DATABASE

Penulis :

- Ahmad Jurnaidi Wahidin
- Nisa Miftachurohmah
- Daniel Alfa Puryono
- Hanes
- Rajif Agung Yunmar
- Salsalina Br Sembiring
- Hasanudin Jayawardana
- Ridho Surya Kusuma
- Hilman Nuril Hadi
- Irpan Adiputra Pardosi
- Jarot Budiasto
- Tri Kustanti Rahayu

Gpress

DATABASE

Ahmad Jurnaidi Wahidin
Nisa Miftachurohmah
Daniel Alfa Puryono
Hanes
Rajif Agung Yunmar
Salsalina Br Sembiring
Hasanudin Jayawardana
Ridho Surya Kusuma
Hilman Nuril Hadi
Irpan Adiputra Pardosi
Jarot Budiasto
Tri Kustanti Rahayu

GET PRESS INDONESIA

DATABASE

Penulis :
Ahmad Jurnaidi Wahidin
Nisa Miftachurohmah
Daniel Alfa Puryono
Hanes
Rajif Agung Yunmar
Salsalina Br Sembiring
Hasanudin Jayawardana
Ridho Surya Kusuma
Hilman Nuril Hadi
Irpan Adiputra Pardosi
Jarot Budiasto
Tri Kustanti Rahayu

ISBN: 978-623-125-632-4

Editor : DR., D.Sc., Drs., Sunarno Sastro Atmodjo, S.E. S.T., S.AP., S.IP., S.Sos
Penyunting : Tri Putri Wahyuni. S.Pd
Desain Sampul dan Tata Letak : Atyka Trianisa, S.Pd

Penerbit : GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jln. Palarik Air Pacah No 26 Kel. Air Pacah
Kec. Koto Tangah Kota Padang Sumatera Barat
Website : www. getpress.co.id
Email : adm.getpress@gmail.com

Cetakan pertama, Februari 2025
Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk dan
dengan cara apapun tanpa izin tertulis dari penerbit.

KATA PENGANTAR

Segala Puji dan syukur atas kehadirat Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan Buku Database ini.

Buku Ini Membahas Introduction Database, Introduction to the Relational Model, Introduction to SQL, Intermediate SQL, Advanced SQL, Database Design Using The E-R Model and Normalization, Relational Database Design and Normalization, Complex Data Types, Application Development, Introduction Big Data, Data Analysis, Information Retrieval.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Padang, Januari 2025

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
INTRODUCTION DATABASE

Oleh Ahmad Jurnaidi Wahidin

1.1 Pendahuluan

Dalam era digital saat ini, data menjadi aset paling berharga bagi individu maupun organisasi. Data dapat membantu dalam pengambilan keputusan, analisis bisnis, dan meningkatkan efisiensi operasional. Basis data atau database adalah istilah dalam teknologi jaringan komputer yang berfungsi untuk menyimpan berbagai jenis data dan memiliki beragam manfaat, teknologi utama yang memungkinkan pengelolaan data secara efisien. Dengan database, informasi dapat disimpan, dikelola, dan diakses dengan mudah, mendukung berbagai aktivitas mulai dari transaksi sehari-hari hingga analisis skala besar.

Database telah berkembang pesat sejak pertama kali diperkenalkan, beradaptasi dengan kebutuhan teknologi dan bisnis yang terus berubah. Perkembangan ini mencakup pengenalan model data baru, teknologi penyimpanan, dan metode akses data yang lebih efisien.

1.2 Definisi Database

Database atau yang dikenal sebagai basis data, berasal dari gabungan kata "basis" dan "data". Data merujuk pada catatan yang terdiri dari kumpulan fakta yang merepresentasikan suatu objek, bersifat mentah, dan tidak memiliki konteks. Di sisi lain, istilah "basis" dapat diartikan sebagai pusat, tempat berkumpulnya objek, atau representasi dari objek tersebut (Ihksan, Susilo and Abdillah, 2023).

Database merupakan kumpulan data yang terorganisir, saling berhubungan, dan disimpan secara elektronik untuk mendukung pengambilan keputusan dan memudahkan proses pengelolaannya. Dengan pengelolaan ini, pengguna dapat dengan mudah mencari, menyimpan, dan menghapus informasi sesuai kebutuhan.

Berbicara mengenai database, terdapat istilah dasar yang dikenal sebagai data. Pada awalnya, data merujuk pada fakta yang dapat dicatat dan disimpan di media komputer, seperti hard disk. Contohnya adalah nama, alamat, atau kota tempat tinggal seorang pelanggan yang merepresentasikan sebuah data. Namun, penting untuk dipahami bahwa data saat ini tidak hanya terbatas pada teks seperti itu, tetapi juga mencakup dokumen, gambar, suara, hingga cuplikan video(Kadir, 2020).

Database adalah perangkat yang dirancang untuk mengumpulkan dan mengelola informasi secara terorganisasi. Database dapat menyimpan data terkait individu, produk, pesanan, atau berbagai jenis informasi lainnya. Awalnya, database sering dimulai sebagai daftar sederhana dalam program pengolah kata atau lembar kerja. Namun, seiring bertambahnya ukuran daftar, masalah seperti duplikasi dan inkonsistensi data mulai muncul, membuat informasi sulit untuk dipahami. Selain itu, kemampuan untuk mencari atau mengambil subset data menjadi terbatas. Ketika tantangan ini terjadi, langkah yang tepat adalah memindahkan data ke sistem database yang dikelola oleh perangkat lunak manajemen database (DBMS).

Sistem database biasanya menggunakan perangkat lunak manajemen database (*Database Management System* atau DBMS) untuk mengatur, menyimpan, dan memanipulasi data. DBMS seperti MySQL, PostgreSQL, Oracle Database, dan Microsoft SQL Server adalah contoh perangkat lunak yang sering digunakan.

1.3 Tujuan dan Manfaat Database

Database dirancang untuk mencapai tujuan tertentu yang memberikan manfaat besar bagi pengguna. Berikut ini adalah penjelasan lebih rinci:

1. Menghindari Duplikasi Data

   Database memastikan bahwa data tidak disimpan berulang kali. Misalnya, informasi pelanggan hanya perlu dimasukkan satu kali dan dapat diakses oleh berbagai departemen. Ini mengurangi redundansi dan meningkatkan efisiensi penyimpanan.

2. Konsistensi dan Keakuratan

   Dengan pengelolaan yang baik, database memastikan data yang disimpan tetap konsisten dan akurat, mengurangi risiko kesalahan informasi. Misalnya, perubahan pada satu bagian data otomatis diperbarui di semua tempat yang relevan, sehingga mencegah data yang saling bertentangan.
3. Kemudahan Akses dan Modifikasi

   Database memungkinkan pengguna untuk mencari informasi dengan cepat menggunakan query. Misalnya, SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengakses dan memanipulasi data dalam database relasional. Database menawarkan kemudahan dan efisiensi dalam menyaring data serta informasi. Namun, penting untuk diingat bahwa setiap database memiliki kemampuan dan kecepatan pemrosesan yang berbeda-beda.
4. Keamanan Data

   Sistem database dilengkapi dengan kontrol akses yang membatasi siapa saja yang dapat melihat atau mengubah data tertentu, melindungi informasi dari akses yang tidak sah. Dengan ini, hanya pengguna yang memiliki izin yang dapat mengakses atau memodifikasi data tertentu, melindungi data dari akses yang tidak sah.
5. Dukungan Pengambilan Keputusan

   Database menyediakan informasi yang terorganisir dan siap digunakan untuk analisis, sehingga membantu organisasi dalam membuat keputusan strategis. Dengan data yang relevan dan terkini, organisasi dapat membuat keputusan strategis yang lebih baik.
6. Efisiensi Penyimpanan

   Database dirancang untuk menyimpan data dengan cara yang efisien, menghindari duplikasi dan memaksimalkan penggunaan ruang penyimpanan. Dengan struktur yang terorganisir, data hanya disimpan sekali dan dapat digunakan oleh banyak aplikasi atau pengguna tanpa duplikasi.
7. Fleksibilitas dan Skalabilitas

   Database modern memungkinkan penyesuaian struktur dan volume data sesuai dengan kebutuhan. Hal ini penting

untuk perusahaan yang terus berkembang atau menghadapi perubahan kebutuhan data.

8. Penghematan Biaya

   Dengan pengelolaan data yang efisien, database membantu mengurangi biaya operasional. Data yang terpusat mengurangi kebutuhan akan pengolahan manual, menghemat waktu dan sumber daya manusia. Database membutuhkan hanya satu unit utama untuk menyimpan seluruh data. Hal ini memungkinkan perusahaan menghindari penggunaan beberapa unit penyimpanan terpisah, sehingga dapat mengurangi biaya operasional.

9. Multi-User

   Database memungkinkan beberapa pengguna untuk mengaksesnya secara bersamaan. Sementara itu, penyimpanan data terpusat disimpan dalam satu sistem atau unit utama.

1.4 Komponen Sistem Database

Sistem database terdiri dari beberapa komponen utama yang saling mendukung untuk memastikan database dapat berfungsi secara optimal. Berikut adalah penjelasan tentang masing-masing komponen(Hariyono *et al.*, 2023) (Idhom, 2024):

1. Pengguna (User)

   Pengguna atau user adalah individu atau kelompok yang memiliki otoritas untuk berinteraksi dengan sistem database. Mereka membutuhkan informasi yang disediakan dalam database. Pengguna dapat diklasifikasikan ke dalam beberapa kategori:

   a. Programmer Aplikasi (*Application Programmer*)

      Pengguna yang bertugas membuat program aplikasi untuk mengakses database menggunakan bahasa pemrograman tertentu.

   b. End User

      Pengguna akhir yang berinteraksi dengan sistem database berdasarkan kebutuhan mereka. Mereka dapat dikelompokkan menjadi pengguna biasa (*naive user*), pengguna kasual (*casual user*), dan pengguna khusus (*specialized user*).

c. Desainer (*Designer*)

Mereka bertanggung jawab atas desain struktur database, memastikan data diatur dengan baik untuk memenuhi kebutuhan organisasi.

d. Administrator (*Database Administrator*)

Individu atau tim yang mengelola keseluruhan sistem database, termasuk keamanan, integritas data, dan pemeliharaan operasional.

2. Data

Data merupakan elemen inti dari sistem database. Data disusun dalam bentuk tabel atau file yang menyimpan informasi terkait entitas dan atribut.

a. Entitas

Merujuk pada objek, orang, tempat, atau kejadian yang informasinya disimpan di database.

b. Atribut

Karakteristik atau informasi yang dimiliki oleh entitas. Sebagai contoh, seorang mahasiswa memiliki atribut seperti nama, alamat, nomor induk, dan jenis kelamin.

Data dari berbagai entitas disusun dan diorganisasi menjadi file yang kemudian digabungkan untuk membentuk sistem informasi yang lengkap.

3. Software DBMS (*Database Management System*)

DBMS adalah perangkat lunak yang digunakan untuk mengelola database. Fungsinya mencakup:

a. Memelihara data agar tetap terstruktur.
b. Memudahkan akses dan manipulasi data oleh pengguna.
c. Mengimplementasikan mekanisme keamanan dan pengendalian data.

Beberapa contoh DBMS yang populer adalah MySQL, PostgreSQL, Microsoft SQL Server, Oracle, dan MongoDB.

4. Sistem Operasi (*Operating System*)

   Sistem operasi adalah perangkat lunak yang mengelola sumber daya perangkat keras dan memungkinkan aplikasi database berfungsi. Sistem operasi yang kompatibel dengan perangkat lunak database sangat penting untuk mendukung kinerja sistem secara keseluruhan. Contoh sistem operasi yang sering digunakan meliputi Windows, Linux, Unix, dan MacOS.

5. Perangkat Keras (*Hardware*)

   Perangkat keras mencakup semua komponen fisik yang diperlukan untuk menjalankan sistem database. Beberapa perangkat keras yang umum digunakan adalah:

   a. Komputer
      Berfungsi sebagai pusat pengolahan data.
   b. Memori Sekunder
      Termasuk hard disk untuk penyimpanan data secara online dan perangkat seperti USB storage untuk penyimpanan offline.
   c. Media Komunikasi
      Mendukung koneksi jaringan untuk sistem database yang diakses secara terdistribusi.

6. Software Pendukung Lainnya

   Selain DBMS, terdapat perangkat lunak pendukung lain yang bersifat opsional. Software ini membantu pengguna dalam menghasilkan laporan, memvisualisasikan data, atau menjalankan fungsi tambahan. Contoh software pendukung meliputi Microsoft SQL Server, Oracle, XBase, dan MySQL.

1.5 Sejarah Perkembangan Database

Sejak peradaban kuno, manusia telah berupaya menyimpan informasi untuk membantu pengelolaan sumber daya dan pengambilan keputusan. Teknik akuntansi yang digunakan oleh bangsa Mesir dan Sumeria menjadi salah satu bukti awal upaya pengorganisasian data. Meski metode ini sederhana, mereka menjadi fondasi penting bagi perkembangan sistem manajemen data yang kita kenal saat ini.

Pada era sebelum teknologi komputer berkembang, sistem penyimpanan data dilakukan secara manual melalui dokumen fisik seperti arsip dan katalog. Lembaga pemerintah, perpustakaan, rumah sakit, dan bisnis menggunakan sistem ini untuk melacak informasi. Meskipun sederhana, prinsip dasar yang digunakan dalam sistem manual ini tetap relevan hingga era database modern, khususnya dalam hal pengorganisasian dan aksesibilitas data.

1. Awal Mula Database Digital

   Peralihan ke database digital dimulai pada tahun 1960-an dengan kemunculan komputer sebagai alat yang lebih efisien dan hemat biaya untuk mengelola informasi. Model jaringan dan model hierarkis menjadi dua pendekatan utama dalam manajemen data pada era ini. Salah satu sistem yang sukses adalah SABRE, sebuah sistem reservasi yang dikembangkan oleh IBM untuk American Airlines. SABRE membuktikan bahwa database terkomputerisasi mampu menangani kebutuhan pengelolaan informasi yang kompleks.

   Model jaringan, seperti yang diusung oleh CODASYL, menggunakan pendekatan hubungan antar data yang menyerupai jaringan. Di sisi lain, model hierarkis, seperti *Information Management System* (IMS) yang dikembangkan oleh IBM, menggunakan struktur pohon untuk mengatur data dalam hubungan satu-ke-banyak. Kedua model ini menjadi tonggak awal dalam teknologi database digital.

2. Pengenalan Database Relasional

   Pada tahun 1970, Edgar F. Codd memperkenalkan model relasional, sebuah pendekatan revolusioner yang memungkinkan data disimpan dalam tabel-tabel yang saling terkait. Model ini memungkinkan aplikasi mencari data berdasarkan isi tabel, bukan melalui navigasi kompleks seperti pada model jaringan atau hierarkis. Konsep ini menjadi dasar bagi pengembangan *Structured Query Language* (SQL), bahasa standar untuk pengelolaan database relasional.

   Pada tahun 1980-an, SQL secara resmi diakui sebagai standar industri oleh *American National Standards Institute* (ANSI). Dengan standardisasi ini, sistem database relasional mulai diadopsi secara luas. Perusahaan seperti Oracle, IBM

dengan Db2, Microsoft dengan SQL Server, serta MySQL menjadi pemain utama dalam menyediakan solusi database relasional yang andal.

3. Perkembangan Database NoSQL dan Big Data

   Memasuki era 2000-an, kebutuhan akan pengelolaan data yang tidak terstruktur mendorong munculnya teknologi NoSQL. Tidak seperti database relasional, NoSQL menawarkan fleksibilitas dalam menangani data yang kompleks dan bervariasi, seperti data media sosial, log sistem, dan informasi dari sensor IoT. Teknologi seperti MongoDB, Couchbase, dan Cassandra menjadi populer karena kemampuannya menangani skala besar dan data yang tersebar di berbagai lokasi.

   Era ini juga ditandai oleh kebangkitan Big Data, di mana volume data yang dihasilkan meningkat secara eksponensial. Teknologi seperti Apache Hadoop dan Apache Spark memberikan solusi untuk memproses data dalam jumlah besar secara paralel, memungkinkan analisis data dalam skala yang sebelumnya tidak mungkin.

4. Database di Era Modern

   Saat ini, database memainkan peran penting dalam berbagai aspek kehidupan, mulai dari layanan penyimpanan awan hingga pengembangan kecerdasan buatan (AI). Sistem database modern dirancang untuk mendukung kebutuhan real-time dan integrasi dengan teknologi analitik. Database grafik, seperti Neo4j dan Amazon Neptune, memungkinkan analisis hubungan antar data yang kompleks, sangat berguna untuk aplikasi seperti deteksi penipuan dan analisis jaringan sosial.

   Selain itu, teknologi cloud computing telah mengubah cara organisasi menyimpan dan mengakses data. Penyedia layanan seperti *Amazon Web Services* (AWS), Google Cloud, dan Microsoft Azure menawarkan solusi database yang fleksibel dan skalabel, memungkinkan perusahaan berfokus pada inovasi tanpa perlu khawatir tentang infrastruktur.

5. Masa Depan Database

   Dengan kemajuan teknologi, database terus beradaptasi untuk memenuhi kebutuhan yang semakin kompleks. Tren masa depan meliputi pengembangan database kuantum yang

memanfaatkan komputasi kuantum untuk memproses data dengan kecepatan luar biasa. Selain itu, konsep database otonom, seperti Oracle Autonomous Database, menunjukkan potensi untuk mengelola dan mengoptimalkan sistem secara mandiri tanpa intervensi manusia.

Teknologi database akan terus menjadi tulang punggung dalam pengelolaan informasi di berbagai sektor, mendukung inovasi di bidang AI, IoT, dan analitik data. Dengan perkembangan ini, masa depan database tampak sangat cerah, membuka peluang baru untuk pengelolaan dan pemanfaatan data dalam skala global.

1.6 Penerapan Database dalam Kehidupan Sehari-hari

Kehadirannya dalam berbagai aspek kehidupan telah memberikan banyak manfaat yang tidak hanya mempermudah pekerjaan manusia tetapi juga meningkatkan efisiensi dan kecepatan dalam pengambilan keputusan. Berikut adalah pembahasan tentang penerapan database dalam kehidupan sehari-hari (Liquid Web, 2021) (omics, 2022):

1. Streaming Video

   Platform seperti YouTube dan Netflix menggunakan database untuk menyimpan informasi tentang konten, riwayat tontonan pengguna, dan preferensi mereka. Teknologi ini memungkinkan sistem memberikan rekomendasi tontonan yang disesuaikan dengan selera pengguna. YouTube, misalnya, menggunakan MySQL yang didukung oleh Vitess untuk penskalaan horizontal, serta memanfaatkan Memcache dan Zookeeper untuk caching dan koordinasi node.

2. Gaming Online

   Banyak permainan online mengandalkan database untuk menyimpan informasi pemain, seperti profil, progres, dan interaksi dengan pemain lain. Contohnya adalah Minecraft, yang memungkinkan pengguna menjadi host server mereka sendiri. Database memungkinkan pengumpulan dan pengelolaan data ini agar pemain dapat berinteraksi secara real-time.

3. Penyimpanan Cloud

   Layanan cloud seperti Google Drive, Microsoft OneDrive, dan Dropbox menggunakan database untuk menyimpan data pengguna secara aman dan terorganisasi. Sistem ini memungkinkan pengguna mengakses data mereka kapan saja, di mana saja, dengan model penyimpanan yang fleksibel dan andal.

4. Media Sosial

   Platform media sosial seperti Facebook, Instagram, dan Twitter menggunakan database besar untuk menyimpan informasi pengguna, seperti daftar teman, preferensi, hingga interaksi. Data ini mendukung fitur rekomendasi, seperti teman yang mungkin Anda kenal atau topik yang sedang tren.

5. e-Commerce

   Marketplace seperti Tokopedia dan Shopee memanfaatkan database untuk mengelola produk, harga, informasi pelanggan, serta riwayat pembelian. Dengan database, penjual dapat membuat rekomendasi produk yang lebih personal kepada pelanggan, sementara pelanggan menikmati pengalaman berbelanja yang lebih aman dan efisien.

6. Prediksi Cuaca

   Lembaga prakiraan cuaca, seperti The Weather Company, menggunakan database untuk menyimpan dan menganalisis data cuaca. Dengan volume data hingga 20 TB per hari, database memungkinkan pengolahan data ini untuk memberikan prediksi cuaca yang akurat.

7. Pemerintahan dan Organisasi

   Pemerintah desa, sekolah, rumah sakit, dan organisasi lainnya menggunakan database untuk mengelola informasi masyarakat, seperti data kependudukan, program sosial, dan layanan publik. Database membantu meningkatkan efisiensi administrasi, transparansi, dan akurasi dalam pengambilan keputusan.

8. Perbankan

   Sistem database digunakan untuk menyimpan informasi rekening nasabah, catatan transaksi, dan layanan keuangan

lainnya. Bank juga menggunakan database untuk mengelola pinjaman, investasi, serta mengamankan data sensitif nasabah dari akses yang tidak sah.

9. Pendidikan

   Database digunakan untuk mengelola informasi mahasiswa, jadwal kelas, dan catatan nilai. Lembaga pendidikan seperti universitas dan sekolah memanfaatkan database untuk memastikan informasi akademik dan administrasi tersimpan secara terstruktur dan mudah diakses.

10. Kesehatan

    Rumah sakit dan klinik menggunakan database untuk menyimpan informasi pasien, riwayat medis, dan jadwal konsultasi. Database juga memungkinkan integrasi dengan sistem asuransi kesehatan untuk mempermudah proses klaim dan pembayaran.

11. Transportasi

    Sistem transportasi publik seperti kereta api dan maskapai penerbangan menggunakan database untuk mengelola jadwal perjalanan, reservasi tiket, dan pelacakan bagasi. Data ini juga digunakan untuk memantau operasional kendaraan secara real-time.

12. Industri Logistik

    Perusahaan logistik menggunakan database untuk melacak pengiriman barang, mengelola inventaris, dan memantau rute pengiriman. Database memungkinkan proses logistik berjalan lebih efisien dan akurat.

DAFTAR PUSTAKA

Hariyono, R. C. S. *et al.* (2023) *Buku Ajar Pengantar Database*. Jambi: PT. Sonpedia Publishing Indonesia.

Idhom, M. (2024) *Database*. Jawa Timur: Thalibul Ilmi Publishing & Education.

Ihksan, M., Susilo, H. and Abdillah, N. (2023) *DATABASE 2023: Konsep Dasar Membangun Database*. Sumatera Barat: Suluah Kato Khatulistiwa.

Kadir, A. (2020) *Dasar Perancangan dan Implementasi Database Relasional (Edisi Revisi)*. 1st edn. Yogyakarta: Andi Offset.

Liquid Web (2021) *10 Database Examples in Real Life*, *liquidweb.com*. Available at: https://www.liquidweb.com/blog/ten-ways-databases-run-your-life/ (Accessed: 20 January 2025).

omics (2022) *10 Database Examples in Real Life*, *omicstutorials.com*. Available at: https://omicstutorials.com/10-database-examples-in-real-life/ (Accessed: 20 January 2025).

BAB 2
INTRODUCTION TO RELATIONAL DATABASE MODEL

Oleh Nisa Miftachurohmah

2.1 Model Basis Data Relasional

Model basis data relasional adalah pendekatan untuk mengorganisir data dalam bentuk tabel yang saling berhubungan. Setiap tabel terdiri dari baris dan kolom, di mana setiap baris mewakili satu entitas dan setiap kolom mewakili atribut dari entitas tersebut. Konsep ini diperkenalkan oleh Edgar F. Codd pada tahun 1970, yang menekankan pentingnya integritas data dan kemudahan akses. Dengan menggunakan model ini, pengguna dapat dengan mudah melakukan operasi seperti pencarian, penyisipan, pembaruan, dan penghapusan data. Hal ini menjadikan model relasional sangat populer dalam pengembangan sistem basis data modern.

