengan operator perbandingan dan operator lainnya untuk membuat kondisi yang lebih kompleks dalam klausa '**WHERE**'.

Operator Pembanding di SQL Server

Operator pembanding atau comparison operator adalah operator yang digunakan untuk membandingkan nilai antara dua atau lebih data. Dalam SQL Server, operator pembanding sering digunakan dalam pernyataan SELECT, WHERE, dan JOIN untuk memfilter data yang diambil dari tabel atau menghubungkan beberapa tabel.

Berikut adalah beberapa operator pembanding yang digunakan dalam SQL Server:

1. **= (sama dengan)**

   Operator = digunakan untuk membandingkan apakah nilai di kedua sisi operator sama atau tidak. Contohnya:

   SELECT * FROM tabel WHERE kolom = ‹nilai›

2. **<> atau != (tidak sama dengan)**

   Operator <> atau != digunakan untuk membandingkan apakah nilai di kedua sisi operator tidak sama. Contohnya:

   ```
   SELECT * FROM tabel WHERE kolom <> <nilai>
   ```

3. **> (lebih besar dari)**

   Operator > digunakan untuk membandingkan apakah nilai di sebelah kiri operator lebih besar dari nilai di sebelah kanan. Contohnya:

   ```
   SELECT * FROM tabel WHERE kolom > 10
   ```

4. **< (lebih kecil dari)**

   Operator < digunakan untuk membandingkan apakah nilai di sebelah kiri operator lebih kecil dari nilai di sebelah kanan. Contohnya:

   ```
   SELECT * FROM tabel WHERE kolom < 10
   ```

5. **>= (lebih besar dari atau sama dengan)**

   Operator >= digunakan untuk membandingkan apakah nilai di sebelah kiri operator lebih besar dari atau sama dengan nilai di sebelah kanan. Contohnya:

   ```
   SELECT * FROM tabel WHERE kolom >= 10
   ```

6. **<= (lebih kecil dari atau sama dengan)**

   Operator <= digunakan untuk membandingkan apakah nilai di sebelah kiri operator lebih kecil dari atau sama dengan nilai di sebelah kanan. Contohnya:

   ```
   SELECT * FROM tabel WHERE kolom <= 10
   ```

7. **BETWEEN (antara)**

   Operator BETWEEN digunakan untuk membandingkan apakah nilai di sebelah kiri operator berada di antara nilai di sebelah kanan. Contohnya:

   SELECT * FROM tabel WHERE kolom BETWEEN 10 AND 20

8. **LIKE (mirip)**

   Operator LIKE digunakan untuk membandingkan apakah nilai di sebelah kiri operator mirip dengan pola di sebelah kanan. Pola biasanya menggunakan wildcard seperti % (untuk mengganti beberapa karakter) dan _ (untuk mengganti satu karakter). Contohnya:

   SELECT * FROM tabel WHERE kolom LIKE ‹nilai%›

Evaluasi / Soal Latihan

1. Apa perbedaan antara operator pembanding, operator logika, dan operator relasional dalam SQL Server? Berikan contoh penggunaannya.
2. Jelaskan perbedaan antara inner join, left join, dan right join pada SQL.
3. Bagaimana cara mengubah dan menghapus data dalam SQL Server? Jelaskan sintaks yang digunakan dan perbedaan antara keduanya.
4. Dalam suatu database, terdapat tabel “Customer” dengan kolom “customer_id”, “customer_name”, “address”, dan “phone_number”. Buatlah perintah SQL untuk menampilkan semua data dalam tabel “Customer”.
5. Dalam suatu database, terdapat tabel “Orders” dengan kolom “order_id”, “order_date”, “customer_id”, dan “total_amount”. Buatlah perintah SQL untuk menambahkan data baru ke dalam

tabel “Orders” dengan nilai “order_id” = 1001, “order_date” = ‘2022-04-01’, “customer_id” = 123, dan “total_amount” = 500000.

DAFTAR PUSTAKA

Connolly, T., & Begg, C. (2014). *Database systems: a practical approach to design, implementation, and management*. Pearson.

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

Davis, G. B., Olson, M. H., & Nielson, R. L. (1985). *Management Information Systems: An Introduction*. McGraw-Hill Book Company.

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems. Addison-Wesley*. Addison-Wesley.

Kendall, K. E., & Kendall, J. E. (2014). *Systems analysis and design*. Pearson.