Salah satu keunggulan utama dari model basis data relasional adalah kemampuannya untuk mengurangi redundansi data. Dengan memisahkan data ke dalam tabel yang berbeda dan menggunakan kunci untuk menghubungkan tabel-tabel tersebut, model ini memungkinkan penyimpanan data yang lebih efisien. Misalnya, informasi tentang karyawan dan departemen dapat disimpan dalam tabel terpisah, dan hubungan antara keduanya dapat ditentukan melalui kunci asing. Ini tidak hanya menghemat ruang penyimpanan tetapi juga memudahkan pemeliharaan data. Ketika data perlu diperbarui, perubahan hanya perlu dilakukan di satu tempat, mengurangi kemungkinan inkonsistensi.

Model basis data relasional juga menawarkan fleksibilitas dalam hal pengambilan data. Pengguna dapat menggunakan *Structured Query Language* (SQL) untuk melakukan *query* yang kompleks, memungkinkan mereka untuk mengambil data dari

beberapa tabel sekaligus. Dengan kemampuan untuk melakukan *join* antara tabel, pengguna dapat menggabungkan informasi dari berbagai sumber untuk analisis yang lebih mendalam. Ini sangat berguna dalam aplikasi bisnis di mana data sering kali tersebar di berbagai tabel. Fleksibilitas ini menjadikan model relasional sangat cocok untuk aplikasi yang memerlukan analisis data yang kompleks.

Selain itu, model basis data relasional mendukung integritas data melalui penggunaan *constraint*. *Constraint* seperti *primary key*, *foreign key*, dan *unique constraints* membantu memastikan bahwa data yang dimasukkan ke dalam basis data memenuhi kriteria tertentu. Ini membantu mencegah kesalahan dan inkonsistensi dalam data, yang sangat penting dalam aplikasi yang memerlukan akurasi tinggi. Dengan adanya *constraint*, pengguna dapat yakin bahwa data yang mereka kelola adalah valid dan dapat diandalkan. Ini juga memudahkan pengelolaan data dalam jangka panjang.

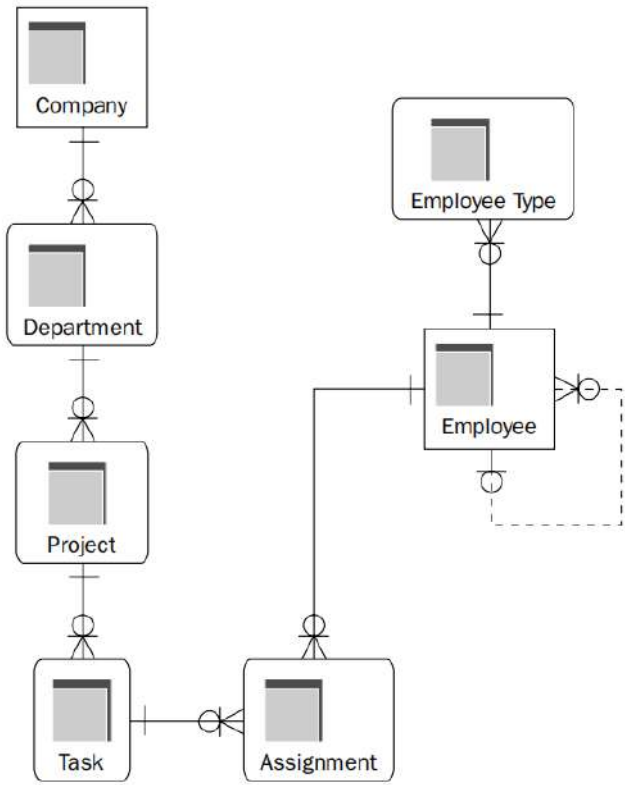
Akhirnya, model basis data relasional memiliki dukungan yang luas dari berbagai sistem manajemen basis data (DBMS) yang ada di pasaran. Banyak DBMS populer seperti MySQL, PostgreSQL, dan Oracle *Database* mengimplementasikan model ini, memberikan pengguna berbagai pilihan untuk kebutuhan spesifik mereka. Dukungan yang luas ini juga berarti bahwa ada banyak sumber daya dan komunitas yang tersedia untuk membantu pengguna dalam mengatasi masalah yang mungkin mereka hadapi. Dengan demikian, model basis data relasional tidak hanya menawarkan struktur yang kuat untuk pengelolaan data, tetapi juga ekosistem yang kaya untuk pengembangan dan pemeliharaan sistem basis data.

2.2 Struktur Basis Data Relational

Model basis data relasional menyempurnakan pembatasan struktur hierarki, tidak sepenuhnya mengabaikan hierarki data, seperti yang ditunjukkan pada **Gambar 2.1**. Tabel apa pun dapat diakses secara langsung tanpa harus mengakses semua objek induk. Triknya adalah mengetahui apa yang harus dicari — jika Anda ingin menemukan alamat karyawan tertentu, Anda harus mengetahui

karyawan mana yang harus dicari, atau Anda dapat memeriksa semua karyawan. Anda tidak perlu mencari seluruh hierarki, dari perusahaan ke bawah, untuk menemukan satu karyawan.

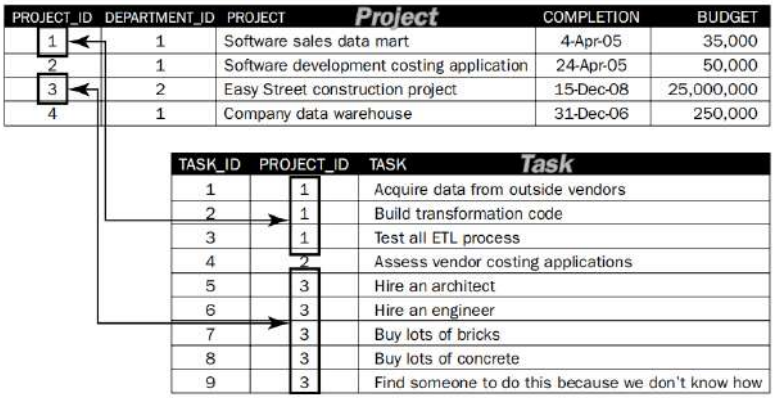
Manfaat lain dari model basis data relasional adalah tabel apa pun dapat ditautkan bersama, terlepas dari posisi hierarkinya. Jelas, harus ada tautan yang masuk akal antara kedua tabel, tetapi Anda tidak dibatasi oleh struktur hierarki yang ketat; oleh karena itu, tabel dapat ditautkan ke sejumlah tabel induk dan sejumlah tabel anak.



Gambar 2.1. Model Relasi Basis Data

Gambar 2.2 menunjukkan contoh bagian kecil dari model basis data relasional yang ditunjukkan pada **Gambar 2.1**. Tabel yang ditampilkan adalah tabel ***Project*** dan ***Task***. Kolom PROJECT_ID

pada tabel ***Project*** mengidentifikasi setiap proyek secara unik dalam tabel ***Project***. Hubungan antara tabel ***Project*** dan ***Task*** adalah hubungan satu ke banyak menggunakan kolom PROJECT_ID, yang diduplikasi dari tabel ***Project*** ke tabel ***Task***. Seperti yang dapat dilihat pada **Gambar 2.2**, tiga entri pertama dalam tabel ***Task*** semuanya merupakan bagian dari proyek *Software sales data mart*.



Gambar 2.2. Sajian Data pada Model Relasi Basis Data

Sebagai contoh lain, dalam sebuah kasus pada **Gambar 2.3,** yang menyimpan informasi tentang tabel ***instructor***. Tabel tersebut memiliki empat kolom: **ID**, ***name***, ***dept_name***, dan ***salary***. Tabel ***course*** pada **Gambar 2.4** menyimpan informasi tentang ***course***, yang terdiri dari ***course_id***, ***title***, ***dept_name***, dan ***credits***.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Gambar 2.3. *Instructor*

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Gambar 2.4. *Course*

Gambar 2.5 menunjukkan tabel ***prereq***. Tabel tersebut memiliki dua kolom, ***course_id*** dan ***prereq_id***. Setiap baris terdiri dari sepasang pengenal ***course*** sehingga ***course*** kedua merupakan prasyarat untuk ***course*** pertama.

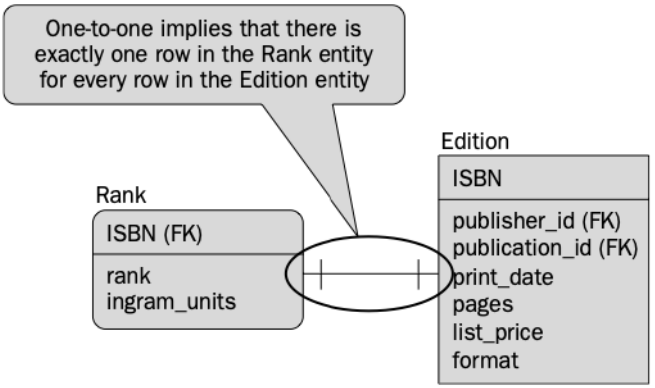
course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Gambar 2.5. *Prereq*

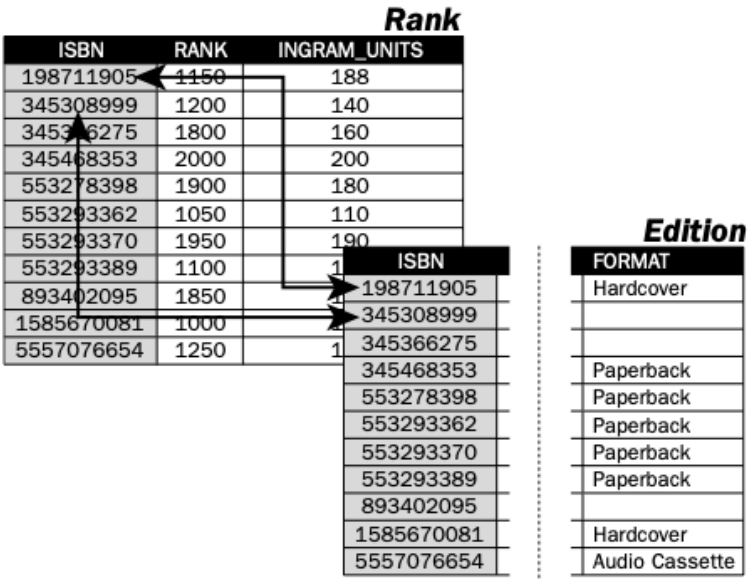
2.3 Kardinalitas Relasi Basis Data

2.3.1 *One-to-one*

Gambar 2.6 memperlihatkan hubungan satu-satu antara **tabel EDITION** dan **RANK**, sehingga untuk setiap entri **EDITION**, terdapat tepat satu entri **RANK**, dan sebaliknya. **Gambar 2.7** memperlihatkan padanan representasi struktur tabel *one-to-one* dengan menggunakan data riil.



Gambar 2.6. Relasi *one to one* antara 2 entitas

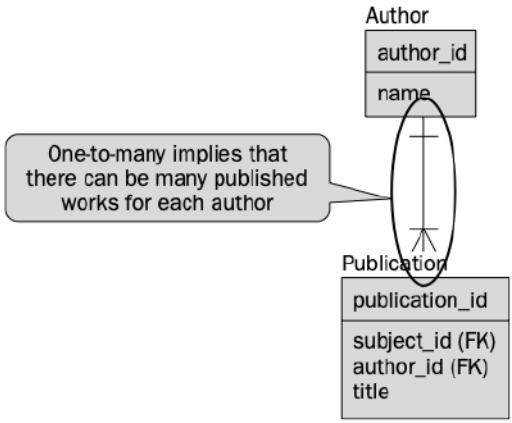


Gambar 2.7. Relasi *one to one* antara 2 tabel

2.3.2 *One-to-many*

Hubungan satu ke banyak sangat umum dalam model basis data relasional antar tabel. **Gambar 2.8** memperlihatkan bahwa rekaman tabel **AUTHOR** dapat memiliki banyak publikasi karena seorang penulis dapat menerbitkan banyak buku (entri rekaman **PUBLICATION**). **Gambar 2.9** menunjukkan diagram data dengan **AUTHOR** di sebelah kanan diagram dan

PUBLICATION masing-masing di sebelah kiri, dalam hubungan *one-to-many*.



Gambar 2.8. Relasi *one to many* antara 2 entitas

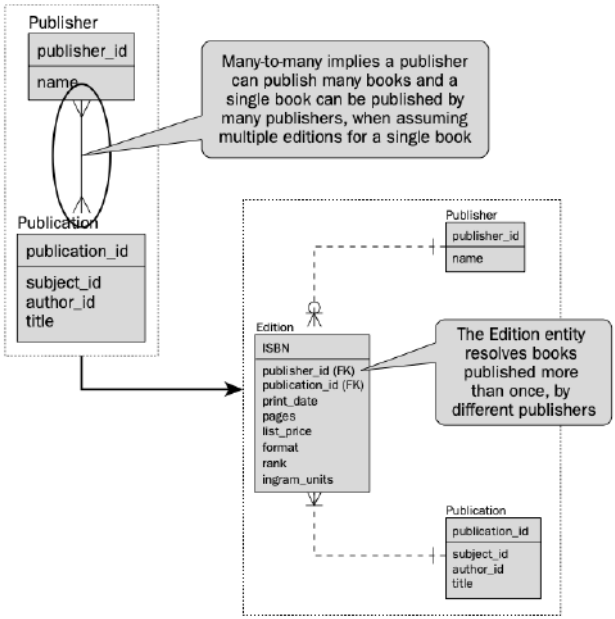
PUBLICATION_ID	SUBJECT_ID	AUTHOR_ID	TITLE
1	7	2	Cities in Flight
2	7	2	A Case of Conscience
3	7	3	Foundation
4	7	3	Second Foundation
5	7	3	Foundation and Em
6	7	3	Foundation's Edge
7	7	3	Prelude to Foundat
9	7	4	Lucifer's Hammer
10	7	4	Footfall
11	7	4	Ringworld
8	15	6	The Complete Works of Shakespeare
12	16	7	Hocus Pocus

AUTHOR_ID	NAME
1	Orson Scott Card
2	James Blish
3	Isaac Azimov
4	Larry Niven
5	Jerry Pournelle
6	William Shakespeare
7	Kurt Vonnegut

Gambar 2.9. Relasi *one to many* antara 2 tabel

2.3.3 *Many-to-many*

Hubungan banyak ke banyak artinya untuk setiap satu rekaman pada satu tabel, ada banyak kemungkinan rekaman pada tabel terkait lainnya, begitu pula sebaliknya (untuk kedua tabel). Pada **Gambar 2.10**, dari kiri ke kanan, hubungan *many-to-many* antara tabel **PUBLISHER** dan **PUBLICATION** diubah menjadi tabel **EDITION**. Sebuah **PUBLISHER** dapat menerbitkan banyak **PUBLICATION** dan satu **PUBLICATION** dapat diterbitkan oleh banyak **PUBLISHER**. Pada **Gambar 2.11** terdapat tujuh edisi berbeda dari **PUBLICATION** Foundation.



Gambar 2.10. Relasi *many to many* antara 2 entitas

TITLE	PUBLISHER	ISBN	PRINTED
Cities in Flight	Overlook Press	1585670081	
A Case of Conscience	Ballantine Books	345438353	
Foundation	**HarperCollins Publishing**	**246118318**	28-A
Foundation	**Books on Tape**	**5553673224**	31-Jan-2
Foundation	**Books on Tape**	**5557076654**	31-Jan-51
Foundation	**Del Rey Books**	**345334787**	31-Dec-85
Foundation	**Del Rey Books**	**345308999**	28-Feb-83
Foundation	**L P Books**	**893402095**	31-May-79
Foundation	**Ballantine Books**	**345336275**	31-Jul-86
Second Foundation	Bantam Books	553293362	
Foundation and Empire	Spectra	553293370	
Foundation's Edge	Spectra	553293389	
Prelude to Foundation	Spectra	553298398	
Lucifer's Hammer	Fawcett Books	449208133	31-May-85
Footfall	Del Rey Books	345323440	31-Jul-96
Ringworld	Ballantine Books	345333926	30-Nov-90

Each edition is uniquely identified by ISBN – unique to each new edition of the same title

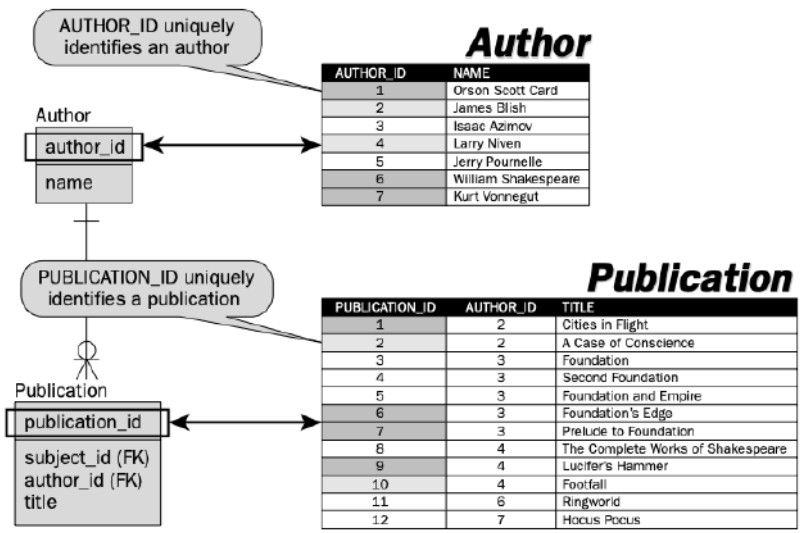
Gambar 2.11. Relasi *many to many* pada tabel

2.4 Keys

2.4.1 *Primary Key*

Primary key digunakan untuk mengidentifikasi secara unik suatu rekaman dalam suatu tabel. Identifikasi unik untuk setiap rekaman diperlukan karena tidak ada cara lain untuk

menemukan rekaman tanpa kemungkinan menemukan lebih dari satu rekaman, jika pengenal unik tidak digunakan. G**ambar 2.12** menunjukkan bidang kunci utama **AUTHOR_ID** untuk tabel **AUTHOR** dan **PUBLICATION_ID** untuk tabel **PUBLICATION**, masing-masing merupakan bidang *primary key* untuk kedua tabel tersebut.



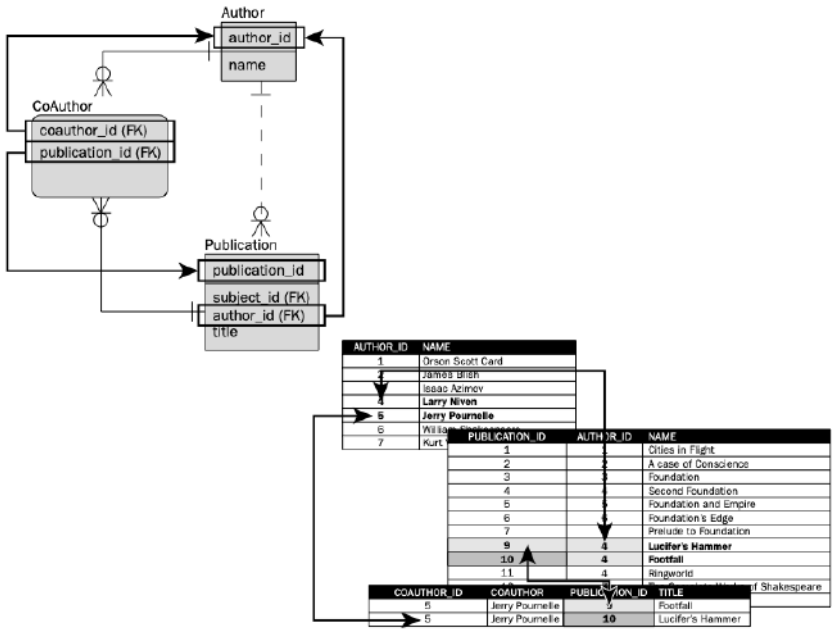
Gambar 2.12. *Primary key* pada ER-D dan Tabel

2.4.2 *Foreign Key*

Foreign key adalah salinan kunci utama yang dibuat dalam tabel anak untuk membentuk sisi berlawanan dari tautan dalam hubungan antar tabel — membangun hubungan basis data relasional. *Foreign key* mendefinisikan referensi untuk setiap rekaman dalam tabel anak, merujuk kembali ke kunci utama dalam tabel induk.

Gambar 2.13 menunjukkan bahwa tabel **PUBLICATION** memiliki kunci asing yang disebut **AUTHOR_ID (FK)**. Ini berarti bahwa setiap rekaman dalam tabel **PUBLICATION** memiliki salinan nilai bidang **AUTHOR_ID** tabel induk, nilai *primary key* tabel **AUTHOR**, dalam bidang *foreign key* **AUTHOR_ID** pada tabel **PUBLICATION**. Dengan kata lain, seorang penulis dapat memiliki banyak buku yang diterbitkan dan tersedia untuk dijual sekaligus. Demikian pula, pada **Gambar 2.14**, tabel

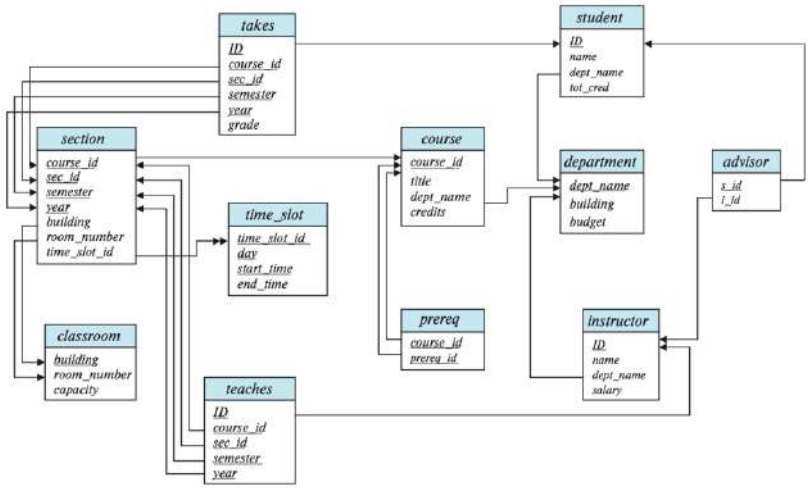
COAUTHOR memiliki *primary key* yang terdiri dari dua bidang, yang juga merupakan gabungan atau gabungan dari hubungan dua *foreign key* kembali ke tabel **AUTHOR** dan tabel **PUBLICATION**.



Gambar 2.13. *Foreign key* pada ER-D dan Tabel

2.5 Diagram Relasi Basis Data

Gambar 2.14 menunjukkan diagram relasi basis data untuk organisasi universitas. Setiap relasi muncul sebagai kotak, dengan nama relasi di bagian atas berwarna biru dan atribut yang tercantum di dalam kotak. Atribut *primary key* ditampilkan dengan garis bawah. Batasan *foreign key* muncul sebagai tanda panah dari atribut *foreign key* dari relasi referensi ke *primary key* dari relasi yang direferensikan.



Gambar 2.14. Diagram Relasi Basis Data Universitas

DAFTAR PUSTAKA

Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM, 13(6), 377-387.

Date, C. J. (2004). "An Introduction to Database Systems." 8th Edition. Addison-Wesley.

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

Powell, G. (2006). “Beginning database Design”. Wiley Publishing.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). "Database System Concepts." 6th Edition. McGraw-Hill.

BAB 3
INTRODUCTION TO SQL

Oleh Daniel Alfa Puryono

3.1 Mengenal Structured Query Language

Structured Query Language (SQL) merupakan bahasa pemrograman untuk mengakses, memanipulasi dan memproses informasi dalam basis data relasional. Pada tahun 1986 SQL sudah menjadi standar *American National Standards Institute* (ANSI) dan *International Organization for Standardization* (ISO) pada tahun 1987. Meskipun SQL sudah menjadi standar ANSI/ISO, akan tetapi ada beberapa versi bahasa dalam SQL itu sendiri. Namun untuk perintah utama, seperti Select, Update, Delete, Insert, Where masih sama dan sesuai dengan standar ANSI (Chamberlin, 2012).

Dasar untuk aplikasi SQL adalah *Relational Database Management System* (RDBMS). Maka tidak heran kalau semua sistem basis data modern seperti MySQL, Microsoft Access, SQL Server, dan Oracle pasti ada SQL di dalamnya.

Proses dalam RDBMS itu sendiri dimulai dari data disimpan dalam tabel basis data. Tabel terdiri dari kumpulan masukan data terkait yang terdiri dari kolom dan baris. Setiap tabel dipecah menjadi entitas yang lebih kecil yang disebut kolom. Kolom dirancang untuk menyimpan informasi spesifik setiap record. Contoh kolom dalam tabel pelanggan terdiri dari id_pelanggan, nama, email, telepon dan alamat. Sedangkan record atau juga disebut baris merupakan setiap entrian yang ada dalam tabel. Misalkan ada 57 rekaman dalam tabel pelanggan.

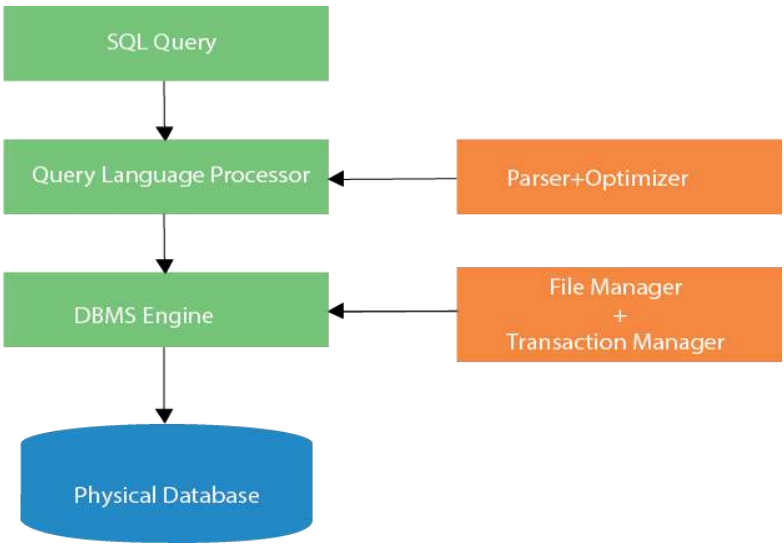
Secara umum, ada empat jenis perintah utama dalam bahasa SQL. Yang pertama disebut *Data Definition Language* (DDL). DDL ini memungkinkan kita untuk membuat dan memodifikasi database itu sendiri. Yang kedua *Data Manipulation Language* (DML). Komponen bahasa ini memungkinkan kita untuk mengambil, memperbarui, menambah, atau menghapus data dalam database. Kemudian komponen yang ketiga yaitu *Data Control Language* (DCL), yaitu

digunakan untuk menjaga keamanan basis data yang tepat. Kemudian yang terakhir adalah *Transaction Control Language* (TCL) komponen ini berfokus pada pengelolaan transaksi dalam basis data. Berikut adalah perintah utama yang ada dalam SQL.

Tabel 3.1. Perintah Dalam SQL

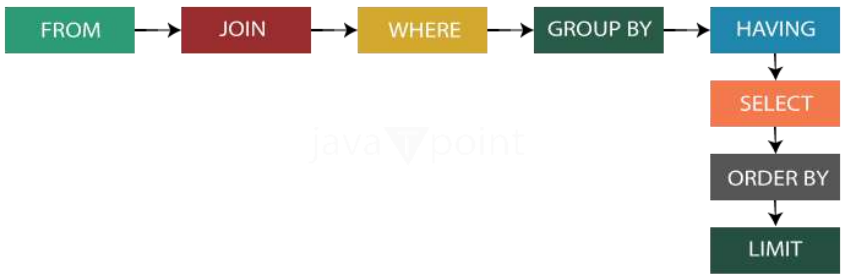
Perintah SQL			
DDL	**DML**	**DCL**	**TCL**
✓ Create ✓ Drop ✓ Alter ✓ Truncate ✓ Rename	✓ Insert ✓ Update ✓ Delete ✓ Select	✓ Grant ✓ revoke	✓ Commit ✓ Rollback ✓ Savepoint

Proses kerja dari SQL adalah sebagai berikut. Saat perintah SQL dijalankan untuk RDBMS, sistem akan mencari cara terbaik untuk menjalankan permintaan tersebut dan mesin SQL menentukan cara menginterpretasikan tugas tersebut (Jaiswal, dkk., 2024).



Gambar 3.1. Proses Kerja SQL

Sedangkan urutan logis eksekusi untuk pernyataan SQL seperti pada gambar 3.2 berikut. Meskipun saat menulis kode SQL, pernyataan SELECT sering ditulis terlebih dahulu untuk memudahkan pemahaman query.



Gambar 3.2. Urutan Eksekusi Query SQL

Dalam representasi diagram di atas, berikut penejlasanya:

1. Parser yaitu pernyataan query untuk memeriksa sintaks yang benar dari data masukan.
2. Optimizer merupakan pernyataan SQL dalam mengoptimalkan algoritma terbaik untuk kode byte.
3. From adalah kata kunci yang digunakan untuk menentukan tabel tempat data diambil.
4. Join adalah pernyataan yang digunakan untuk menggabungkan data lebih dari satu tabel berdasarkan kolom umum di antara tabel-tabel yang ada.
5. Where merupakan kata kunci yang berfungsi sebagai pernyataan kondisional dalam SQL.
6. Group by biasanya digunakan untuk mengelompokkan kolom berdasarkan rekaman yang berbeda dari tabel.
7. Having adalah klausa yang juga berfungsi seperti pernyataan kondisional dalam SQL. Klausa ini sebagian besar digunakan bersama dengan klausa group by untuk memfilter rekaman.
8. Select adalah pernyataan DML yang digunakan untuk mendapatkan data dari basis data.
9. Order by merupakan perintah yang dipakai untuk mengurutkan data dalam urutan tertentu. Seperti "ASC"

untuk urutan kecil kebesar dan "DESC" untuk urutan dari besar ke kecil.

10. Limit biasanya digunakan untuk menentukan berapa banyak baris yang dibatasi oleh pernyataan select dalam SQL.

3.2 Data Definition Language

DDL adalah salah satu perintah utama bahasa SQL yang dipakai untuk mendefinisikan struktur atau skema basis data. Komponen perintah ini memungkinkan pengguna untuk membuat, mengubah serta menghapus objek-objek dalam basis data. Seperti tabel, indeks maupun tampilan. Perintah ini juga dipakai untuk mengelola metadata dan menggambarkan struktur serta skema dalam basis data, sehingga memungkinkan pengguna untuk mengatur bagaimana data akan disimpan dan diorganisir dalam basis data (Poppy, 2024). Berikut adalah perintah dalam DDL yang biasanya digunakan.

1. CREATE

   Interuksi ini dipakai untuk membuat basis data, tabel serta tampilan dan indeks. Contonya seperti pada tabel 3.2 berikut.

Tabel 3.2. Contoh Perintah Create

Membuat databases	`CREATE DATABASE CV_dapcenter;`

Membuat tabel	`CREATE TABLE pelanggan (`
	`  id_pelanggan INT PRIMARY KEY,`
	`  nama VARCHAR(100),`
	`  email VARCHAR(100),`
	`  telepon VARCHAR(15)`
	`);`

Membuat tampilan	`CREATE VIEW pelanggan_view AS` `SELECT nama, email FROM pelanggan;`
Membuat indeks	`CREATE INDEX idx_pelanggan_telepon ON pelanggan (telepon);`

2. ALTER

   Interuksi ini dipakai untuk mengubah struktur tabel pada data basis yang sudah ada. Berikut contoh penggunaanya.

```
ALTER TABLE pelanggan ADD Alamat text;
```

3. DROP
   Interuksi ini dipakai untuk menghapus tabel pada basis data yang tidak dipakai lagi.
   ```
   1 DROP TABLE pelanggan;
   ```
4. TRUNCATE
   Interuksi ini dipakai untuk menghapus semua record dari tabel dengan cepat, namun tetap mempertahankan struktur tabel yang ada.
   ```
   1 TRUNCATE TABLE pelanggan;
   ```
5. RENAME
   Interuksi ini dipakai untuk mengganti nama tabel yang ada di database. Penggunaanya seperti berikut:
   ```
   1 RENAME TABLE COSTUMER TO pelanggan;
   ```

DDL seringkali dianggap sebagai perintah yang berisiko tinggi karena beberapa alasan seperti:

1. *Irreversibility*: perintah DDL seperti DROP dapat menghapus objek database secara permanen, dan setelah dihapus, data tersebut tidak dapat dikembalikan. Hal ini membuat perintah DDL sangat berisiko karena kesalahan dalam penggunaannya dapat menyebabkan data hilang secara permanen.
2. Keterlibatan struktur: perintah DDL seperti create dan alter dapat mempengaruhi struktur database secara langsung. Jika perintah ini dipaksakan pada database yang sudah ada dan ada kesalahan dalam penggunaan. Hal ini juga dapat menyebabkan struktur database menjadi tidak konsisten atau tidak sesuai dengan kebutuhan aplikasi.
3. Potensi kerusakan: perintah DDL seperti truncate dapat menghapus semua record dari tabel. Hal ini dapat menyebabkan kerusakan data, jika tidak dilakukan dengan hati-hati dalam melakukannya.

3.3 Data Manipulation Language

DML merupakan bagian penting dalam SQL yang dipakai untuk mengelola dan memanipulasi data pada basis data. DML mencakup berbagai perintah seperti menambah, mengubah, dan

menghapus data pada tabel. Perintah ini berfungsi untuk melakukan operasi dasar yang dikenal sebagai *Create, Read, Update, Delete* (CRUD).

Berbeda dengan DDL, yang berfokus pada struktur dan skema database, DML khusus menangani manipulasi data itu sendiri. Dalam konteks SQL, DML sering dianggap sebagai subbahasa dari SQL yang lebih luas (Feby, 2023).

Berikut adalah perintah-perintah yang termasuk dalam DML:

1. INSERT
   Dipakai untuk memasukan data baru ke dalam sebuah tabel. Contoh perintahnya sebagai berikut :

```sql
INSERT INTO CV_dapcenter.pelanggan (id_pelanggan, nama, email, telepon, alamat)
VALUES (022, 'Elshadai', 'elshadai@gmail.com', '08123456789', 'Jl. SMAN 1 No. 1');
```

2. SELECT
   Dipakai untuk mengambil atau membaca data dari tabel tertentu. Jadi perintah ini juga memungkinkan pengguna untuk mengambil informasi spesifik dari database berdasarkan kriteria tertentu. Berikut adalah contoh lengkap dari beberapa perintah SELECT yang dapat digunakan untuk mengelola data pelanggan.

```sql
-- Menampilkan semua data pelanggan
SELECT * FROM CV_dapcenter.pelanggan;

-- Menampilkan hanya nama dan email pelanggan
SELECT nama, email FROM CV_dapcenter.pelanggan;
-- Menampilkan data pelanggan dengan email tertentu

SELECT * FROM CV_dapcenter.pelanggan
WHERE email = 'elsadai@gmail.com';

-- Menampilkan data pelanggan dalam urutan alfabetik berdasarkan nama
SELECT * FROM CV_dapcenter.pelanggan
ORDER BY nama ASC;

-- Menampilkan 10 data pelanggan terakhir yang ditambahkan
SELECT * FROM CV_dapcenter.pelanggan
ORDER BY id_pelanggan DESC
LIMIT 10;
```

3. UPDATE

   Dipakai untuk memperbarui data yang sudah ada di dalam tabel. Berikut adalah contoh lengkap dari beberapa perintah UPDATE.

```sql
-- Memperbarui alamat pelanggan dengan email tertentu
UPDATE CV_dapcenter.pelanggan
SET alamat = 'Jl. Baru Pati No. 111'
WHERE email = 'maman11@gmail.com';

-- Memperbarui nama pelanggan dengan id tertentu
UPDATE CV_dapcenter.pelanggan
SET nama = 'Jehovan Narendra W'
WHERE id_pelanggan = 011;

-- Memperbarui telepon pelanggan dengan email tertentu
UPDATE CV_dapcenter.pelanggan
SET telepon = '08123455758'
WHERE email = 'jeni.jono@gmail.com';
```

4. DELETE

   Dipakai untuk menghapus data pada tabel, berikut contoh penerapanya.

```sql
-- Menghapus data pelanggan dengan email tertentu
DELETE FROM CV_dapcenter.pelanggan
WHERE email = 'jeni.jono@gmail.com';

-- Menghapus semua data pelanggan
DELETE FROM CV_dapcenter.pelanggan;
```

Jadi DML adalah komponen kunci dalam bahasa SQL yang memungkinkan manipulasi data secara efisien dan efektif. Maka dengan memahami perintah-perintah dasar DML seperti Insert, Select, Update, dan Delete, pengguna dapat melakukan berbagai operasi dalam database, serta menjadikannya alat yang sangat berharga bagi pengembang dan administrator. Selain itu juga dapat membantu menjaga integritas dan akurasi informasi dalam database.

3.4 Data Control Language

DCL dipakai untuk mengelola hak akses dan kontrol keamanan pada database. DCL berfungsi untuk memberikan atau mencabut izin kepada pengguna untuk mengakses dan memanipulasi data di dalam sistem database.

Selain itu DCL juga dapat digunakan untuk memantau dan mengaudit penggunaan database, sehingga administrator dapat mengetahui aktivitas apa saja yang dilakukan oleh pengguna. Jadi DCL sangat penting dalam menjaga keamanan dan integritas data, terutama dalam sistem multi-user (Maulid, 2022).

Berikut dua perintah utama dalam DCL yang sering digunakan:

1. GRANT

   Instruksi GRANT dipakai untuk memberi pengguna hak akses tertentu. Jadi administrator dapat menentukan jenis akses yang diberikan, seperti kemampuan untuk melakukan operasi SELECT, INSERT, UPDATE, atau DELETE pada tabel tertentu.

   Contoh: memberikan akses Select pada tabel pelanggan dalam database cv_dapcenter ke pengguna daniel.

```
1 GRANT SELECT ON cv_dapcenter.pelanggan TO 'daniel'@'localhost';
```

   Sedangkan untuk memberikan semua hak akses *(full access)* kepada pengguna pada semua objek dalam database adalah:

```
1 GRANT ALL PRIVILEGES ON cv_dapcenter.* TO 'admin'@'localhost';
```

   Kita juga bisa memberikan beberapa hak akses sekaligus seperti INSERT, UPDATE, atau DELETE.

```
1 GRANT INSERT, UPDATE, DELETE ON cv_dapcenter.pelanggan TO 'daniel'@'localhost';
```

2. REVOKE

   Hak akses yang telah diberikan sebelumnya dicabut dengan menggunakan perintah REVOKE ini. Jadi administrator dapat membatasi akses pengguna ke data tertentu jika diperlukan.

```
1 REVOKE SELECT ON CV_dapcenter.pelanggan FROM 'daniel'@'localhost';
```

   Namun jika hanya ingin mencabut hak akses yang lebih spesifik, seperti hanya mencabut hak INSERT, kita dapat menggunakan sintaks yang lebih spesifik seperti

```
1 REVOKE INSERT ON CV_dapcenter.pelanggan FROM daniel @'localhost';
```

3.5 *Transaction Control Language*

TCL juga merupakan bagian dari SQL yang berfokus pada pengelolaan transaksi di database. TCL digunakan untuk mengendalikan dan mengelola perbaikan yang telah dilakukan oleh pernyataan DML. Sehingga pengguna dapat memastikan bahwa serangkaian operasi database dieksekusi dengan aman dan konsisten (Maulid, 2022) dan (Nursyafitri, 2023).

Beberapa alasan menggapa perintah TCL sangat penting dalam pengelolaan basis data.

1. Perintah TCL membantu menjaga integritas data dengan memastikan bahwa semua perubahan yang dilakukan dalam transaksi berhasil atau tidak sama sekali. Jadi jika ada kesalahan dalam salah satu pernyataan, maka seluruh transaksi dapat dibatalkan untuk mencegah data yang tidak konsisten.
2. Pengguna dapat mengelompokkan beberapa pernyataan DML menjadi satu transaksi logis atau manajemen transaksi. Sehingga hal ini memungkinkan pengguna untuk melakukan beberapa operasi sekaligus dan menjamin bahwa semua operasi tersebut berhasil sebelum menyimpan perubahan ke database.
3. Bisa melakukan rollback dan recovery. Sehingga jika terjadi kesalahan selama eksekusi transaksi, TCL memungkinkan pengguna untuk membatalkan perubahan yang telah dilakukan dan mengembalikan database pada keadaan sebelumnya.

Berikut adalah perintah utama dalam TCL:

1. COMMIT
   Perintah ini digunakan untuk menyimpan semua perubahan yang telah dilakukan selama transaksi ke dalam database. Setelah perintah COMMIT dijalankan, perubahan tidak dapat dibatalkan.

```sql
BEGIN;
-- Menambahkan data pelanggan baru
INSERT INTO CV_dapcenter.pelanggan (nama, email, telepon, alamat)
VALUES ('alfa puryono', 'alfa@gmail.com', '085727369123', 'Jl. Kamandowo No. 13 Pati');

-- Memperbarui alamat pelanggan yang sudah ada
UPDATE CV_dapcenter.pelanggan
SET alamat = 'Jl. Kopi Tempur No. 11'
WHERE email = 'suwardi@gmail.com';

-- Menyimpan semua perubahan secara permanen
COMMIT;
```

2. ROLLBACK

   Perintah ini digunakan untuk membatalkan semua perubahan yang telah dilakukan selama transaksi jika terjadi kesalahan. Setelah ROLLBACK dijalankan, database akan kembali ke keadaan sebelum transaksi dimulai.

```sql
BEGIN;
-- Menambahkan data pelanggan baru
INSERT INTO CV_dapcenter.pelanggan (nama, email, telepon, alamat)
VALUES ('alfa puryono', 'alfa@gmail.com', '085727369123', 'Jl. Kamandowo No. 13 Pati');

-- Memperbarui alamat pelanggan yang sudah ada
UPDATE CV_dapcenter.pelanggan
SET alamat = 'Jl. Kopi Tempur No. 11'
WHERE email = 'suwardi@gmail.com';

-- Oops, terjadi kesalahan atau perubahan rencana
ROLLBACK;
```

3. SAVEPOINT

   Perintah ini digunakan untuk menetapkan titik simpan dalam suatu transaksi. sehingga pengguna dapat kembali ke titik tersebut jika diperlukan.

```sql
BEGIN;
-- Menambahkan data pelanggan baru
INSERT INTO CV_dapcenter.pelanggan (nama, email, telepon, alamat)
VALUES ('alfa puryono','alfa@gmail.com', '085727369123','Jl.Kamandowo No.13 Pati');
SAVEPOINT sp1;  -- Menandai titik setelah penambahan
```

3.6 Constraints

Constraints adalah aturan atau batasan yang diterapkan pada data dalam tabel untuk menjaga integritas, konsistensi dan keamanan data. Misalnya, penggunaan foreign key dapat memastikan bahwa setiap referensi ke entitas lain dalam basis data valid dan konsisten. Penggunaan not null juga dapat memastikan bahwa kolom sensitif tidak bisa dibiarkan kosong. Maka dalam pengelolaan basis data, penggunaan constraints sangat penting untuk memastikan bahwa data yang disimpan memenuhi kriteria tertentu serta dapat diandalkan. Karena penggunaan constraints dapat membantu memastikan bahwa data yang dimasukkan ke dalam tabel cocok dengan susunan yang telah ditentukan. Hal ini dapat mencegah data yang tidak valid dapat dimasukkan pengguna dengan sembarangan ke dalam basis data (Gunawan, 2024).

Berikut adalah beberapa jenis constraints yang biasanya digunakan dalam basis data:

1. Primary key digunakan untuk menetapkan satu atau beberapa kolom sebagai kunci primer, sehingga memungkinkan setiap baris dalam tabel diidentifikasi secara unik.
2. Unique untuk memastikan bahwa nilai dalam satu kolom atau kombinasi kolom adalah unik di antara semua nilai dalam kolom tersebut.
3. Foreign key digunakan untuk menghubungkan kolom dalam satu tabel dengan kolom primary key atau unique key ke tabel lain. Selain itu juga dapat memastikan integritas referensial antara dua tabel.
4. Check digunakan untuk menetapkan kondisi tertentu yang harus dipenuhi oleh nilai dalam kolom, seperti rentang nilai yang diizinkan atau format data yang valid.
5. Not null dipakai untuk memastikan pada kolom tertentu tidak boleh kosong atau NULL, sehingga mencegah nilai yang tidak valid atau tidak lengkap dimasukkan ke dalam basis data.
6. Default digunakan untuk menetapkan nilai pada kolom jika tidak ada nilai yang disediakan saat memasukkan data ke dalam tabel.

Berikut contoh penggunaan perintah constraints:

```
CREATE TABLE CV_dapcenter.pelanggan (
 id_pelanggan INT AUTO_INCREMENT PRIMARY KEY, -- PRIMARY KEY constraint
 nama VARCHAR(100) NOT NULL,                  -- NOT NULL constraint
 email VARCHAR(100) UNIQUE NOT NULL,          -- UNIQUE and NOT NULL constraints
 telepon VARCHAR(15),
 alamat TEXT,
 tanggal_daftar TIMESTAMP DEFAULT CURRENT_TIMESTAMP,-- DEFAULT constraint
 CHECK (LENGTH(telepon) = 12) -- CHECK constraint untuk memastikan telepon 12 digit
);
```

DAFTFAR PUSTAKA

Chamberlin, D. D. (2012). Early history of SQL. *IEEE Annals of the History of Computing,* 34(4), 78-82.

Feby, Dita. 2023. *Perbedaan Sistem Operasi DDL Vs DML di Comment Strart SQL,* (https://dqlab.id/perbedaan-sistem-operasi-ddl-vs-dml-di-comment-start-sql, Diakses: 2 Oktober 2024).

Gunawan, Nanang. 2024. *Mengenal Constraints Pada Database,* (https://www.nananggunawan.com/2024/04/mengenal-con-straints-pada-database.html. Diakses: 2 Oktober 2024).

Jaiswal, Sanoo. 2024. *SQL Tutorial,* (https://www.javatpoint.com/dbms -sql-introduction Diakses: 2 Oktober 2024).

Maulid, Reyvan. 2022. *Pahami Sistem Operasi SQL dengan 5 Perintah Dasar,* (https://dqlab.id/pahami-sistem-operasi-sql-dengan-5-perintah-dasarnya. Diakses: 6 Oktober 2024).

Nursyafitri, Gifa Delyani. 2023. *Mengenal Command DDL, DML, DCL, dan TCL,* (https://dqlab.id/mengenal-command-ddl-dml-dcl-dan-tcl-dalam-mysql, Diakses: 6 Oktober 2024).