Kroenke, D. M., & Auer, D. J. (2016). *Database concepts*. Pearson.

Laudon, K. C., & P., L. J. (2014). *Management Information Systems: Managing the Digital Firm* (13th ed.). Pearson.

O'Brien, J. A., & Marakas, G. M. (2018). *Management Information Systems* (10th ed.). McGraw-Hill Education.

Stair, R. M., & Reynolds, G. W. (2013). *Principles of Information Systems* (11th ed.). Cengage Learning.

Stair, Ralph M., & Reynolds, G. W. (2019). *Principles of Information Systems* (12th ed.). Cengage Learning.

BIOGRAFI PENULIS

Arie Gunawan, S.Kom., M.M.S.I

Lahir di kota Jakarta pada tanggal 10 April 1978. Penulis adalah Dosen di Fakultas Teknologi Komunikasi dan Informatika Program Studi Sistem Informasi Universitas Nasional. Menamatkan pendidikan program Sarjana (S1) di Universitas Gunadarma Jakarta prodi Sistem Informasi dan menyelesaikan program Pasca Sarjana (S2) di Universitas Gunadarma prodi Sistem Informasi. Saat ini penulis masih kuliah S3 di Universitas Pendidikan Bandung, Jawa Barat. Sebelum menjadi dosen pada tahun 2019, penulis sebelumnya adalah seorang praktisi yang bekerja di beberapa perusahaan, yaitu:

- PT Nusantara Surya Sakti sebagai MIS Dept Head
- PT Karunia Abadi Mandiri Persada sebagai IT Manager

- PT Semesta Finance sebagai IT Supervisor
- Penulis memiliki keahlian dalam bidang:
- Database: MS SQL Server 2005/2008/2014, MySQL
- Programming: Visual Basic 6.0, Visual Basic.Net, ASP.Net VB, Java NetBeans, Android Studio, C++, Python, HTML 5, Bootstrap, CSS, AJAX Toolkit
- ETL Tools: SSIS (SQL Server Integration Services), Pentaho, Talend, DTS (Data Transformation System)
- BI Tools: Kyubit, Tableau, PowerBI

Sari Ningsih, S.Si., MM

Lahir pada tanggal 2 Juni 1967 di Jakarta. Penulis merupakan dosen di Fakultas Teknologi Komunikasi dan Informatika Program Studi Sistem Informasi Universitas Nasional. Lulus program Sarjana (S1) Matematika Komputasi di FMIPA Universitas Indonesia dan Pasca Sarjana (S2) Ilmu Manajemen Keuangan Asuransi Universitas Gunadarma. Menjadi dosen sejak tahun 1991 sd sekarang tahun 2023, Pengalaman bekerja sebagai Kaprodi Manajemen Informatika Universitas Nasional, Ketua Unit Penjaminan Mutu FTKI Universitas Nasional dan cukup aktif mengajar dibeberapa kampus PTS di Jakarta

antara lain Universitas Gunadarma, Universitas Pancasila, Universitas Mercu buana dan beberapa PTS lainnya.

Penulis memiliki keahlian mengajar dalam bidang:

- Matematika: Aljabar Linier, Kalkulus, Statistika dan Probabilitas, Matematika Diskrit, dll
- Riset Operasional, Metode Numerik
- Data Mining

Dhieka Avrilia Lantana, S.Kom., M.Kom

Lahir pada tanggal 12 April 1989 di Jakarta. Penulis merupakan dosen di Fakultas Teknologi Komunikasi dan Informatika Program Studi Informatika Universitas Nasional. Lulus program Sarjana (S1) dan Pasca Sarjana (S2) di Institut Pertanian Bogor (IPB University) program studi Ilmu Komputer. Sebelum menjadi dosen pada tahun 2019, penulis sebelumnya bekerja di beberapa perusahan, yaitu:

- Accenture Solutions Sdn. Bhd. sebagai Senior Account Optimizer
- PT Kreatif Media Karya (KMK) sebagai Search Engine Optimization (SEO)
- Badan Pusat Sistem Informasi (BPSI) Universitas Nasional sebagai programmer

Penulis memiliki keahlian dalam bidang:

- Artificial Intelligence
- Data Mining
- Programming language: PHP, java, Python, CSS, HTML, C++
- Digital Marketing: Google Ads, Facebook Ads, Instagram Ads