Poppy, Daniel. 2024. *Guide To DDL,* (https://docs.getdbt.com/terms/ddl. Diakses: 6 Oktober 2024).

BAB 4
INTERMEDIATE SQL

Oleh Hanes

4.1 Pendahuluan

Pada Bab sebelumnya kita telah mempelajari tentang *Introduction SQL* yaitu pengenalan bahasa baku yang digunakan untuk membuat dan menkueri basis data relasional. Kueri merupakan permintaan akan data atau informasi dari sebuah atau beberapa tabel di dalam satu atau beberapa basis data. Penggunaan kueri dalam menghasilkan data dari tabel tidaklah cukup. Diperlukan fungsi lain yang dapat membantu pengguna dalam memahami pembacaan data.

Bab ini membahas fungsi agregat dan kueri dari beberapa tabel basis data, dan akan dijabarkan dalam beberapa bagian. Beberapa pendekatan yang digunakan untuk mendapatkan hasil kueri dari lebih dari satu tabel akan dijelaskan lebih jelas, termasuk penggunaan *inner* dan *outer join*.

Penerapan kueri pada Bab ini menggunakan basis data *AdvantureWorks* yang merupakan sampel basis data yang disediakan dan dapat diunduh di halaman web Microsoft.

4.2 Fungsi Agregat

SQL sangat baik dalam melakukan agregasi data seperti yang dapat dilakukan pada tabel pivot di aplikasi *spreadsheet*. Fungsi agregat dapat digunakan sepanjang melakukan kueri untuk mendukung menghasilkan informasi yang lebih berguna bagi pembacaan data. Sebuah fungsi agregat melakukan kalkulasi terhadap sekumpulan nilai dan mengembalikan satu nilai. Kecuali fungsi COUNT, fungsi agregat lainnya akan mengabaikan nilai null. Fungsi agregat sering digunakan bersamaan dengan fungsi GROUP BY dari pernyataan SELECT. Beberapa fungsi agregat yang akan dibahas pada Bab ini adalah COUNT, SUM, MIN, MAX, dan AVG.

4.2.1 Fungsi COUNT

Fungsi COUNT digunakan untuk menghitung seberapa banyak baris di dalam kolom tertentu. Fungsi COUNT merupakan fungsi agregat yang paling sederhana karena dapat diverifikasi dengan mudah. COUNT selalu mengembalikan sebuah nilai bertipe data *integer*.

Kueri baku untuk fungsi COUNT adalah

```
COUNT ( { [ [ ALL | DISTINCT ] expression ] | * } )
```

ALL menerapkan fungsi agregat ke semua nilai. ALL berfungsi sebagai default.

DISTINCT menentukan fungsi COUNT untuk mengembalikan jumlah dari data unik yang bukan *null*.

Expression berupa tipe data apa pun kecuali *image*, *ntext*, atau *text*. COUNT tidak mendukung fungsi atau *subquery* dalam sebuah *expression*.

***** menentukan fungsi COUNT menghitung semua baris untuk menentukan jumlah keseluruhan baris tabel yang akan dikembalikan. COUNT (*) tidak menggunakan parameter dan tidak mendukung penggunaan DISTINCT. COUNT (*) tidak memerlukan parameter expression karena menurut definisi, COUNT tidak menggunakan informasi tentang kolom tertentu. Baris yang memiliki nilai yang sama (duplikat) atau bernilai *null* akan dihitung sebagai baris baru.

Kueri : Penggunaan COUNT dan DISTINCT

```
SELECT COUNT(DISTINCT JobTitle)
FROM HumanResources.Employee;
```

Contoh kueri di atas mengembalikan jumlah jenis pekerjaan yang dapat diemban oleh karyawan. Jumlah yang akan dikembalikan oleh kueri diatas adalah 67 baris jenis pekerjaan. Tabel HumanResources.Employee memiliki 290 baris data dimana setiap karyawan memiliki jenis pekerjaan dan memungkinkan ada beberapa karyawan yang memiliki jenis pekerjaan yang sama. Fungsi DISTINCT digunakan untuk menghilangkan duplikasi pada field JobTitle sehingga datanya menjadi unik. Setelah data pada field JobTitle menjadi unik, kemudian fungsi COUNT akan menghitung jumlah baris dari data tersebut.

Kueri : Tampilkan jumlah keseluruhan karyawan yang ada pada tabel HumanResource.Employee.

```
SELECT COUNT(*)
FROM HumanResources.Employee;
```

Hasil dari kueri di atas adalah sebanyak 290 data karyawan di Adventure Works Cycles.

4.2.2 Fungsi SUM

Fungsi SUM mengembalikan penjumlahan semua nilai, atau hanya pada nilai DISTINCT, di dalam sebuah expression.

```
SUM ( [ ALL | DISTINCT ] expression
```

ALL menerapkan fungsi agregat ke semua nilai. ALL berfungsi sebagai default.

DISTINCT menentukan pengembalian hasil penjumlahan dari nilai yang unik.

Expression merupakan konstanta, kolom, atau fungsi, dan berbagai kombinasi matematika, bitwise, dan string. *Expression* bisa berupa tipe data numerik atau perkiraan numerik, kecuali bit. Fungsi agregat dan subquery tidak diperbolehkan pada parameter ini.

Kueri : Penggunaan SUM untuk mengembalikan kesimpulan data

```
SELECT ShipMethodID, SUM(Freight)
FROM Sales.SalesOrderHeader
GROUP BY ShipMethodID;
```

Hasil dari kueri di atas adalah total biaya kirim dari pesanan penjualan yang telah dilakukan.

```
ShipMethodID
-------------------- ---------------------
1                    733967.672
5                    2449462.5798
```

4.2.2 Fungsi MIN

Fungsi ini mengembalikan nilai terkecil dari sekumpulan nilai. MIN mengabaikan semua nilai null. Jika sekumpulan data mengandung karakter, maka MIN akan mengurutkan data tersebut dan mengambil nilai terkecilnya.

```
MIN ( [ ALL | DISTINCT ] expression)
```

ALL menerapkan fungsi agregat ke semua nilai. ALL berfungsi sebagai default.

DISTINCT memastikan setiap nilai bersifat unik.

Expression merupakan konstanta, kolom, atau fungsi, dan berbagai kombinasi matematika, bitwise, dan string. *Expression* dapat digunakan dengan tipe data numerik, char, varchar, Uniqueidentifier, atau datetime, kecuali bit. Fungsi agregat dan subquery tidak diperbolehkan pada parameter ini.

Kueri : Tampilkan tarif pajak terkecil yang dapat dikenakan kepada pelanggan yang melakukan pesanan penjualan.

```
SELECT                                          MIN(TaxRate)
FROM Sales.SalesTaxRate ;
```

Hasil dari kueri di atas menunjukkan bahwa tarif pajak yang dapat dikenakan pada penjualan adalah 5% dari total penjualan.

```
----------
5.0
```

4.2.4 Fungsi MAX

Fungsi ini mengembalikan nilai terbesar dari sekumpulan nilai. MAX mengabaikan semua nilai null. Fungsi MAX akan mengembalikan nilai NULL jika tidak ada baris yang terpilih. Jika sekumpulan data mengandung karakter, maka MAX akan mengurutkan data tersebut dan mengambil nilai terbesarnya.

```
MAX ( [ ALL | DISTINCT ] expression )
```

ALL menerapkan fungsi agregat ke semua nilai. ALL berfungsi sebagai default.

DISTINCT memastikan setiap nilai bersifat unik.

Expression merupakan konstanta, kolom, atau fungsi, dan berbagai kombinasi matematika, bitwise, dan string. *Expression* dapat digunakan dengan tipe data numerik, char, varchar, Uniqueidentifier, atau datetime, kecuali bit. Fungsi agregat dan subquery tidak diperbolehkan pada parameter ini.

Kueri : Tampilkan tarif pajak terkecil yang dapat dikenakan kepada pelanggan yang melakukan pesanan penjualan.

```
SELECT                                          MAX(TaxRate)
FROM Sales.SalesTaxRate ;
```

Hasil dari kueri di atas menunjukkan bahwa tarif pajak yang dapat dikenakan pada penjualan adalah 19.60% dari total penjualan.

```
----------
19.60
```

4.2.5 Fungsi AVG

Fungsi AVG mengembalikan nilai rata-rata dari sekumpulan nilai. Fungsi ini mengabaikan baris yang bernilai null.

```
AVG ( [ ALL | DISTINCT ] expression )
```

ALL menerapkan fungsi agregat ke semua nilai. ALL berfungsi sebagai default.

DISTINCT memastikan fungsi AVG hanya beroperasi pada satu nilai unik.

Expression merupakan tipe data numerik, kecuali bit. Fungsi agregat dan subquery tidak diperbolehkan pada parameter ini.

Jika tipe data dari *expression* adalah sebuah tipe data alias, maka tipe data alias tersebut yang akan dikembalikan. Namun, jika tipe data dasar dari tipe data alias ditingkatkan, misalnya dari *tinyint* ke *int*, nilai yang dikembalikan akan mengambil tipe data yang ditingkatkan tersebut, dan bukan tipe data alias.

AVG menghitung rata-rata sekumpulan nilai dengan membagi jumlah nilai tersebut dengan jumlah nilai bukan null. Jika jumlahnya melebihi nilai maksimum untuk tipe data dari nilai yang dikembalikan, AVG akan mengembalikan pesan kesalahan.

Kueri : Tampilkan harga rata-rata dari data produk dengan ketentuan produk tidak berulang.

```
SELECT AVG(DISTINCT ListPrice)
FROM Production.Product;
```

Hasil dari kueri di atas dapat dilihat di bawah ini.

```
----------------------------
437.4042
```

Kueri : Tampilkan harga rata-rata dari semua baris data produk terrmasuk semua nilai duplikat.

```
SELECT AVG(ListPrice)
FROM Production.Product ;
```

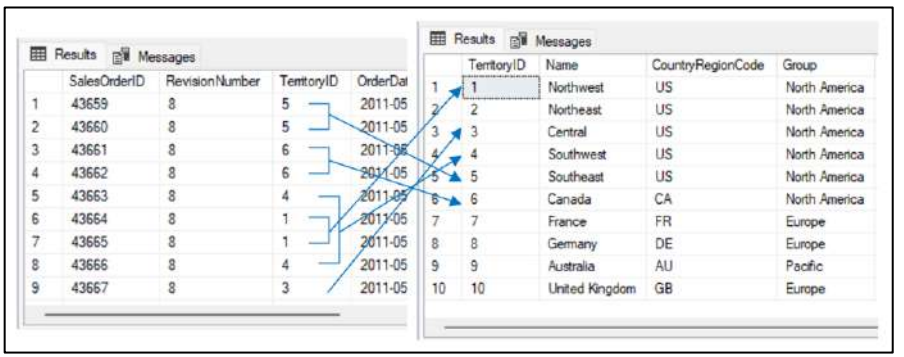
Hasil dari kueri di atas dapat dilihat di bawah ini.

```
---------------------------
438.6662
```

4.3 Bekerja Dengan Beberapa Tabel

Keunggulan Database Management System (DBMS) khususnya basis data relasional adalah bekerja dengan banyak tabel. Beberapa tabel dapat digabungkan dengan kueri jika diantara tabel tersebut memiliki hubungan. Hubungan tersebut terbentuk dengan mengaitkan antara kunci utama dari sebuah tabel yang menjadi kunci tamu pada tabel lain, dan nilai-nilai di dalam kedua tabel tersebut memiliki domain yang sama (Jeffrey dkk, 2022).

Pada gambar 4.1 menunjukkan hubungan antara dua tabel yang merupakan bagian dari basis data AdvantureWork. Dapat dilihat bahwa kolom TerritoryID di tabel SalesOderHeader sesuai dengan nilai TerritoryID di tabel SalesTerritory. Dengan begitu dapat disimpulkan bahwa pesanan penjualan 43659 dan 43660 dipesan oleh pelanggan yang berada di wilayah Southwest dengan TerritoryID 5.



Gambar 4.1. Hubungan antara tabel SalesOrderHeader dan SalesTerritory

Dalam database relasional, data dari kedua tabel digabungkan menjadi satu tabel dan kemudian ditampilkan atau digunakan sebagai informasi pada sebuah formulir atau laporan.
Pada basis data relasional, operasi yang sering digunakan untuk menggabungkan data dari dua atau lebih tabel yang terkait menjadi

satu tabel disebut JOIN. Secara eksplisit, JOIN memerlukan perintah ON yang disertakan pada klausa FROM dengan kata kunci seperti INNER, OUTER, FULL, LEFT, RIGHT, dan UNION yang mungkin berbeda untuk beberapa produk DBMS. Apa pun bentuk JOIN yang digunakan, harus ada satu perintah ON atau WHERE untuk setiap pasangan tabel yang digabungkan. Jika dua tabel ingin digabungkan, maka diperlukan satu kondisi ON atau WHERE, tetapi jika tiga tabel ingin digabungkan, maka diperlukan dua kondisi ON atau WHERE karena terdapat 2 (dua) pasangan (tabel A dengan B dan B dengan C), dan seterusnya. Kebanyakan sistem mendukung hingga 10 pasang tabel dalam satu kueri SQL.

4.3.1 Equi-Join

Equi-Join menggabungkan data berdasarkan kesamaan antara nilai-nilai yang ada pada kolom-kolom yang sama pada tabel yang akan digabungkan. Sebagai contoh, jika kita ingin mengetahui data wilayah dari pesanan penjualan, kita bisa mendapatkannya dari dua tabel yaitu SalesOrderHeader dan Sales Territory. Kita dapat mencocokkan antara pemesanan penjualan dengan wilayah yang ada, kemudian menampilkan data lainnya seeperti SalesOrderID, SalesOrderNumber, OrderDate, TerritoryID, dan TerritoryName untuk menjawab informasi yang dibutuhkan.

Kueri : Apa ID pesanan penjualan, nomor pesanan penjualan, tanggal pesanan, ID Wilayah, dan nama wilayah untuk semua wilayah penjualan yang telah dilakukan ?

```
SELECT SalesOrderID, SalesOrderNumber, OrderDate,
Sales.SalesOrderHeader.TerritoryID,
Sales.SalesTerritory.TerritoryID, Name
FROM Sales.SalesOrderHeader, Sales.SalesTerritory
WHERE sales.SalesOrderHeader.TerritoryID =
Sales.SalesTerritory.TerritoryID
```

Hasil :

	SalesOrderID	SalesOrderNumber	OrderDate	TerritoryID	TerritoryID	Name
1	43659	SO43659	2011-05-31 00:00:00.000	5	5	Southeast
2	43660	SO43660	2011-05-31 00:00:00.000	5	5	Southeast
3	43661	SO43661	2011-05-31 00:00:00.000	6	6	Canada
4	43662	SO43662	2011-05-31 00:00:00.000	6	6	Canada
5	43663	SO43663	2011-05-31 00:00:00.000	4	4	Southwest
6	43664	SO43664	2011-05-31 00:00:00.000	1	1	Northwest
7	43665	SO43665	2011-05-31 00:00:00.000	1	1	Northwest
8	43666	SO43666	2011-05-31 00:00:00.000	4	4	Southwest
9	43667	SO43667	2011-05-31 00:00:00.000	3	3	Central

Gambar 4.2. Hasil Kueri Equi-Join

Gambar 4.2 menunjukkan hasil penggambungan data dari dua tabel yaitu SalesOrderHeader dan SalesTerritory. Dua kolom TerritoryID yang dihasilkan berasal dari tabel yang berbeda dan menunjukkan bahwa kedua tabel tersebut telah cocok yang menghasilkan satu baris untuk setiap pemesanan. Pada kueri kita mengawali kolom TerritoryID dengan nama tabel, hal ini bertujuan agar SQL mengetahui kolom TerritoryID yang direferensikan berasal dari kedua tabel tersebut. Kita tidak perlu memberikan awalan nama tabel pada SalesOrderID, SalesOrderNumber, OrderDate, dan Name karena kolom-kolom tersebut hanya terletak pada salah satu tabel.

Kecocokan antar tabel terlihat pada klausa WHERE pada kueri join. Jika klausa tersebut dihilangkan, maka SQL mengembalikan kombinasi setiap baris pada tabel pesanan dengan setiap baris pada tabel wilayah tanpa melihat kecocokan antara kedua tabel tersebut. Dalam hal ini, gabungan data dari kedua tabel tersebut mengembalikan hasil yang tidak bermakna. Jika tabel pesanan terdapat 10 baris dan tabel wilayah terdapat 5 baris data, maka hasil yang akan dikembalikan adalah 50 baris data yaitu perkalian antara 10 dan 5. Penggambungan data tersebut disebut dengan gabungan Cartesian. Gabungan Cartesian terjadi ketika klausa WHERE menggabungkan setiap komponen data pada tabel atau memiliki kondisi yang salah. Gabungan Cartesian juga dapat dibuat dengan menggunakan klausa CROSS JOIN pada pernyataan FROM. Misalkan FROM SalesOrderHeader CROSS JOIN

SalesTerritory akan menghasilkan gabungan Cartesian jutaan baris data.
Kata kunci INNER JOIN … ON digunakan membuat equi-join pada klausa FROM. Beberapa sistem SQL mungkin tidak memerlukan kata kunci INNER mengawali klausa JOIN.
Kueri : Apa ID pesanan penjualan, nomor pesanan penjualan, tanggal pesanan, ID Wilayah, dan nama wilayah untuk semua wilayah penjualan yang telah dilakukan ?

```
SELECT SalesOrderID, SalesOrderNumber, OrderDate,
Sales.SalesOrderHeader.TerritoryID,
Sales.SalesTerritory.TerritoryID, Name
FROM Sales.SalesOrderHeader INNER JOIN Sales.SalesTerritory
ON sales.SalesOrderHeader.TerritoryID =
Sales.SalesTerritory.TerritoryID
```

Hasil : sama seperti kueri sebelumnya

Untuk menyederhanakan kueri join, kita dapat menggunakan klausa JOIN ... USING jika sistem DBMS yang dipakai mendukung klausa tersebut. Penggunaan klausa tersebut hanya dapat dilakukan jika basis data menggunakan nama kolom yang identik untuk kunci utama dan kunci tamu yang menghubungkan kedua tabel yang akan dihubungkan, seperti yang telah dilakukan pada tabel SalesOrderHeader dan SalesTerritory pada kolom TerritoryID. Berikut kueri yang dapat digunakan:

```
SELECT SalesOrderID, SalesOrderNumber, OrderDate,
Sales.SalesOrderHeader.TerritoryID,
Sales.SalesTerritory.TerritoryID, Name
FROM Sales.SalesOrderHeader JOIN Sales.SalesTerritory
USING (TerritoryID);
```

Perlu diperhatikan bahwa penggunaan klausa JOIN ... ON dan JOIN ... USING akan mengembalikan fungsi WHERE menjadi penyaring data untuk menghasilkan cakupan data yang lebih kecil.
Kueri : Apa ID pesanan penjualan, nomor pesanan penjualan, tanggal pesanan, ID Wilayah, dan nama wilayah untuk wilayah pesanan penjualan 'Canada' yang telah dilakukan ?

```
SELECT SalesOrderID, SalesOrderNumber, OrderDate,
Sales.SalesOrderHeader.TerritoryID,
```

```
Sales.SalesTerritory.TerritoryID, Name
FROM Sales.SalesOrderHeader INNER JOIN Sales.SalesTerritory
ON sales.SalesOrderHeader.TerritoryID =
Sales.SalesTerritory.Territory
WHERE Name = 'Canada';
```

Hasil :

	SalesOrderID	SalesOrderNumber	OrderDate	TerritoryID	TerritoryID	Name
1	43661	SO43661	2011-05-31 00:00:00.000	6	6	Canada
2	43662	SO43662	2011-05-31 00:00:00.000	6	6	Canada
3	43668	SO43668	2011-05-31 00:00:00.000	6	6	Canada
4	43672	SO43672	2011-05-31 00:00:00.000	6	6	Canada
5	43674	SO43674	2011-05-31 00:00:00.000	6	6	Canada
6	43677	SO43677	2011-05-31 00:00:00.000	6	6	Canada
7	43679	SO43679	2011-05-31 00:00:00.000	6	6	Canada
8	43697	SO43697	2011-05-31 00:00:00.000	6	6	Canada
9	43769	SO43769	2011-06-18 00:00:00.000	6	6	Canada
10	43800	SO43800	2011-06-23 00:00:00.000	6	6	Canada

Gambar 4.3. Hasil Kueri Equi-Join Dengan Kondisi Tertentu

Setelah kueri dijalankan, SQL mengembalikan 4067 baris data pesanan penjualan yang dilakukan di wilayah ‘Canada’.

4.3.2 Outer Join

Penggambungan data dari dua tabel tidak selamanya memiliki kecocokan. Hal ini ditandai dengan tidak adanya baris pada sebuah tabel di tabel yang lain. Misalkan pada contoh kasus sebelumnya, mungkin ada pelanggan di sebuah wilayah yang tidak melakukan pemesanan penjualan. Jika menggunakan equi-join, data yang dimaksud tidak dapat dihasilkan karena tidak ada kecocokan yang terjadi di antara tabel SalesOrderHeader dan SalesTerritory pada saat digabungkan (Jeffrey dkk, 2022).

Perusahaan mungkin tertarik untuk mengetahui lebih dalam mengapa pelanggan dari wilayah tersebut belum melakukan pemesanan. Bagian terkait mungkin bisa menganalisis lebih dalam terhadap potensi pada wilayah tersebut, sehingga dapat membantu perusahaan untuk mengambil keputusan ekspansi perusahaan di wilayah tersebut. Dengan menggunakan outer join, baris yang tidak memiliki nilai yang cocok pada kolom yang dicocokkan juga

dimunculkan dalam tabel hasil. Nilai NULL akan muncul di kolom yang tidak ada kecocokan antar tabel tersebut.

Outer join dapat digunakan dalam sebagian besar DBMS, namun perintah yang digunakan bisa saja berbeda diantara DBMS tersebut. Pada bagian ini menggunakan perintah dengan standar American National Standsrds Institute (ANSI) (Jeffrey dkk, 2022).

Kueri : Tampilkan nama wilayah, ID wilayah, dan nomor pemesanan penjualan untuk semua wilayah yang terdapat pada tabel SalesTerritory. Sertakan nama wilayah dan ID wilayah meskipun tidak terdapat pesanan pada wilayah tersebut.

```
SELECT Sales.SalesTerritory.TerritoryID, Name, SalesOrderID
FROM Sales.SalesTerritory LEFT OUTER JOIN
Sales.SalesOrderHeader
ON SalesTerritory.TerritoryID = SalesOrderHeader.TerritoryID;
```

Penggunaan LEFT OUTER JOIN dipakai karena urutan tabel SalesTerritory pada kueri disertakan terlebih dahulu setelah klausa FROM atau berada di sebelah kiri. Disamping itu, ekspetasi hasil yang akan dikembalikan adalah semua data wilayah yang berada pada tabel tersebut tanpa melihat kecocokannya dengan tabel SalesOrderHeader. Jika urutan tabelnya dibalik, maka penggunaan RIGHT OUTER JOIN yang akan dipilih dan hasil yang diharapkan akan sama dengan penggunaan LEFT OUTER JOIN (Jeffrey dkk, 2022).

Penggunaan OUTER JOIN tidak mudah untuk diterapkan pada penggabungan lebih dari dua tabel. Pengembalian hasil dari kueri tersebut mungkin saja berbeda antara satu DBMS dengan DBMS lainnya. Jadi perlu dipastikan untuk selalu melakukan analisis dan pengujian terhadap hasil yang dikembalikan agar kita dapat memahami bagaimana sebuah DBMS menginterpretasikan kueri tersebut (Jeffrey dkk, 2022).

Selain itu, tabel hasil penggabungan dari kueri OUTER JOIN menghasilkan nilai NULL sebagai nilai pada kolom yang tidak memiliki kecocokan data. Jika pada kolom tersebut terdapat nilai NULL, maka kita tidak dapat mengetahui apakah nilai tersebut berasal dari ketidakcocokan antar kedua kolom atau berasal dari kolom nilai tersebut. Untuk itu kita perlu menjalan kueri lain untuk memeriksa kebenaran dari hasil yang dikembalikan. Perlu

diperhatikan bahwa kolom dengan pembatasan nilai NOT NULL akan mengembalikan nilai NULL pada tabel hasil penggabungan (Jeffrey dkk, 2022).

Keuntungan dari penggunaan OUTER JOIN adalah informasi yang dibutuhkan tidak hilang karena semua data pada sisi kiri (LEFT OUTER JOIN) atau kanan (RIGHT OUTER JOIN) akan dikembalikan oleh kueri (Jeffrey dkk, 2022).

DAFTAR PUSTAKA

Jeffrey A. Hoffer, V. Ramesh, & Heikki Topi. 2022. *Modern Database Management 13th Edition*. Penerbit: Pearson.

BAB 5
ADVANCED SQL

Oleh Rajif Agung Yunmar

Seiring dengan meningkatnya kebutuhan operasi basis data yang lebih canggih, fitur-fitur dasar SQL sering kali tidak cukup untuk menjawab berbagai tantangan yang lebih kompleks dalam manajemen data kontemporer. Untuk mengatasi keterbatasan ini, SQL tingkat lanjut muncul dengan berbagai alat dan kemampuan canggih, sehingga kemudian mendorong efisiensi, fleksibilitas, dan keamanan pada database yang lebih baik.

Stored program adalah fitur SQL tingkat lanjut dalam database yang memungkinkan eksekusi tugas-tugas kompleks dengan lebih cepat dan efisien. Fitur ini terdiri dari tiga bagian: *stored procedure*, *stored function*, dan *trigger*. Secara sederhana, *stored program* adalah kumpulan perintah SQL yang disimpan di dalam server database dan bisa digunakan kapan saja tanpa harus menulis ulang.

Karena dijalankan langsung di server, fitur ini dapat mengurangi lalu lintas data karena semua perintah SQL diproses langsung di dalam server database (Steven Feuerstein, 2006), tanpa perlu bolak-balik mengirimkan data dari dan ke aplikasi klien. Sehingga pada akhirnya, membuat performa sistem secara keseluruhan menjadi lebih cepat.

Beberapa keuntungan utama dari *stored program* diantaranya adalah:

1. Lebih cepat dan efisien: Semua proses berjalan langsung di server, aplikasi tidak perlu bolak-balik mengirim data. Hal ini membuat sistem bekerja lebih cepat dan ringan.
2. Otomatisasi : Tugas yang dilakukan secara berulang seperti memproses data, mencatat log, atau pemeriksaan validitas data dapat dilakukan secara otomatis dengan *stored program*.

3. Keamanan yang lebih baik: *Stored program* membantu mencegah serangan berbahaya seperti SQL *injection*. Hal ini dikarenakan data diproses di dalam database. Selain itu, pengguna hanya bisa menjalankan program yang diizinkan saja, tanpa harus mengakses seluruh database.
4. Mudah dalam pengelolaan : Apabila terdapat perubahan sistem, cukup memperbarui program di dalam database tanpa perlu mengubah aplikasi yang menggunakannya.
5. *Reusable* : *Stored program* dapat digunakan oleh berbagai aplikasi tanpa harus menulis ulang kode, sehingga menghemat waktu dan tenaga dalam pengembangan sistem.

Perangkat lunak database modern menggunakan *stored program* berdasarkan standar terbuka yang ditetapkan oleh ANSI dan ISO (Michels et al., 2018). Artinya, pengguna dapat membuat *stored program* dengan sintaks yang serupa di berbagai perangkat lunak database yang ada di pasaran saat ini, tanpa perlu banyak melakukan penyesuaian. Selain itu, *stored program* juga memungkinkan pengembang untuk membuat logika pemrograman langsung di dalam database, seperti: perulangan, percabangan, penggunaan variabel lokal, dan lain sebagainya.

5.1 Persiapan *Sample* Data

Pada bab ini, kita akan mempelajari berbagai konsep terkait dengan *stored program* dan cara menggunakannya dalam database. Agar lebih mudah dipahami, kita akan menggunakan contoh data nyata. Sebagai bahan latihan, kita akan bekerja dengan tabel *products* yang berisi informasi tentang berbagai produk. **Error! Reference source not found.** di bawah ini menunjukkan struktur tabel yang akan kita gunakan untuk menyimpan data produk.

Tabel 5.1. Deskripsi tabel products.

Kolom	Tipe Data	Deskripsi
id	INT	Pengenal unik untuk setiap produk, digunakan sebagai kunci utama (Primary Key) dan dibuat secara otomatis.
name	VARCHAR(50)	Nama produk dengan panjang maksimum 50 karakter.
price	DECIMAL(10,2)	Disimpan sebagai nilai desimal dengan maksimal 10 digit, termasuk 2 angka di belakang koma (misalnya: 9999999.99).
stock	INT	Jumlah item yang tersedia dalam stok untuk produk tersebut.
category	VARCHAR(30)	Kelompok atau jenis produk, dengan panjang maksimum 30 karakter.

Tabel *products* akan diisi dengan lima contoh produk sebagai data sampel. Data ini akan digunakan untuk membantu dalam memahami konsep dan implementasi *stored program* secara lebih praktis. Gambar 5. di bawah merupakan contoh isi dari tabel produk yang akan digunakan.

```
+----+-----------+-------+-------+-------------+
| id | name      | price | stock | category    |
+----+-----------+-------+-------+-------------+
|  1 | Laptop    |   800 |    10 | Electronics |
|  2 | Phone     |   500 |    15 | Electronics |
|  3 | Keyboard  |    50 |    20 | Accessories |
|  4 | Mouse     |    30 |    25 | Accessories |
+----+-----------+-------+-------+-------------+
```

Gambar 5.1. Data pada tabel products

5.2 Stored Procedures

Stored procedure adalah kumpulan perintah SQL yang disimpan di dalam database dengan nama tertentu. Dengan *stored procedure*, kita bisa menjalankan perintah yang sama berulang kali dengan hanya dengan memanggil nama prosedurnya saja (W3Schools, 2025). *Stored procedure* juga mendukung parameter

input dan output, yang membuatnya lebih fleksibel dalam mengolah data (Hoffer et al., 2016). Berikut ini adalah sintaks dasar untuk membuat *stored procedure*:

```
CREATE PROCEDURE procedure_name(parameter_1, parameter_2,
parameter_n)
BEGIN
  -- SQL statement will be executes
END
```

Sintaks yang tersebut di atas dapat dijelaskan sebagai berikut:

1. CREATE PROCEDURE: Perintah untuk membuat *stored procedure* baru dalam database.
2. Procedure_name : Nama unik diberikan pada prosedur, yang akan digunakan saat memanggilnya. Nama prosedur bersifat unik.
3. Parameter_1, parameter_2, parameter_n : Parameter yang digunakan dalam prosedur, dengan tiga jenis:
   a. IN : Digunakan untuk memasukkan data ke dalam prosedur.
   b. OUT : Digunakan untuk menghasilkan output dari prosedur.
   c. INOUT : Bisa digunakan untuk input dan output sekaligus.
4. BEGIN ... END : Bagian utama dari prosedur yang berisi perintah SQL yang akan dijalankan. Semua logika dan struktur perintah ditulis di dalamnya.

Sebagai contoh, kita akan menerapkan *stored procedure* untuk menghitung harga sebuah produk setelah dikenai diskon. Prosedur ini akan menerima harga awal dan persentase diskon sebagai parameter input dan mengembalikan harga setelah diskon sebagai parameter output.

```
CREATE PROCEDURE CalculateTotalPayment(IN price INT,
                    IN discount INT,
                    OUT total INT)
BEGIN
  -- Calculate the total payment after discount
```

```
    SET total = price - (price * discount / 100);
END;
```

Pada prosedur CalculateTotalPayment()diatas, terdapat tiga parameter utama, yaitu: price sebagai parameter input yang merepresentasikan harga barang sebelum diskon; discount sebagai parameter input yang menunjukkan persentase diskon yang akan diberikan; dan total sebagai parameter output yang mengembalikan harga akhir setelah diskon diberikan. Total harga dihitung di dalam tubuh *stored procedure* dengan menggunakan pernyataan: "SET total = price - (price * discount / 100);" akan menentukan harga akhir dengan mengurangkan jumlah diskon dari harga asli.

Contoh *stored procedure* diatas dapat dipanggil dengan cara tertentu, yaitu dengan menyebutkan nama prosedur, diikuti oleh parameter yang diperlukan. Berikut contoh implementasi pemanggilannya:

```
CALL CalculateTotalPayment(200000, 10, @result);
SELECT @result AS TotalPayment;
```

Ketika *stored procedure* CalculateTotalPayment() dieksekusi, parameter 200000 merepresentasikan harga barang, sementara 10 menandakan persentase diskon. Jumlah total setelah menerapkan diskon akan disimpan dalam variabel @result, yang kemudian ditampilkan menggunakan pernyataan SELECT dengan alias TotalPayment.

5.3 Functions

Dalam hal sintaks dan struktur penulisan, *function* (fungsi) dan *stored procedure* memiliki beberapa kemiripan (Yu, 2016), tetapi terdapat beberapa hal utama yang membedakan keduanya, yaitu:

1. Parameter : Fungsi hanya mendukung parameter IN, sehingga tidak dapat menggunakan OUT atau INOUT untuk mengembalikan nilai melalui parameter.
2. *Return value* : Fungsi harus selalu mengembalikan satu nilai tunggal, dan tipe datanya harus didefinisikan dalam deklarasi fungsi.

3. Pemanggilan : Fungsi dapat dipanggil langsung dalam pernyataan SQL seperti SELECT, WHERE, atau JOIN, berbeda dengan *stored procedure* yang dipanggil secara terpisah.

Pembuatan fungsi dalam database memerlukan sintaks dan struktur dasar sebagai berikut:

```
CREATE FUNCTION function_name(parameter_1, parameter_2, parameter_n)
RETURNS data_type
BEGIN
  -- SQL statement will be executes
  <SQL Instruction>
  RETURN <value>;
END;
```

Sintaks dasar untuk membuat fungsi dalam database terdiri dari beberapa elemen utama, yaitu:

1. CREATE FUNCTION : Digunakan untuk mendefinisikan fungsi baru.
2. function_name : Nama fungsi yang akan digunakan untuk pemanggilan.
3. parameter_1, parameter_2, parameter_n: Daftar parameter yang semuanya harus bertipe IN.
4. RETURNS data_type : Menentukan tipe data dari nilai yang akan dikembalikan oleh fungsi saat dieksekusi.
5. BEGIN ... END : Blok utama tempat logika fungsi dituliskan.
6. Komponen-komponen utama dalam tubuh fungsi meliputi:
   a. <SQL Instruction> : Bagian ini dapat berisi operasi perhitungan, manipulasi data, atau proses lainnya.
   b. RETURN <value> : Pernyataan ini menentukan nilai yang akan dikembalikan oleh fungsi, yang harus sesuai dengan tipe data yang telah ditentukan dalam RETURNS.

Sebagai contoh, berikut ini kita akan membuat sebuah fungsi untuk menghitung harga setelah diskon. Melalui implementasi fungsi dibawah ini, kita dapat membandingkannya dengan *stored procedure*, dan melihat

perbedaannya dalam cara pemanggilan dan penggunaannya di dalam SQL.

```
CREATE FUNCTION CalculateDiscountedPrice(price INT, discount INT)
RETURNS INT
BEGIN
  -- Calculate the discounted price
  RETURN price - (price * discount / 100);
END;
```

Fungsi CalculateDiscountedPrice() dibuat untuk menghitung harga akhir suatu produk setelah diskon diberikan. Fungsi ini membutuhkan dua buat parameter input, yaitu: price INT, yang menerima harga item sebagai bilangan bulat; dan discount INT, yang menerima persentase diskon sebagai bilangan bulat. Kata kunci RETURNS menentukan bahwa fungsi ini akan mengembalikan nilai bertipe INT, yaitu harga akhir setelah diskon diberikan. Perhitungan diskon dilakukan dalam blok BEGIN ... END, dan nilai yang dihasilkan dikembalikan menggunakan pernyataan RETURNS, dengan rumus: price - (price * discount / 100).

Dengan menggunakan fungsi, kita dapat langsung memanggilnya dalam pernyataan SELECT, WHERE, atau JOIN, sehingga lebih fleksibel dibandingkan dengan *stored procedure*, yang biasanya harus dipanggil secara terpisah. Sebagai contoh, dibawah ini kita menjalankan pernyataan SELECT untuk menghitung harga setelah diskon:

```
SELECT CalculateDiscountedPrice(1000, 10) AS DiscountedPrice;
```

Setelah perintah SELECT dieksekusi, fungsi CalculateDiscountedPrice() akan menghitung harga akhir setelah menerapkan diskon 10% dari harga awal 1000. Hasil ini kemudian akan ditampilkan dalam kolom DiscountedPrice, sesuai dengan alias yang diberikan dalam pernyataan SELECT.

```
+----------------+
| DiscountedPrice|
+----------------+
|             900 |
+----------------+
```

5.4 Operasi SQL Dalam Tubuh *Stored Program*

Pada bagian sebelumnya, kita telah membahas bahwa di dalam tubuh *stored program*, baik itu *function* ataupun *stored procedure*, dimungkinkan untuk mengeksekusi berbagai pernyataan SQL, termasuk SELECT, INSERT, UPDATE, DELETE, dan banyak lagi. Pada bagian ini, akan dijelaskan contoh operasi SQL di dalam tubuh *stored program*, yang menggambarkan bagaimana perintah-perintah SQL ini dapat digunakan untuk memanipulasi data secara langsung di dalam *function* atau *stored procedure*. Metode ini memungkinkan otomatisasi operasi basis data yang kompleks, meningkatkan efisiensi dan meminimalkan kebutuhan untuk kueri terpisah untuk dapat dieksekusi di luar *stored program*.

Berikut adalah contoh *stored procedure* yang digunakan untuk memperbarui jumlah stok suatu produk. Prosedur ini memerlukan ID produk dan jumlah stok baru sebagai input, sehingga database dapat langsung mengubah data sesuai kebutuhan.

```
CREATE PROCEDURE UpdateStockByID(product_id INT,
new_stock INT)
BEGIN
  -- Update stock for the product with the specified ID
  UPDATE products
  SET stock = new_stock
  WHERE id = product_id;
END;
```

Dalam *stored procedure* UpdateStockByID(), terdapat dua parameter utama, yaitu: product_id yang merupakan ID produk yang stoknya ingin diperbarui; dan new_stock yang merupakan jumlah stok baru yang akan diterapkan pada produk tersebut. Di

dalam prosedur, perintah UPDATE digunakan untuk mengubah nilai stock pada tabel produk berdasarkan product_id yang diberikan. Sebelum prosedur ini dijalankan, tabel products masih berisi data stok lama yang belum diperbarui.

id	name	price	stock	category
1	Laptop	800	10	Electronics
2	Phone	500	15	Electronics
3	Keyboard	50	20	Accessories
4	Mouse	30	25	Accessories

Stored procedure UpdateStockByID()dapat dijalankan dengan perintah berikut:

```
CALL UpdateStockByID(3, 120);
```

Setelah prosedur UpdateStockByID() dijalankan, dengan parameter ID = 3 dan nilai stok baru = 120, maka jumlah stok produk tersebut akan diperbarui dalam database. Sehingga data produk dengan ID = 3 akan menjadi sebagai berikut.

id	name	price	stock	category
3	Keyboard	50	120	Accessories

5.5 Penggunaan Variable

Dalam pembuatan *stored program*, termasuk *function* dan *stored procedure*, variabel berguna dalam menyimpan nilai sementara dan melakukan perhitungan. Cara mendeklarasikan variabel dalam *stored program* adalah sebagai berikut:

```
DECLARE variable_name data_type [DEFAULT value];
```

Deklarasikan variabel dalam *stored program* diatas dapat dijelaskan sebagai berikut:

1. Variable_name: Nama yang diberikan untuk variabel.

2. Data_type: Jenis data yang dapat disimpan, seperti INT, DECIMAL, atau VARCHAR.
3. DEFAULT value (opsional): Nilai awal yang akan diberikan, jika tidak ada nilai lain yang diberikan saat deklarasi.

Berikut adalah contoh sederhana *stored program* menggunakan variabel lokal untuk menghitung harga produk setelah ditambahkan pajak.

```
CREATE FUNCTION CalculatePriceWithTax(price DECIMAL(10, 2))
RETURNS DECIMAL(10, 2)
BEGIN
  -- Declare a local variable for tax rate
  DECLARE tax_rate DECIMAL(5, 2);

  -- Set the tax rate
  SET tax_rate = 0.10;

  -- Return the price after adding tax
  RETURN price + (price * tax_rate);
END;
```

Fungsi CalculatePriceWithTax() memiliki satu parameter, yaitu price, yang mewakili harga awal produk sebelum pajak. Di dalam fungsi, variabel lokal tax_rate dideklarasikan dan diberi nilai 10%, atau 0.10 menggunakan pernyataan SET. Fungsi ini kemudian menghitung harga akhir dengan menambahkan pajak berdasarkan tax_rate dan mengembalikan hasil perhitungannya. Untuk menghitung harga produk setelah dikenai pajak, kita dapat mengimplementasikan fungsi CalculatePriceWithTax() dalam pernyataan SQL SELECT seperti contoh di bawah ini :

```
SELECT id, name, price, CalculatePriceWithTax(price) AS PriceWithTax
FROM products;
```

Setelah pernyataan SQL tersebut diatas dieksekusi, hasilnya akan menampilkan kolom tambahan bernama PriceWithTax, yang menunjukkan harga setelah pajak. Dalam

contoh ini, misalnya harga produk Laptop yang memiliki harga awal 800, dan pajak yang diberikan adalah 10%, maka harga setelah pajak menjadi 880. Outputnya akan terlihat seperti berikut:

```
+----+-----------+-------+--------------+
| id | name      | price | PriceWithTax |
+----+-----------+-------+--------------+
|  1 | Laptop    |   800 |       880.00 |
|  2 | Phone     |   500 |       550.00 |
|  3 | Keyboard  |    50 |        55.00 |
|  4 | Mouse     |    30 |        33.00 |
+----+-----------+-------+--------------+
```

5.6 Conditional Statement

Conditional statement (pernyataan bersyarat) adalah struktur kontrol dalam SQL yang dapat memfasilitasi eksekusi blok kode tertentu berdasarkan suatu kondisi. Pernyataan ini berguna dalam pengambilan keputusan dengan cara melakukan evaluasi terhadap keadaan tertentu. Dua jenis pernyataan bersyarat yang sering digunakan adalah IF ELSE dan CASE.

Bab ini akan mengulas penggunaan pernyataan bersyarat menggunakan IF ELSE, karena lebih mudah dipahami dalam menyampaikan konsep terkait. Secara teknis, IF ELSE dapat menggantikan CASE, tetapi tidak selalu berlaku sebaliknya. Struktur IF ELSE juga cocok digunakan dalam berbagai skenario, baik yang sederhana maupun kompleks.

Berikut ini contoh fungsi yang menggunakan pernyataan IF ELSE untuk menentukan pemberian diskon berdasarkan harga suatu produk. Kondisi yang diberikan adalah jika harga produk lebih besar atau sama dengan 500, maka akan diberikan diskon 20%. Jika harga kurang dari 500, maka akan diberikan diskon sebesar 10%.

```
CREATE FUNCTION CalculateDiscount(price DECIMAL(10, 2))
RETURNS DECIMAL(10, 2)
BEGIN
  -- Declare a variable to store the discount
  DECLARE discount_rate DECIMAL(5, 2);
```

```
  -- Apply conditional logic using IF ELSE
  IF price >= 500 THEN
    SET discount_rate = 0.20; -- 20% discount for price >= 500
  ELSE
    SET discount_rate = 0.10; -- 10% discount for price < 500
  END IF;

  -- Return the discounted price
  RETURN price - (price * discount_rate);
END;
```

Pada fungsi CalculateDiscount()di atas, terdapat satu buah parameter saja, yaitu price. Parameter ini mewakili harga produk dan digunakan sebagai input. Setelah dijalankan, fungsi ini memeriksa variabel price dengan pernyataan IF ELSE. Jika harga produk adalah lebih besar atau sama dengan 500, maka akan diberikan diskon 20%. Jika tidak, maka diberikan diskon 10%. Fungsi ini kemudian mengembalikan harga akhir setelah diskon dihitung dengan rumus: "price - (price * discount_rate)".

Berikut ini adalah isi tabel *products* yang akan digunakan untuk mengilustrasikan penerapan fungsi CalculateDiscount(). Dimana ia fungsi tersebut akan menghitung diskon berdasarkan harga produk itu sendiri.

id	name	price	stock	category
1	Laptop	800	10	Electronics
2	Phone	500	15	Electronics
3	Keyboard	50	20	Accessories
4	Mouse	30	25	Accessories

Untuk menghitung harga setelah diskon, kita menggunakan fungsi CalculateDiscount() dalam kueri SQL. Berikut ini adalah contoh implementasi fungsi tersebut dalam pernyataan SELECT:

```
SELECT id, name, price, CalculateDiscount(price) AS DiscountedPrice
FROM products;
```

Eksekusi perintah SQL SELECT diatas akan menghasilkan kueri dengan kolom tambahan bernama DiscountedPrice, yang menampilkan harga setelah diskon. Berikut adalah contoh tampilan tabel yang dihasilkan dari perintah SQL tersebut:

```
+----+-----------+-------+-----------------+
| id | name      | price | DiscountedPrice|
+----+-----------+-------+-----------------+
|  1 | Laptop    |  800  |          640.00 |
|  2 | Phone     |  500  |          400.00 |
|  3 | Keyboard  |   50  |           45.00 |
|  4 | Mouse     |   30  |           27.00 |
+----+-----------+-------+-----------------+
```

5.7 Loops

Dalam pemrograman, *loop* (perulangan) berfungsi sebagai struktur kontrol yang memungkinkan eksekusi berulang dari sebuah blok kode selama kondisi yang ditentukan bernilai benar. Hal ini sangat berguna dalam pemrograman untuk tugas yang memerlukan pengulangan, seperti memproses data atau menghitung sesuatu berkali-kali. Dengan menggunakan teknik perulangan, kita bisa menghindari penulisan kode yang berulang-ulang, sehingga membuat program lebih efisien dan mengurangi kemungkinan kesalahan.

Dalam database, terdapat beberapa jenis teknik perulangan seperti WHILE, REPEAT UNTIL, dan LOOP. Bab ini akan fokus pada perulangan WHILE karena lebih sederhana dan mudah digunakan, sehingga cocok bagi pemula dalam memahami konsep perulangan.

Sekarang kita akan melihat contoh *stored procedure* yang di dalamnya menggunakan teknik perulangan WHILE untuk keperluan mengubah stok barang dalam kategori tertentu berdasarkan nilai yang diberikan sebagai parameter.

```
DELIMITER //

CREATE PROCEDURE UpdateCategoryStock( IN category_name
VARCHAR(50),
                    IN stock_increment INT)
BEGIN
  DECLARE current_id INT DEFAULT 1;
```

```
  DECLARE max_id INT;

  -- Get the maximum ID from the products table
  SELECT MAX(id) INTO max_id FROM products;

  WHILE current_id <= max_id DO
    -- Update the stock for products in the specified category
    UPDATE products
    SET stock = stock + stock_increment
    WHERE id = current_id AND category = category_name;

    -- Increment the current_id
    SET current_id = current_id + 1;
  END WHILE;
END; //

DELIMITER ;
```

Pada kode SQL diatas, *stored procedure* UpdateCategoryStock() memiliki dua input parameter, yaitu category_name untuk menentukan kategori produk yang stoknya akan diperbarui dan stock_increment untuk jumlah stok yang akan ditambahkan. Di dalamnya, terdapat dua variabel lokal: current_id, yang menyimpan ID produk yang sedang diproses, dan max_id, yang berisi ID terbesar dalam tabel produk. *Stored procedure* UpdateCategoryStock() dapat dieksekusi sebagai berikut:

```
CALL UpdateCategoryStock('Accessories', 5);
```

Saat dieksekusi, perulangan WHILE dalam prosedur UpdateCategoryStock() akan berjalan mulai dari ID produk pertama, dengan current_id diatur ke 1, hingga mencapai ID tertinggi yang disimpan dalam max_id. Setiap kali perulangan berjalan, prosedur akan mengecek apakah produk dengan ID tersebut termasuk dalam kategori yang ditentukan. Jika sesuai,

stok produk akan diperbarui sesuai dengan jumlah yang ditentukan. Dibawah ini merupakan isi data pada tabel products sebelum prosedur UpdateCategoryStock() dijalankan:

id	name	price	stock	category
1	Laptop	800	10	Electronics
2	Phone	500	15	Electronics
3	Keyboard	50	20	Accessories
4	Mouse	30	25	Accessories

Setelah prosedur dijalankan, stok untuk setiap produk dalam kategori "Accessories" akan bertambah 5, sehingga produk seperti Keyboard dan Mouse akan mengalami perubahan. Sementara itu, produk dalam kategori "Electronics " tidak akan terpengaruh karena tidak termasuk dalam kategori yang ditentukan. Berikut adalah data terbaru dalam tabel products setelah pembaruan dilakukan melalui prosedur UpdateCategoryStock().

id	name	price	stock	category
1	Laptop	800	10	Electronics
2	Phone	500	15	Electronics
3	Keyboard	50	25	Accessories
4	Mouse	30	30	Accessories

5.8 Cursor

Cursor adalah fitur dalam database modern yang memungkinkan pemrosesan hasil kueri satu baris dalam satu waktu (Carlos Coronel, 2018). *Cursor* dapat digunakan baik dalam *function* maupun *stored procedure*, terutama ketika operasi data tidak bisa diselesaikan hanya dengan satu pernyataan SQL, seperti saat diperlukan perintah SQL dengan logika yang lebih kompleks. Dibawah ini ini adalah sintaks dasar yang digunakan dalam kursor (Oracle, 2016):

```
DECLARE cursor_name CURSOR FOR <query>;
OPEN cursor_name;
FETCH NEXT FROM cursor_name INTO <variables>;
CLOSE cursor_name;
```

Sintaks dasar penggunaan *cursor* diatas dapat dijelaskan sebagai berikut:

1. Deklarasi (DECLARE): Menentukan *cursor* dan kueri yang akan digunakan untuk mengambil data.
2. Pembukaan (OPEN): Membuka *cursor*, menjalankan kueri, dan menyimpan hasilnya.
3. Mengambil Data (FETCH): Mengambil data dari hasil kueri satu baris dalam satu waktu.
4. Menutup (CLOSE): Menutup *cursor* setelah selesai digunakan untuk melepaskan sumber daya.
5. Dealokasi (DEALLOCATE): Menghapus *cursor* dari memori, meskipun pada beberapa database langkah ini bersifat opsional.

Selanjutnya, kita akan melihat bagaimana *cursor* digunakan dalam *stored procedure*. Prosedur ini dirancang untuk menampilkan nama produk dengan status "Out of Stock " jika stoknya adalah nol. *Cursor* digunakan untuk membaca data dari tabel produk secara berurutan dan memproses item yang kehabisan stok.

```
DELIMITER //

CREATE PROCEDURE ShowOutOfStockProducts()
BEGIN
  DECLARE p_name VARCHAR(100);
  DECLARE p_stock_status VARCHAR(20);
  DECLARE finished INT DEFAULT 0;

  -- Define a cursor to select name of products where stock is 0
  DECLARE cur CURSOR FOR SELECT name, 'Out of Stock' FROM
products
              WHERE stock = 0;

  -- Open the cursor
  OPEN cur;
```

```
  -- Fetch the first row
  FETCH cur INTO p_name, p_stock_status;

  -- Loop through the cursor using WHILE
  WHILE finished = 0 DO
    -- Output the product name and stock status
    SELECT p_name AS Name, p_stock_status AS StockStatus;

    -- Fetch the next row
    FETCH cur INTO p_name, p_stock_status;

    -- Check if there are no more rows
    IF ROW_COUNT() = 0 THEN
      SET finished = 1;
    END IF;
  END WHILE;

  -- Close the cursor
  CLOSE cur;
END //

DELIMITER ;
```

Dalam *stored procedure* ShowOutOfStockProducts() diatas, terdapat tiga variabel lokal yang didefinisikan: p_name untuk menyimpan nama produk, p_stock untuk menyimpan jumlah stok, dan finished untuk menandakan bahwa iterasi sudah selesai. *Cursor* digunakan untuk mengambil nama produk dari tabel yang stoknya kosong atau nol. Jika stok suatu produk = 0, pesan 'Out of Stock akan ditampilkan. Prosedur ini bisa dijalankan dengan perintah berikut:

```
CALL ShowOutOfStockProducts();
```

Saat *stored procedure* ShowOutOfStockProducts() dijalankan, *cursor* akan diinisiasi dengan perintah OPEN cur. Kemudian, data pertama diambil menggunakan FETCH cur INTO p_name, p_stock_status. Perulangan WHILE digunakan untuk

memproses setiap baris yang memenuhi kondisi, yaitu produk dengan stok 0. Setelah setiap FETCH, jika tidak ada data lagi (ditandai dengan ROW_COUNT() = 0), variabel finished diberi nilai 1, yang digunakan untuk mengakhiri perulangan. Setiap kali perulangan, nama produk dan status stok akan ditampilkan dengan pernyataan SELECT. Akhirnya, kursor akan ditutup dengan perintah CLOSE cur.

Sebagai simulasi, misalnya kita membuat stok beberapa produk seperti Keyboard dan Mouse menjadi 0, seperti tabel berikut:

id	name	price	stock	category
1	Laptop	800	10	Electronics
2	Phone	500	15	Electronics
3	Keyboard	50	0	Accessories
4	Mouse	30	0	Accessories

Saat kita eksekusi *stored procedure* ShowOutOfStockProducts(), semua nama produk yang memiliki stok 0 akan ditampilkan bersama status "Out of Stock ". Berikut adalah contoh hasil yang ditampilkan setelah prosedur dieksekusi:

Name	StockStatus
Keyboard	Out of Stock
Mouse	Out of Stock

5.10 Trigger

Dalam database modern, *trigger* adalah fitur yang memungkinkan database menjalankan perintah secara otomatis sebagai respons terhadap kejadian tertentu, seperti penambahan, pembaruan, atau penghapusan pada suatu data. *Trigger* umumnya digunakan untuk menjaga integritas data, mencatat aktivitas, atau melakukan validasi secara otomatis. Namun, penggunaannya perlu dilakukan secara hati-hati, karena jika terlalu banyak *trigger* yang melakukan operasi berat, maka ia dapat menurunkan performa database secara keseluruhan.

Selain itu, *trigger* juga dapat menyulitkan proses *debugging*, karena eksekusinya terjadi di belakang layar. Hal ini membuat pelacakan aktivitasnya menjadi lebih sulit.

Secara umum, karakteristik dan format untuk membuat *trigger* adalah sebagai berikut:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
  -- Trigger Body
  -- Statements to execute when the trigger is invoked
END;
```

Karakteristik dan struktur umum *trigger* dapat dijelaskan sebagai berikut:

1. Waktu Eksekusi:
   a. BEFORE: *Trigger* dijalankan sebelum operasi data (INSERT, UPDATE, DELETE) dieksekusi.
   b. AFTER: *Trigger* dijalankan setelah operasi data (INSERT, UPDATE, DELETE) selesai.
2. Jenis Operasi:
   a. INSERT: *Trigger* dijalankan ketika sebuah baris baru ditambahkan ke tabel.
   b. UPDATE: *Trigger* dijalankan ketika baris yang sudah ada dimodifikasi.
   c. DELETE: *Trigger* dijalankan ketika sebuah baris dihapus dari tabel.
3. *Action*: *Trigger* memungkinkan penerapan logika tertentu, seperti modifikasi data, mencatat aktivitas, atau mencegah perubahan pada data.
4. *Trigger Body*: Bagian ini berisi pernyataan SQL yang mendefinisikan tindakan yang dilakukan oleh *trigger* ketika dijalankan.

Selanjutnya, kita akan melihat contoh *trigger* yang dibuat untuk mencegah stok produk menjadi negatif. *Trigger* ini

memastikan stok produk tidak bisa kurang dari nol. Jika ada upaya untuk mengubah stok menjadi angka negatif, *trigger* akan menolak melanjutkan eksekusi dan menampilkan pesan kesalahan. Berikut adalah contoh penerapannya:

```
DELIMITER //

CREATE TRIGGER PreventNegativeStock
BEFORE UPDATE ON products
FOR EACH ROW
BEGIN
  -- Jika stok baru kurang dari 0, munculkan error
  IF NEW.stock < 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Stock cannot be negative';
  END IF;
END //

DELIMITER ;
```

Kode *trigger* PreventNegativeStock() di atas, akan dijalankan setiap kali ada operasi UPDATE pada tabel *products*. Kondisi IF NEW.stock < 0 akan memeriksa apakah nilai stok baru (NEW.stock) dari record yang sedang diperbarui kurang dari 0. Jika stok baru bernilai negatif, *trigger* akan memberikan pesan kesalahan.

Pernyataan SIGNAL SQLSTATE digunakan untuk memunculkan kesalahan SQL tertentu. Dalam kasus ini, SQLSTATE '45000' adalah kode kesalahan umum yang menunjukkan kesalahan yang diakibatkan oleh pengguna. Ketika *trigger* mendeteksi perintah pengubahan stok menjadi bernilai negatif, perintah UPDATE akan dihentikan, dan pesan kesalahan akan ditampilkan kepada pengguna.

Setelah *trigger* ini dibuat, ia akan berjalan secara otomatis di belakang layar, dan akan beraksi setiap kali ada perintah UPDATE pada tabel produk. Sebagai contoh, jika ada pengguna yang mencoba memperbarui produk dengan ID = 1 dengan mengubah stoknya menjadi -5 melalui perintah sebagai berikut:

```
UPDATE products SET stock = -5 WHERE id = 1;
```

Maka, *trigger* akan otomatis bekerja dengan memeriksa nilai NEW.stock untuk produk dengan ID = 1. Karena stok baru bernilai -5, maka hal tersebut melanggar aturan bahwa stok tidak boleh negatif, *trigger* akan menolak perubahan data dan menampilkan pesan kesalahan. Database menampilkan pesan kesalahan kepada pengguna sebagai berikut:

```
ERROR 1644 (45000): Stock cannot be negative
```

DAFTAR PUSTAKA

Carlos Coronel, S. M. (2018). *Database Systems: Design, Implementation, & Management*. Cengage Learning.

Hoffer, J. A., Ramesh, V., & Topi, H. (2016). *Modern Database Management* (12th Ed.). Pearson.

Michels, J., Hare, K., Kulkarni, K., Zuzarte, C., Liu, Z. H., Hammerschmidt, B., & Zemke, F. (2018). The New and Improved SQL. *ACM SIGMOD Record*, *47*(2), 51–60. https://doi.org/10.1145/3299887.3299897

Oracle. (2016). *Database Programming with PL/SQL: Introduction to Explicit Cursors*. https://academy.oracle.com/

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill Education.

Steven Feuerstein, G. H. (2006). *MySQL Stored Procedure Programming*. O'Reilly Media.

W3Schools. (2025). *Stored Procedure*. https://www.w3schools.com/sql/sql_stored_procedures.asp

Yu, W. (2016). Study on Database Programming and Its Statistics. *Proceedings of the 2nd International Conference on Electronics, Network and Computer Engineering (ICENCE 2016)*. https://doi.org/10.2991/icence-16.2016.172

BAB 6
RANCANGAN DATABASE MENGGUNAKAN MODEL E-R DAN NORMALISASI

Oleh Salsalina Br Sembiring

6.1 Entity-Relationship Model

Sebelum membangun suatu sistem informasi, seorang analis sistem harus terlebih dahulu memahami konsep-konsep desain data yang baik, antara lain struktur data dan karakteristik sistem manajemen basis data. Struktur data adalah kerangka kerja untuk mengatur, menyimpan, dan mengelola data. Struktur data terdiri dari file atau tabel yang berinteraksi dengan berbagai cara. Setiap file atau tabel berisi data tentang orang, tempat, benda, atau peristiwa (Joseph S. Valacich and Joey F. George, 2020).

Pemodelan data merupakan bagian penting dalam menentukan persyaratan sistem, hal ini karena :

1. Karakteristik data diidentifikasikan selama pemodelan data sangat penting untuk merancang basis data, program, dan luaran sistem.
2. Data merupakan aspek yang sangat kompleks pada sistem informasi
3. Karakteristik data (seperti panjang karakter, format, dan keterhubungan antar data) menjadi pembeda bagi organisasi yang memiliki bisnis yang sejenis

Salah satu model yang dapat digunakan untuk pemodelan data ini adalah *Entity-Realtionship* (E-R) Model. Model ini menjadi representasi data dan menampilkan model detail struktur data organisasi. Tujuan pemodelan ini adalah untuk menunjukkan banyaknya aturan tentang arti dan keterhubungan antar data (Joseph S. Valacich and Joey F. George, 2020)

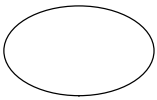
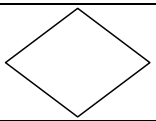
Teknik yang digunakan untuk memodelkan data pada model ini adalah E-R Diagram. E-R Diagram merupakan diagram yang menunjukkan hubungan logis dan interaksi antar entitas sistem serta memberikan gambaran keseluruhan sistem dan cetak biru untuk membuat struktur data fisik(Shelly Cashman, 2019).

6.2 Menggambar E-R Diagram

Dalam model E-R, entitas direpresentasikan sebagai objek yang memiliki atribut, dan hubungan antar entitas di representasikan sebagai keterhubungan yang dinyatakan dengan simbol-simbol grafis.

Adapun notasi yang digunakan dalam E-R Diagram seperti pada gambar dibawah ini :

Tabel 6.1. Notasi E-R Diagram

Notasi	Keterangan
	Entitas merupakan objek yang menjadi fokus dalam suatu database. Entitas bisa berupa manusia, tempat, benda, atau keadaan mengenai data yang dibutuhkan.
	Atribut merupakan informasi yang melengkapi entitas. Sebuah entitas harus memiliki primary keyyang membuat entitas unik dan atribut lainnya yang menjelaskan entitas.
	Relasi merupakan hubungan antara dua entitas atau lebih

Sumber :(Afiifah et al., 2022)

Tahapan dalam menggambar E-R Diagram adalah sebagai berikut :

1. Identifikasi entitas
   Langkah pertama dalam menggambar ERD adalah membuat daftar entitas yang dapat ditemukan selama tahap analisis sistem dan mempertimbangkan sifat hubungan yang menghubungkannya. Entitas dapat berupa orang, tempat,

objek, peristiwa, atau konsep dalam lingkungan pengguna yang datanya ingin disimpan oleh organisasi. Contohnya :

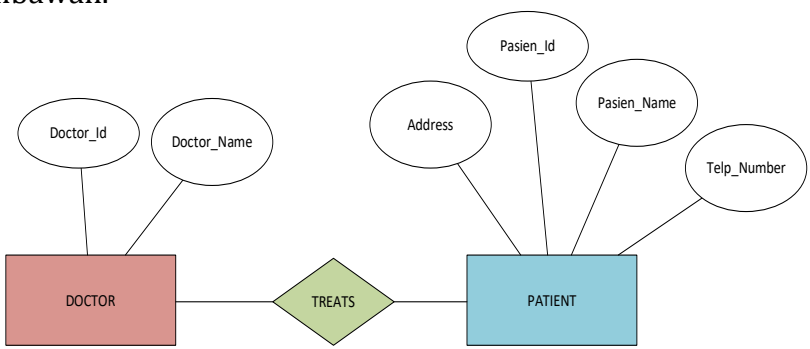
- Orang : KARYAWAN, PASIEN, SISWA
- Tempat : SEKOLAH, TOKO, GUDANG
- Objek : MESIN, BANGUNAN, MOBIL, PRODUK
- Peristiwa : PENJUALAN, PENDAFTARAN
- Konsep : AKUN, KURSUS, PUSAT KERJA

Entitas diberi label dengan kata benda tunggal, dan hubungan antar entitas(relasi) diberi label dengan kata kerja.

Gambar 6.1. Contoh E-R Diagram
Sumber : (Shelly Cashman, 2019)

2. Identifikasi Atribut

Langkah berikutnya setelah entitas-entitas diidentifikasi adalah menentukan atribut-atribut yang terkait dengan entitas yang telah diidentifikasi di tahap sebelumnya. Atribut adalah karakteristik yang menjelaskan entitas(Joseph S. Valacich and Joey F. George, 2020). Misalnya, untuk entitas "Pasien," atribut mungkin mencakup No-Pasien, nama, alamat, nomor telepon, dan sebagainya. Perhatikan contoh dibawah.



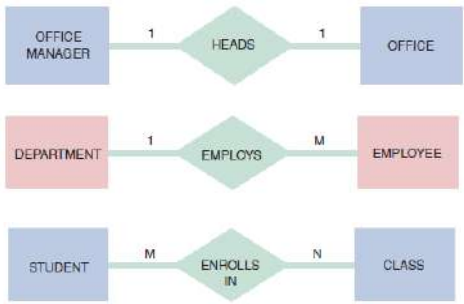
Gambar 6.2. Contoh Atribut Entitas
Sumber : (Shelly Cashman, 2019)

3. Identifikasi Hubungan
   Setelah entitas dan atributnya telah ditentukan, langkah selanjutnya adalah mengidentifikasi hubungan antara entitas-entitas tersebut. Hubungan menggambarkan bagaimana entitas-entitas tersebut berinteraksi di dalam sistem. Ada tiga jenis hubungan antar entitas: satu ke satu (*one to one* disimbolkan dengan 1:1), satu ke banyak (*one to many* disimbolkan dengan 1:M), dan banyak ke banyak (*Many to Many* disimbolkan dengan M:N). Identifikasi hubungan ini penting untuk memahami bagaimana data akan berinteraksi dalam basis data.

Contoh hubungan 1:1

Contoh hubungan 1:M

Contoh hubungan M:N



Gambar 6.3. Contoh hubungan antar entitas
Sumber : (Shelly Cashman, 2019)

Hubungan satu ke satu (1:1) artinya setiap contoh entitas (*instance*) dalam entitas pertama berhubungan dengan tepat hanya satu contoh entitas (*instance*) dalam entitas kedua, dan sebaliknya.
Hubungan satu ke banyak (1:M) menggambarkan hubungan diantara dua entitas dimana satu entitas pada sisi "satu (1)" berhubungan dengan banyak entitas pada sisi "banyak (M)". Artinya bahwa satu contoh entitas dalam entitas pertama dapat terhubung dengan beberapa contoh entitas dalam entitas kedua, tetapi setiap contoh entitas dalam entitas kedua hanya terhubung dengan satu contoh entitas dalam entitas pertama.

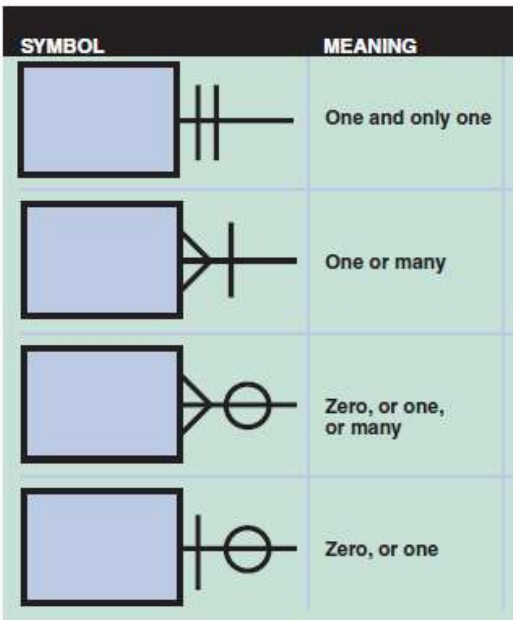
Hubungan banyak ke banyak (M:N) menggambarkan hubungan diantara dua entitas dimana banyak contoh entitas pada sisi "banyak (M)" berhubungan dengan banyak contoh entitas pada sisi "banyak (N)" juga. Ini berarti bahwa banyak contoh entitas dalam entitas pertama dapat terhubung dengan banyak contoh entitas dalam entitas kedua, dan sebaliknya

4. Menentukan kunci primer
   Setelah hubungan antar entitas ditentukan, langkah selanjutnya adalah dengan menentukan kunci utama (*primary key*) setiap entitas. Kunci primer adalah atribut atau kombinasi atribut yang dapat digunakan untuk mengidentifikasi secara unik setiap entitas dalam suatu entitas. Menentukan kunci primer ini penting untuk memastikan integritas data dan akses data yang efisien.
5. Menggambar ERD
   Setelah semua informasi yang diperlukan telah dikumpulkan, Anda dapat mulai menggambar ERD menggunakan simbol-simbol grafis yang sesuai. Anda dapat menggunakan perangkat lunak seperti Microsoft Visio, Lucidchart, atau aplikasi lainnya yang mendukung pembuatan diagram ERD. Dalam E-R Diagram, entitas digambarkan berupa kotak dengan nama entitas berada didalamnya, sementara atribut digambarkan berupa lingkaran atau oval dengan nama atribut juga berada didalamnya, dan hubungan antar entitas digambarkan berupa garis yang menghubungkan antara entitas.
6. Validasi dan Revisi
   Setelah ERD selesai digambar, langkah selanjutnya adalah melakukan validasi dan revisi jika diperlukan. Pastikan ERD menggambarkan struktur data dan hubungan antara entitas yang akurat dalam sistem sesuai dengan yang direncanakan. Dapat juga dapat meminta umpan balik dari pemangku kepentingan lainnya untuk memastikan ERD memenuhi kebutuhan perusahaan yang sesungguhnya

6.3 Kardinalitas

Pada saat analis sistem menggambar ERD, seorang analis harus mendefinisikan hubungan antar entitas dengan lebih rinci. Kardinalitas adalah sebuah teknik yang dapat digunakan untuk merincikan hubungan antar entitas tersebut. Kardinalitas menggambarkan hubungan numerik antara dua entitas dan menunjukkan bagaimana contoh entitas (*instance*) dari satu entitas berhubungan dengan contoh entitas (*instance*) dari entitas lain. Sebagai contoh, perhatikan hubungan antara dua entitas: CUSTOMER dan ORDER. Seorang pelanggan dapat membuat satu pesanan, membuat banyak pesanan, atau sama sekali tidak membuat pesanan, namun setiap pesanan harus memiliki satu dan hanya satu pelanggan.

Seorang analis sistem dapat menggambarkan hubungan ini dengan menggunakan notasi kardinalitas dengan menggunakan simbol-simbol seperti yang terlihat pada gambar 6.4 dibawah ini.



Gambar 6.4. Simbol Kardinalitas
Sumber ((Shelly Cashman, 2019)

Simbol notasi kardinalitas yang umum digunakan disebut notasi kaki gagak karena bentuknya terdiri dari lingkaran, batang, dan simbol kompleksitas. Garis bilah satu menunjukkan satu, garis bilah dua menunjukkan satu dan hanya satu, lingkaran menunjukkan nol, dan garis kaki gagak (simbol kompleksitas) menunjukkan banyak.
Perhatikan contoh dibawah ini :

Pada contoh pertama, satu dan hanya satu CUSTOMER dapat memesan dari nol hingga banyak entitas ORDER.

Pada contoh kedua, satu dan hanya satu ORDER dapat mencakup satu atau banyak ITEM ORDERED.

Pada contoh ketiga, satu dan hanya satu KARYAWAN dapat memiliki satu PASANGAN atau tidak sama sekali.

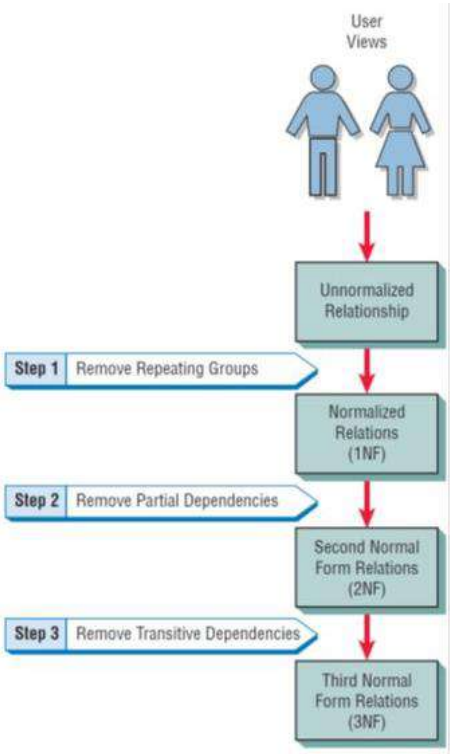
Pada contoh keempat, satu KARYAWAN, atau banyak karyawan, atau tidak sama sekali, dapat ditugaskan pada satu PROYEK, atau banyak proyek, atau tidak sama sekali. (Shelly Cashman, 2019)

6.4 Normalisasi

Normalisasi adalah proses perancangan basis data yang bertujuan untuk mengatur tabel-tabel dalam sebuah basis data agar

meminimalkan redundansi data dan memastikan keutuhan (integritas) data. Tujuan utama dari normalisasi adalah untuk mengatur data agar tidak terjadi redudansi data sehingga analis dapat menggunakan data dengan lebih efisien (Shelly Cashman, 2019).

Proses normalisasi biasanya melibatkan empat tahap: bentuk tidak normal (UNF-unnormalized), bentuk normal pertama (1NF-first normal form), bentuk normal kedua (2NF-*second normal form*), dan bentuk normal ketiga (3NF-*third normal form*). Bentuk normalisasi ketiga ini mewakili rancangan terbaik. Kebanyakan database harus dirancang dalam bentuk normal ketiga. Perhatikan bahwa ada bentuk normal di luar 3NF, tetapi jarang digunakan dalam sistem berorientasi bisnis. Persyaratan untuk tahapan normalisasi ini dapat dilihat pada gambar dibawah :



Gambar 6.5. Tahapan Normalisasi
Sumber : (Kenneth E. Kendall and Julie E. Kendall, 2014)

Tahap pertama dari tabel yang tidak normal akan dilakukan normal pertama dengan cara menghilangkan semua kelompok berulang dan menidentifikasi kunci utama. setelah tabel berada di normal pertama maka akan dilakukan tahap berikutnya agar memenuhi normalisasi kedua dengan cara mengubah ketergantungan parsial. Artinya semua atribut yang bukan kunci sepenuhnya bergantung pada kunci utama. Selanjutnya tabel dipastikan memenuhi normalisasi ketiga dengan cara mengubah ketergantungan transitif, artinya sesuatu atribut yang bukan kunci tidak boleh bergantung pada atribut yang bukan kunci lainnya.

6.4.1 Format Notasi Standar

Mendesain tabel akan lebih mudah jika format notasi standar digunakan untuk memperlihatkan struktur tabel, field (atribut), dan kunci utama. Format notasi standar dimulai dengan nama tabel, diikuti dengan tanda kurung yang berisi nama atribut yang dipisahkan dengan koma. Atribut kunci utama digarisbawahi, sebagai berikut:

NAMA (<u>ATRIBUT 1</u>, ATRIBUT 2, ATRIBUT 3, ...)

Selama desain data, analis harus mampu mengenali sekelompok atribut yang berulang. Kelompok berulang adalah sekumpulan satu atau lebih atribut yang dapat muncul beberapa kali dalam satu baris, dengan setiap kemunculan memiliki nilai berbeda.

Contoh dari grup berulang ditunjukkan pada Gambar 6.5 dibawah. Dua penjualan sales berisi beberapa item, yang merupakan kelompok berulang dalam SALESPERSON_NUMBER yang sama. Perhatikan baris SALESPERSON-NUMBER, SALESPERSON_NAME, dan SALES_AREA berisi beberapa pengulangan baris untuk CUSTOMER_NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION, dan SALES_AMOUNT.

Grup yang berulang dapat dianggap sebagai sekumpulan record anak (*subsidiary*) yang terdapat dalam record induk (*main*).

SALESPERSON NUMBER	SALESPERSON NAME	SALES AREA	CUSTOMER NUMBER	CUSTOMER NAME	WAREHOUSE NUMBER	WAREHOUSE LOCATION	SALES AMOUNT
3462	Waters	West	18765	Delta Systems	4	Fargo	13540
			18830	A. Levy and Sons	3	Bismarck	10600
			19242	Ranier Company	3	Bismarck	9700
3593	Dryne	East	18841	R. W. Flood Inc.	2	Superior	11560
			18899	Seward Systems	2	Superior	2590
			19565	Stodola's Inc.	1	Plymouth	8800
etc.							

Gambar 6.6. Contoh LAPORAN PENJUALAN tidak normal
Sumber : (Kenneth E. Kendall and Julie E. Kendall, 2014)

Dalam desain tabel diatas, SALESPERSON-NUMBER memiliki grup berulang yang berisi beberapa CUSTOMER-NUMBER.

SALESPERSON-NUMBER adalah kunci utama untuk tabel LAPORANPENJUALAN, dan CUSTOMER-NUMBER berfungsi sebagai kunci utama untuk grup berulang. Karena berisi grup berulang, tabel LAPORAN-PENJUALAN dinyatakan tidak normal.

Desain tabel yang berisi grup berulang disebut tidak normal. Notasi standar untuk merepresentasikan desain yang tidak normal adalah dengan menempatkan grup atribut yang berulang di dalam tanda kurung. Contoh tabel yang tidak normal terlihat seperti bentuk berikut:

NAMA (ATRIBUT1, ATRIBUT2, ATRIBUT3, (ATRIBUT BERULANG1, ATRIBUT BERULANG2, …))

Jadi untuk tabel LAPORAN_PENJUALAN diatas dapat dinotasikan sebagai berikut :

LAPORAN-PENJUALAN (SALESPERSON NUMBER, SALESPERSON_NAME, SALES_AREA, (CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION, SALES_AMOUNT))

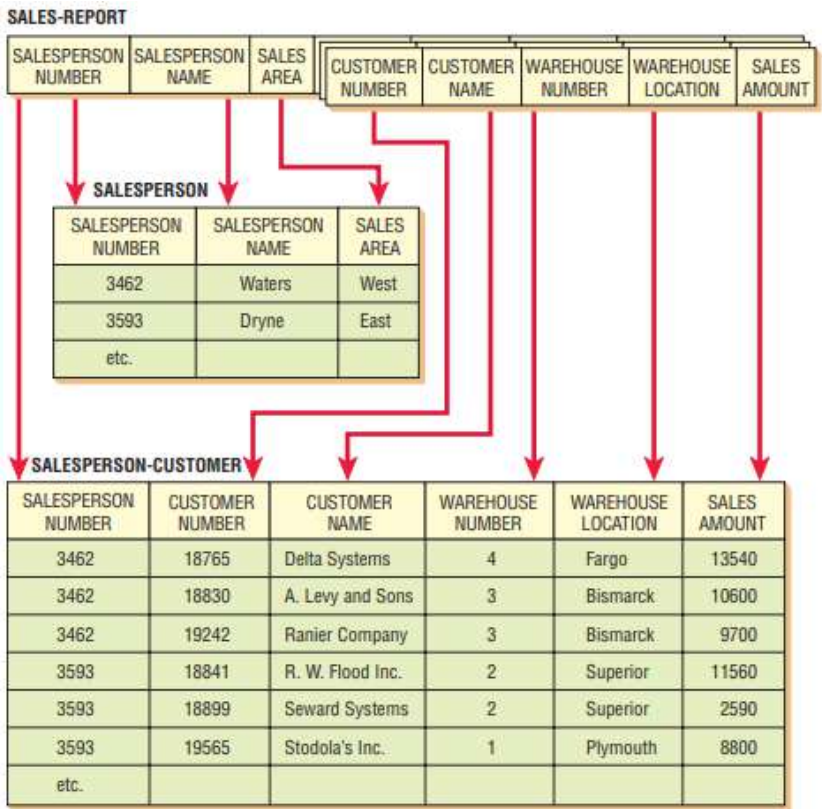
Notasi tersebut menunjukkan bahwa tabel LAPORAN_PENJUALAN berisi delapan kolom. Atribut SALESPERSON NUMBER digarisbawahi untuk menunjukkan bahwa atribut tersebut adalah kunci utama. Atribut CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION, SALES_AMOUNT terdapat dalam tanda kurung untuk menunjukkan

bahwa atribut tersebut merupakan atribut kelompok berulang. Perhatikan bahwa CUSTOMER_NUMBER juga digarisbawahi karena atribut ini bertindak sebagai kunci utama dari kelompok berulang.

6.4.2 Normalisasi Pertama (1NF)

Tabel dinyatakan berada dalam bentuk normal pertama (1NF) jika tidak terdapat kelompok berulang. Untuk mengubah desain yang tidak normal menjadi 1NF, kunci utama tabel harus diperluas agar menyertakan kunci utama dari grup berulang.

Pada contoh tabel di Gambar 6.5 diatas tabel tidak normal LAPORAN_PENJUALAN akan dipecah kedalam dua tabel terpisah. Tabel tersebut akan dinamakan SALESPERSON dan SALESPERSON_CUSTOMER seperti pada gambar dibawah ini :



Gambar 6.7. Contoh normalisasi pertama
Sumber : (Kenneth E. Kendall and Julie E. Kendall, 2014)

Gambar 6.7 diatas menunjukkan tabel tidak normal LAPORAN_PENJUALAN dinormalisasikan dengan cara memisahkan tabel tersebut ke dalam dua tabel baru. Pecahan tabel tersebut yang pertama yaitu tabel SALES yang terdiri dari kunci utama SALESPERSON_NUMBER, serta atributnya yang lain yaitu SALESPERSON_NAME, dan SALES_AREA.

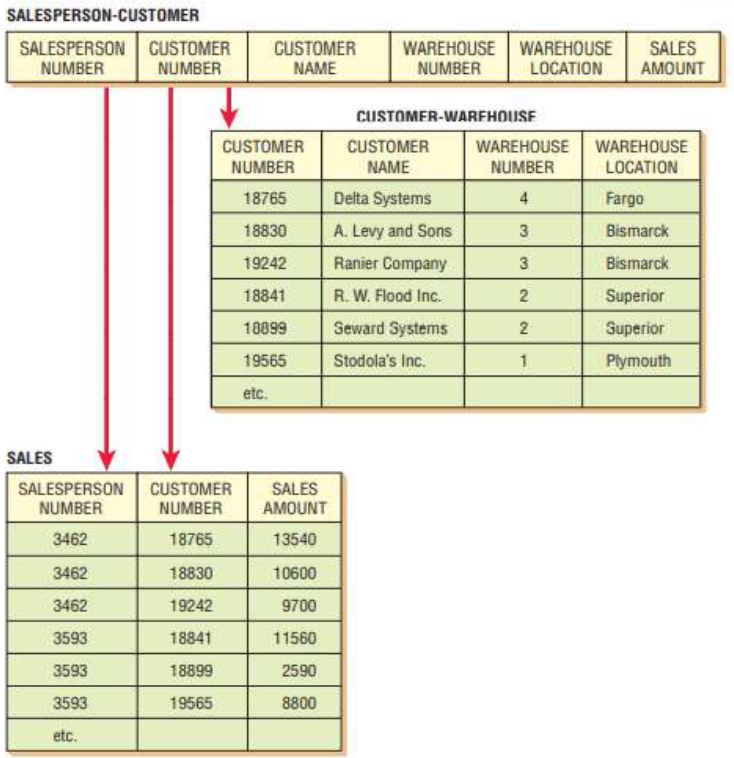
Tabel kedua, SALESPERSON_CUSTOMER, mengandung kunci utama dari tabel SALESPERSON dan semua atribut kelompok berulang dijadikan satu tabel. Dengan mengetahui SALESPERSON_NUMBER, tidak secara otomatis berarti bahwa akan diketahui CUSTOMER_NAME, SALES_AMOUNT, dan WAREHOUSE_LOCATION, oleh sebab itu harus digunakan sebuah kunci gabungan yaitu SALESPERSON_NUMBER dan CUSTOMER_NUMBER untuk dapat mengakses semua informasi. Jika dibuat dalam notasi penulisan narmalisasi sebagai berikut :
SALESPERSON (SALESPERSON NUMBER, SALESPERSON_NAME, SALES_AREA)
SALESPERSON_CUSTOMER (SALESPERSON NUMBER, CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION, SALES_AMOUNT)

Tabel SALESPERSON_CUSTOMER merupakan hubungan normalisasi pertama tetapi belum ideal karena masih ada atribut yang tidak bergantung secara fungsional ke semua kunci dalam tabel yaitu SALESPERSON_NUMBER, CUSTOMER_NUMBER, hanya SALES_AMOUNT yang bergantung secara fungsional ke kunci ini, tetapi untuk CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION hanya bergantung secara fungsional ke CUSTOMER_NUMBER, oleh sebab itu harus dilakukan normalisasi yang kedua.

6.4.3 Normal Kedua (2NF)

Seperti pada dijelaskan pada gambar 6.5, untuk normalisasi kedua dapat dilakukan dengan mengubah ketergantungan parsial yaitu semua atribut akan bergantung secara fungsional pada kunci utama. Hal ini dilakukan dengan cara menghilangkan semua atribut

yang bergantung sebagian dan membentuk tabel baru dalam tabel lain.



Gambar 6.8. Contoh Normal kedua
Sumber : (Kenneth E. Kendall and Julie E. Kendall, 2014)

Seperti pada gambar diatas, terlihat bagaimana tabel SALESPERSON_CUSTOMER dipisahkan ke dalam dua tabel baru. CUSTOMER_WAREHOUSE dan SALES. Jika pakai notasi normalisasi, dapat dituliskan sebagai berikut :
SALES (SALESPERSON NUMBER, CUSTOMER NUMBER, SALES_AMOUNT)
CUSTOMER_WAREHOUSE (CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER, WAREHOUSE_LOCATION)

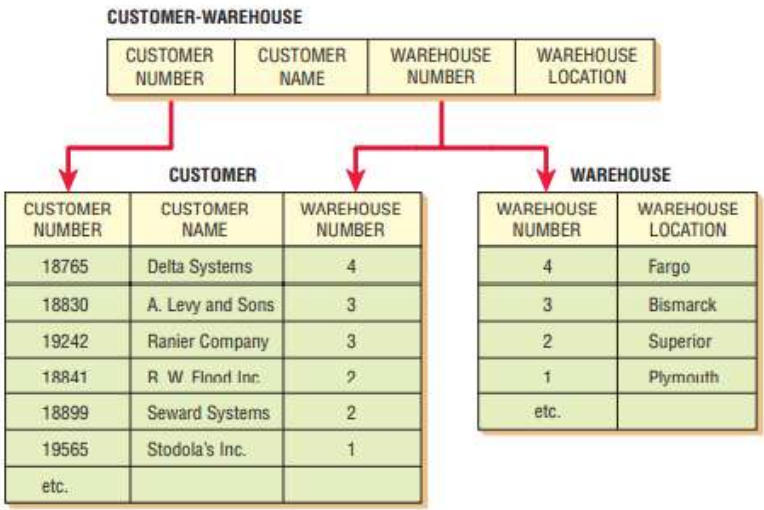
Tabel CUSTOMER-WAREHOUSE berada dalam bentuk normalisasi kedua. Tabel tersebut masih dapat disederhanakan lagi

karena terdapat penambahan ketergantungan dalam tabel. Beberapa atribut bukan kunci tidak hanya bergantung pada kunci utama, tapi juga bergantung pada atribut bukan kunci lainnya. Ketergantungan ini disebut sebagai ketergantungan transitif.

Perhatikan atribut pada tabel CUSTOMER_WAREHOUSE. Semua atribut memang sudah bergantung pada atribut kunci pada tabel, tapi atribut WAREHOUSE_LOCATION tidak hanya bergantung pada kunci tapi juga bergantung pada WAREHOUSE_NUMBER. Oleh sebab itu masih perlu disederhanakan dalam normak ketiga.

6.4.4 Normalisasi Ketiga (3NF)

Normalisasi ketiga dilakukan apabila semua atribut bukan kunci seluruhnya bergantung secara fungsional pada kunci utama dan tidak terdapat ketergantungan transitif. Pada contoh sebelumnya dapat dilakukan normalisasi ketiga dengan menguraikan tabel CUSTOMER-WAREHOUSE kedalam dua tabel seperti pada gambar dibawah ini.



Gambar 6.9. Contoh Normalisasi ketiga
Sumber : (Kenneth E. Kendall and Julie E. Kendall, 2014)

Jadi pakai notasi normalisasi, dapat dituliskan sebagai berikut :

CUSTOMER (CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER)
WAREHOUSE (WAREHOUSE_NUMBER, WAREHOUSE_LOCATION)

Kunci utama untuk tabel CUSTOMER adalam CUSTOMER NUMBER, dan kunci utama untuk WAREHOUSE adalah WAREHOUSE NUMBER.
Akhirnya, tabel yang tidak normal LAPORAN_PENJUALAN telah dilakukan normalisasi sampai normalisasi ketiga menjadi empat tabel. Adapun hasil akhirnya adalah :

SALESPERSON (SALESPERSON_NUMBER, SALESPERSON_NAME, SALES_AREA)
SALES (SALESPERSON NUMBER, CUSTOMER NUMBER, SALES_AMOUNT)
CUSTOMER (CUSTOMER NUMBER, CUSTOMER_NAME, WAREHOUSE_NUMBER)
WAREHOUSE (WAREHOUSE_NUMBER, WAREHOUSE_LOCATION)

DAFTAR PUSTAKA

Afiifah, K.' *et al.* (2022) 'Universitas Negeri Jakarta; Jl. Rawamangun Muka Raya No.11 RW.14 Rawamangun', *JURNAL INTECH*, 3(1), pp. 8–11.

Joseph S. Valacich and Joey F. George (2020) *Modern Systems Analysis and Design*. 9th edn. Pearson Education.

Kenneth E. Kendall and Julie E. Kendall (2014) *SYSTEMS Analysis and design*. Pearson Education Limited.

Shelly Cashman (2019) *System Analysis And Design*. 12th edn. America: Cengage.

BAB 7
RELATIONAL DATABASE DESIGN AND NORMALIZATION

Oleh Hasanudin Jayawardana

7.1 Pendahuluan

Model Relasional didasarkan pada konsep matematika yaitu konsep himpunan, yang secara fisik direpresentasikan sebagai tabel [Connoly, 2015].Relasi adalah tabel yang terbentuk dari kolom dan baris. Kolom dari suatu tabel disebut juga dengan atribut (attribute). Sedangkan Basis Data Relasional merupakan suatu Kumpulan relasi yang dinormalisasi dengan nama relasi yang berbeda..

Gambar 7.1. Struktur Tabel (Sumber : Connolly, 2015)

Elemen-elemen utama dalam Tabel Relasional ":

1. **Tabel (Relasi)** : tempat untuk menyimpan data yang terdiri dari baris dan kolom.
2. **Kolom (Atribut)** : Sifat atau kategori dari entitas yang disimpan dalam tabel.
3. **Baris (Tuple / Record)** : Set data untuk suatu entitas atau objek.

4. **Kunci (Key)** : Elemen penting dalam mengidentifikasi baris dalam tabel.

7.1.1 Konsep kunci dalam Basis data Relasional

Untuk memahami cara kerja basis data relasional, kita perlu memahami berbagai jenis kunci yang diguanakan dalam tabel yaitu :

1. **Primary Key** : Kunci utama yang dipilih untuk menentukan kolom unik (yang isi nya tidak akan yang sama dalam kolom tersebut) dalam suatu tabel. Pada kolom yang menjadi primary key tidak boleh kosong atau tidak ada nilai yang dimasukkan di dalamnya.
2. **Foreign Key** : atribut atau Kumpulan atribut dalam suatu tabel yang digunakan untuk menghubungkan antar tabel (dengan nama atribut / kolom yang sama).
3. **Unique Key** : Kunci yang memastikan bahwa setiap nilai dalam kolom tersebut adalah unik (tidak akan sama satu dengan lainnya), tetapi memungkinkan nilai null.

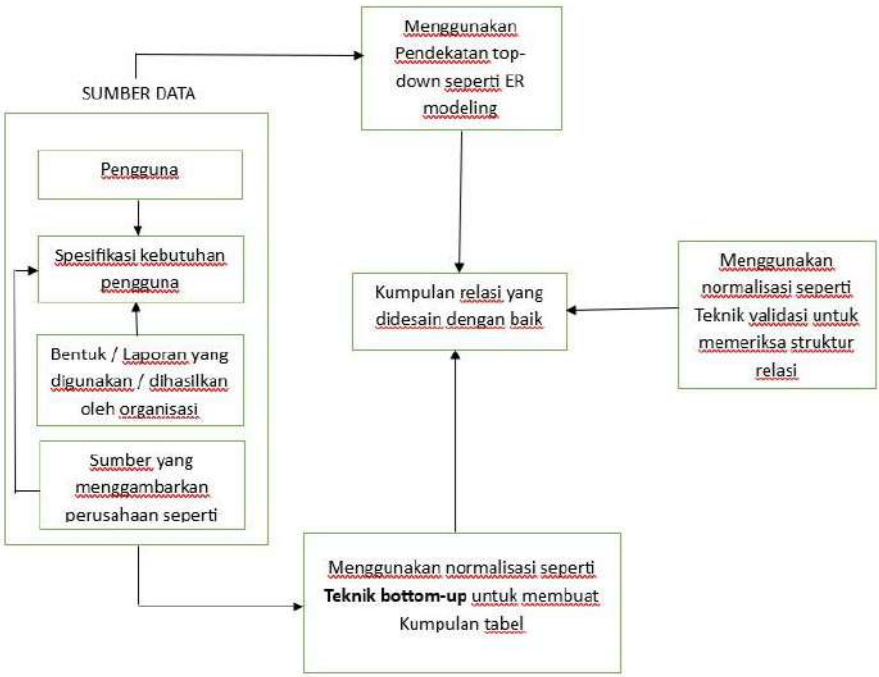
7.2 Normalisasi

Normailsasi adalah Suatu teknik untuk menghasilkan sekumpulan relasi dengan properti yang diinginkan berdasarkan kebutuhan data suatu perusahaan (Connolly, 2015).

Tujuan Normalisasi adalah untuk mengidentifikasi serangkaian tabel yang saling berhubungan satu dengan lainnya. Karakteristik rangkaian tabel-tabel tersebut meliputi :

1. Jumlah minimal atribut yang diperlukan untuk mendukung kebutuhan data.
2. Atribut-atribut yang memiliki hubungan logika yang dekat yang ditemukan dalam relasi yang sama.
3. Meminimalkan redundansi, diman setiap atribut hanya muncul sekali, dengan pengecualian penting untuk atribut yang membentuk seluruh atau sebagian kunci asing (*foreign key*), yang penting untuk penggabungan tabel terkait.

Keuntungan basis data yang telah dinormalisasi adalah basis data tersebut lebih mudah untuk diakses dan dipelihara, dan lebih menghemat penyimpanan dalam media penyimpanan komputer.



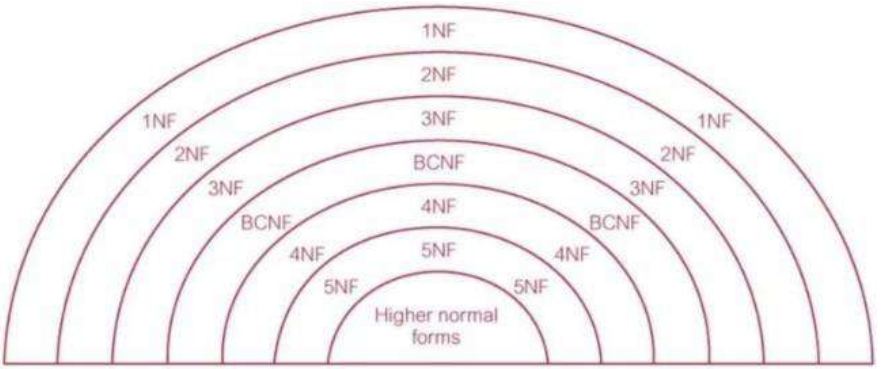
Gambar 7.2. Bagaimana normalisasi dapat digunakan untuk mendukung desain database (Sumber : Connoly, 2015)

7.2.1 Proses Normasilsasi

Normalisasi adalah proses memastikan bahwa setiap tabel memiliki struktur yang logis dan konsisten, sehingga mengurangi kesalahan data dan meningkatkan integitas data (Silberschatz, 2020).

Proses Normalisasi terdiri dari beberapa tahapan antara lain :

1. Bentuk Normal Pertama / First Normal Form (1NF)
2. Bentuk Normal Kedua / Second Normal Form (2NF)
3. Bentuk Normal Ketiga / Third Normal Form (3NF)
4. Bentuk Normal *Boyce Codd / Boyce Codd* Normal Form (BCNF)
5. Bentuk Normal Ke-empat / Fourth Normal Form (4NF)
6. Bentuk Normal Kelima / Fifth Normal Form (5NF)



Gambar 7.3. Ilustrasi diagramatik keterhubungan antara bentuk bentuk Normal (Sumber : Connolly, 2015)

Normalisasi struktur tabel akan mengurangi redundansi data. Jika terjadi perulangan grup, perulangan tersebut harus dihilangkan dengan memastikan bahwa setiap baris mendefinisikan sebuah instance entitas tunggal dan setiap perpotongan baris-kolom hanya memiliki satu nilai [Coronel, 2020]

Tabel yang dinormalisasi menjadi bentuk normal pertama atau First Normal Form adalah tabel yang salah satu atau lebih kolom nya memiliki beberapa 1 nilai atau lebih. Berikut contoh tabel yang belum dinormalisasi dan akan dilakukan normalisasi bentuk pertama.

Syarat bentuk normal Pertama (1NF) adalah nilai atribut nya bersifat atomik (hanya 1 nilai dalam satu kolom).

Tabel 7.1. Penyewaan Mobil

ID pelanggan	Nama pelanggan	No. kendaraan	Tanggal Mulai menyewa	Tanggal Selesai menyewa	harga sewa	Pemilik kendaraan	ID Pemilik kendaraan
3003	Umar	PS 8012	1-Jan-25	7-Jan-25	750000	Usrok	O2309
		PS 8001	1-Jan-25	14-Jan-25	1500000	Ucriet	O2312
		PS 8005	3-Jan-25	5-Jan-25	300000	Ucriet	O2312
3045	Ujang	PS 8005	6-Jan-25	12-Jan-25	750000	Ucriet	O2312
		PS 9012	8-Jan-25	12-Jan-25	400000	Usrok	O2309

Pada tabel di atas, kondisi tabel tersebut dapat disebut belum dinormalisasi, oleh karena itu, dirubah dulu menjadi bentuk Normal Pertama (1NF) di mana setiap atribut memiliki 1 nilai saja.

Tabel 7.2. Penyewaan Mobil (Bentuk Normal Pertama)

ID pelanggan	Nama pelanggan	No. kendaraan	Tanggal Mulai menyewa	Tanggal Selesai menyewa	harga sewa	Pemilik kendaraan	ID Pemilik kendaraan
3003	Umar	PS 8012	1-Jan-25	7-Jan-25	750000	Usrok	O2309
3003	Umar	PS 8001	1-Jan-25	14-Jan-25	1500000	Ucriet	O2312
3003	Umar	PS 8005	3-Jan-25	5-Jan-25	300000	Ucriet	O2312
3045	Ujang	PS 8005	6-Jan-25	12-Jan-25	750000	Ucriet	O2312
3045	Ujang	PS 9012	8-Jan-25	12-Jan-25	400000	Usrok	O2309

7.2.2 *Second Normal Form* (2NF)

Langkah selanjutnya membuat tabel yang telah menjadi bentuk normal pertama menjadi bentuk normal kedua. Syarat untuk menjadi bentuk normal kedua adalah :

1. Tabel telah memenuhi bentuk normal pertama
2. Setiap atribut yang bukan primary key memiliki ketergantungan fungsional pada primary key

Bentuk Normal Kedua mensyaratkan fungsi ketergantungan penuh terhadap primary key, sehingga bila dalam sebuah tabel memiliki 2 kolom yang berpotensi menjadi primary key, maka perlu dilakukan pemisahan tabel menjadi 2 atau lebih tabel baru agar semua kolom yang lain benar-benar terhubung dengan 1 primary key.

Pada tabel yang dinormalisasi pada bentuk normal 1 di atas, maka dapat dinormalisasi ke bentuk normalisasi ke 2. Pada tabel tersebut masih terdapat 2 kolom yang memiliki nilai bersifat unik sehingga perlu dipecah menjadi beberapa tabel baru. Tabel yang diperoleh dari hasil normaliasi ke bentuk Normal Kedua dapat dilihat pada tabel 7-3, tabel 7-4 dan tabel 7-5.

7.2.3 *Third Normal Form* (3NF)

Syarat untuk merubah ke bentuk normal ketiga adalah :

1. Tabel telah memenuhi bentuk normal pertama dan bentuk normal kedua
2. Tidak ada atribut yang bukan primary key tidak bergantung secara transitif pada kunci utama.

Setelah tabel tersebut memenuhi bentuk normal kedua, maka untuk memenuhi bentuk normal ketiga, maka kita harus memeriksa apakah pada tabel tersebut masih memiliki atribut yang bersifat transitive dependency (ketergantungan transitif), bila masih ada maka atribut tersebut perlu dipindahkan dari tabel tersebut dengan menempatkannya dalam tabel yang baru.

Jika kita lihat pada 3 tabel yang dihasilkan dari normalisasi ke-dua, lebih kecil ekungkinan tabel tersebut mengalami redundansi. Akan tetapi dapat terjadi anomali. Misalkan bila kita ingin memperbaharui data pada ID pemilik kendaraan, maka perubahan itu akan berpengaruh pada tabel lainnya dalam database. Anomali pembaharuan ini dikarenakan adanya ketergantungan tansitif (*transitive dependency*), sehingga kita perlu menghilangkan ketergantugan tersebut dengan melanjutkan ke bentuk normal ketiga.

Pada tabel 7.3 dan 7.5, semua atribut yang bukan primary key merupakan atribut yang benar-benar bergantung penuh hanya pada primary key nya.

Tabel 7.3. Penyewaan Mobil (Bentuk Normal Pertama)

ID pelanggan	Nama pelanggan	No. kendaraan	Tanggal Mulai menyewa	Tanggal Selesai menyewa	harga sewa	Pemilik kendaraan	ID Pemilik kendaraan
3003	Umar	PS 8012	1-Jan-25	7-Jan-25	750000	Usrok	O2309
3003	Umar	PS 8001	1-Jan-25	14-Jan-25	1500000	Ucriet	O2312
3003	Umar	PS 8005	3-Jan-25	5-Jan-25	300000	Ucriet	O2312
3045	Ujang	PS 8005	6-Jan-25	12-Jan-25	750000	Ucriet	O2312
3045	Ujang	PS 9012	8-Jan-25	12-Jan-25	400000	Usrok	O2309

Tabel 7.4. Penyewaan

ID Pelanggan	Nomor Kendaraan	Tanggal Mulai Menyewa	Tanggal Selesai menyewa	Harga Sewa
3003	PS 8012	1-Jan-25	7-Jan-25	750000
3003	PS 8001	1-Jan-25	14-Jan-25	1500000
3003	PS 8005	3-Jan-25	5-Jan-25	300000
3045	PS 8005	6-Jan-25	12-Jan-25	750000
3045	PS 9012	8-Jan-25	12-Jan-25	400000

Tabel 7.5. Kendaraan yang disewakan

No kendaraan	Pemilik Kendaraan	ID Pemilik Kendaraan
PS 8012	Usrok	O2309
PS 8001	Ucriet	O2312
PS 8005	Ucriet	O2312
PS 9012	Usrok	O2309

Tabel 7.6. Pelanggan

ID Pelanggan	Nama Pelanggan
3003	Umar
3045	Ujang

Pada tabel 7-3 :
ID pelanggan, No kendaraan -> Tanggal mulai menyewa, Tanggal selesai menyewa, dan harga sewa (atribut ID pelanggan yang merupakan primary key Bersama dengan No kendaraan mempengaruhi atribut tanggal mulai menyewa, tanggal selesai menyewa serta harga sewa)
Pada tabel 7-5 :
ID pelanggan -> Nama pelanggan (atribut nama dipengaruhi oleh ID pelanggan)
Sedangkan pada tabel 7-4 :
No kendaraan -> Pemilik kendaraan dan ID kendaraan
Semua hubungan ketergantungan di atas merupakan ketergantungan penuh terhadap primary key, tetapi pada tabel 6-4 terhadap hubungan ketergantungan transitif yaitu antara atribut pemilik kendaraan dan ID pemilik kendaraan.
Pemilik kendaraan -> ID pemilik kendaraan
Oleh karena itu kita harus menghilangkan ketergantungan tersebut dengan membuat dua tabel baru sebagai berikut :

Tabel 7.7. kendaraan sewa

No kendaraan	ID pemilik kendaraan
PS 8012	O2309
PS 8001	O2312
PS 8005	O2312
PS 9012	02309

Tabel 7.8. Pemilik kendaraan

ID pemilik kendaraan	Pemilik kendaraan
O2309	Usrok
O2312	Ucriet

7.2.4 *Boyce Codd Normal Form* (BCNF)

BCNF didasarkan pada ketergantungan fungsional yang memperhitungkan semua kunci kandidat dalam suatu relasi. Suatu relasi berada dalam BCNF jika dan hanya jika setiap determinan merupakan kunci kandidat

Untuk contoh tabel di atas, tabel telah memenuhi bentuk Boyce Codd sehingga tdiak lagi diperlukan normalisasi ke BCNF.

7.2.5 *Fourth Normal Form* (4NF)

Suatu relasi berada dalam 4NF jika dan hanya jika untuk setiap ketergantungan multinilai nontrivial merupakan kunci kandidat dari relasi tersebut. Syarat tabel dapat dinormalisasikan ke bentuk normal ke-4 adalah :

1. Telah memenuhi bentuk normal ke-4
2. Suatu relasi berada dalam 4NF jika dan hanya jika untuk setiap ketergantungan multinilai nontrivial merupakan kunci kandidat dari relasi tersebut.
   Karena tabel pada contoh di atas telah memenuhi BCNF, maka tidak diperlukan lagi normalisasi ke bentuk Normal Ke-4 atau Fourth Normal Form.

7.2.6 Fifth Normal Form (5NF)

Bentuk Normal ke-lima mengisyaratkan tabel-tabel telah :

1. Memenuhi bentuk normal ke-4
2. Tidak ada kolom yang bergantung pada relasi join

Karena tabel pada contoh di atas telah memenuhi BCNF, maka tidak diperlukan lagi normalisasi ke bentuk Normal Ke-5 atau Fifth Normal Form.

Tidak semua tabel atau database harus dinormalisasi ke bentuk normal ke-5, apabila tabel atau database tidak lagi berpotensi menimbulkan peluang redudansi saat dinormalisasi pada tingkat normal di bawah bentuk normal ke-5, maka tidak diperlukan normalisasi ke bentuk yang lebih tinggi karena telah sempurna.

Dengan demikian, kita dapatkan bentuk normalisasi dari contoh tabel di atas menjadi 4 tabel berikut yang telah dinormalisasi sebagai berikut :

Tabel 7.9. Penyewaan

ID Pelanggan	Nomor Kendaraan	Tanggal Mulai Menyewa	Tanggal Selesai menyewa	Harga Sewa
3003	PS 8012	1-Jan-25	7-Jan-25	750000
3003	PS 8001	1-Jan-25	14-Jan-25	1500000
3003	PS 8005	3-Jan-25	5-Jan-25	300000
3045	PS 8005	6-Jan-25	12-Jan-25	750000
3045	PS 9012	8-Jan-25	12-Jan-25	400000

Tabel 7.10. Kendaraan yang disewakan

No kendaraan	Pemilik Kendaraan	ID Pemilik Kendaraan
PS 8012	Usrok	O2309
PS 8001	Ucriet	O2312
PS 8005	Ucriet	O2312
PS 9012	Usrok	O2309

Tabel 7.11. Pelanggan

ID Pelanggan	Nama Pelanggan
3003	Umar
3045	Ujang

Tabel 7.12. daftar kendaraan sewa

No kendaraan	ID pemilik kendaraan
PS 8012	O2309
PS 8001	O2312
PS 8005	O2312
PS 9012	02309

Tabel 7.13. pemilik kendaraan

ID pemilik kendaraan	Pemilik kendaraan
O2309	Usrok
O2312	Ucriet

DAFTAR PUSTAKA

Connolly, Thomas dkk (2015), *Database Systems A Practical Approach to Design Implementation and Management*, Pearson.

Coronel, Carlos dkk. 2017. *Database Systems : Design, implementation, and Management, Chengage*.

Date, C.J. 2003. *An Introduction to Database Systems*, Addison-Wesley.

Silberschatz, Abraham dkk (2020), *Database System Concepts*, Mc Graw Hill Education

BAB 8
COMPLEX DATA TYPES

Oleh Ridho Surya Kusuma

8.1 Pendahuluan

Dalam pengelolaan data modern, *complex data types* memainkan peran yang semakin penting, terutama mengingat perkembangan pesat dalam teknologi informasi dan komunikasi. Complex data types (tipe data kompleks) merujuk pada struktur data yang dapat menyimpan informasi lebih rumit dibandingkan dengan tipe data sederhana (*primitive types*). Tipe data sederhana, seperti integer, float, dan boolean, hanya dapat menyimpan satu nilai pada satu waktu, sedangkan tipe data kompleks mampu merepresentasikan koleksi nilai, struktur hierarkis, atau bahkan objek yang lebih kompleks. Contohnya termasuk array, record, JSON, XML, dan data spasial, yang memungkinkan penyimpanan data yang lebih terstruktur dan terorganisir.

Signifikansi *complex data types* tidak dapat diabaikan, terutama dengan kemunculan teknologi seperti Internet of Things (IoT), big data, dan *data science*. Misalnya, aplikasi IoT menghasilkan data dari sensor-sensor yang sering kali datang dalam format tidak terstruktur atau semi-terstruktur, seperti JSON dan XML. Format ini memerlukan tipe data kompleks untuk mendukung penyimpanan dan pemrosesan data yang efisien (Shrouf & Miragliotta, 2015; Hu & Shu, 2023). Selain itu, dalam *big data analytics*, kemampuan untuk menyimpan dan menganalisis data kompleks menjadi kunci untuk mendapatkan wawasan berharga dari data yang besar dan beragam (Qin *et al.*, 2016; Sasaki, 2022).

Berbagai contoh nyata menunjukkan pentingnya *complex data types* dalam berbagai aplikasi modern. Salah satu contohnya adalah penyimpanan data dalam format JSON, yang banyak digunakan dalam pengembangan aplikasi web dan API.

JSON memungkinkan penyimpanan data dalam struktur yang mudah dibaca dan ditransfer, serta mendukung data bersarang (*nested data*) untuk merepresentasikan hubungan antar entitas (Widya et al., 2018). Selain itu, data geospasial merupakan contoh penting lainnya, di mana informasi lokasi dan atribut terkait disimpan dalam format yang mendukung analisis spasial dan pemetaan, seperti yang digunakan dalam sistem pemetaan transportasi dan analisis wilayah.

Penggunaan *complex data types* menjadi semakin relevan dalam konteks IoT dan big data. Dalam sistem manajemen berbasis IoT, data dari berbagai sensor harus disimpan dan dianalisis untuk meningkatkan efisiensi operasional, seperti pada sistem energi pintar (Shrouf and Miragliotta, 2015). Di sisi lain, kemampuan untuk memproses dan menganalisis data kompleks memungkinkan perusahaan mengambil keputusan yang lebih strategis berdasarkan data besar dan beragam (Qin *et al.*, 2016; Sasaki, 2022).

Secara keseluruhan, kompleksitas data yang dihasilkan oleh teknologi modern menuntut pendekatan baru dalam pengelolaan dan analisis data. Dengan pemahaman mendalam tentang *complex data types* dan penerapannya dalam konteks basis data, mahasiswa informatika dapat lebih siap menghadapi tantangan teknologi masa kini. Pemanfaatan tipe data ini tidak hanya meningkatkan efisiensi operasional tetapi juga membuka jalan bagi inovasi aplikasi baru yang dapat memanfaatkan data dengan lebih efektif. Bab ini akan memberikan pemahaman menyeluruh, mulai dari konsep dasar hingga implementasi praktis, untuk mendukung kebutuhan pengelolaan data yang semakin kompleks.

8.2 Jenis-Jenis Complex Data Types

Dalam dunia basis data, pemahaman tentang jenis-jenis complex data types sangat penting untuk mengelola dan memanipulasi data yang semakin kompleks. Dalam subbab ini, kita akan membahas beberapa jenis complex data types yang umum digunakan, termasuk array, JSON dan XML, structs dan nested data, geospatial data, serta user-defined types (UDT).

8.2.1 Aray

Array adalah struktur data yang menyimpan sekumpulan elemen dengan tipe data yang sama. Elemen-elemen dalam array diakses menggunakan indeks, yang biasanya dimulai dari nol. Dalam basis data, array memungkinkan penyimpanan data yang terstruktur dan dapat diakses dengan efisien. Representasi array dalam basis data sering kali melibatkan penggunaan tipe data khusus yang mendukung penyimpanan koleksi data.

Di MySQL, array tidak didukung secara langsung, tetapi dapat diimplementasikan menggunakan tipe data JSON untuk menyimpan koleksi nilai. PostgreSQL, di sisi lain, mendukung array sebagai tipe data bawaan, memungkinkan pengguna untuk mendefinisikan kolom dengan tipe array seperti `INTEGER` atau `TEXT`. MongoDB, sebagai basis data dokumen, juga memungkinkan penyimpanan array dalam dokumen JSON, di mana array dapat berisi berbagai tipe data, termasuk objek dan nilai primitive (Yusof & Man, 2018; Rmis & Topcu, 2020).

Kelebihan penggunaan array termasuk efisiensi dalam penyimpanan dan akses data, serta kemudahan dalam pengelolaan data yang terstruktur. Namun, kekurangan array adalah keterbatasan dalam fleksibilitas, terutama ketika ukuran array perlu diubah, yang dapat menyebabkan overhead dalam pengelolaan memori (Tampubolon et al., 2021; Vodnanský, 2021)

8.2.2 Json & XML

JSON (*JavaScript Object Notation*) dan XML (*eXtensible Markup Language*) adalah dua format yang umum digunakan untuk penyimpanan dan transportasi data. JSON dikenal karena sintaksisnya yang lebih ringkas dan kemudahan dalam parsing, menjadikannya pilihan populer untuk aplikasi web dan API. XML, meskipun lebih verbose, menawarkan kemampuan untuk mendefinisikan skema dan struktur data yang lebih kompleks (Yusof & Man, 2018; Heyvaert et al., 2019).

PostgreSQL mendukung tipe data JSON dan JSONB, yang memungkinkan penyimpanan dan manipulasi data dalam

format JSON dengan efisiensi tinggi. MongoDB, sebagai basis data dokumen, secara alami menggunakan format JSON untuk menyimpan dokumen, memberikan fleksibilitas dalam struktur data dan kemudahan dalam pengambilan data (Mironov et al., 2020; Rmis & Topcu, 2020).

Dalam hal efisiensi, JSON sering kali lebih cepat dalam proses parsing dibandingkan XML, dengan estimasi bahwa JSON dapat diproses hingga seratus kali lebih cepat daripada XML dalam beberapa kasus (Yusof & Man, 2018; Heyvaert et al., 2019). Namun, XML memiliki keunggulan dalam hal validasi skema dan interoperabilitas, yang dapat menjadi faktor penting dalam aplikasi tertentu (Khan et al., 2019; Guo & Onstein, 2020).

8.2.3 Structs dan Nested Data

Data bersarang merujuk pada struktur data yang memungkinkan penyimpanan elemen di dalam elemen lain, menciptakan hierarki data yang kompleks. Structs adalah tipe data yang terdiri dari beberapa field, yang masing-masing dapat memiliki tipe data yang berbeda, dan sering digunakan untuk merepresentasikan objek dalam basis data.

Dalam Apache Hive, structs dapat digunakan untuk menyimpan data yang terstruktur dengan baik. Hive mendukung tipe data kompleks, termasuk structs dan arrays, yang memungkinkan pengguna untuk menyimpan data dalam format yang lebih terorganisir dan mudah diakses (Ding *et al.*, 2020; Zhai *et al.*, 2022). Penyimpanan data bersarang memungkinkan analisis yang lebih mendalam dan fleksibilitas dalam pengelolaan data.

8.2.4 Geospatial Data

Data geospasial mencakup informasi yang berkaitan dengan lokasi dan bentuk objek di permukaan bumi. Tipe data geospasial yang umum digunakan termasuk point (titik), polygon (poligon), dan line (garis). Data ini sangat penting dalam aplikasi seperti pemetaan, analisis lingkungan, dan perencanaan kota (Suroso and Martono, 2018; Truică *et al.*, 2021).

PostGIS adalah ekstensi untuk PostgreSQL yang menambahkan dukungan untuk data geospasial, memungkinkan pengguna untuk melakukan query dan analisis spasial dengan efisien. Oracle Spatial juga menawarkan fitur serupa, memberikan kemampuan untuk menyimpan dan memanipulasi data geospasial dalam basis data relasional (Anderson, 2017; Hu *et al.*, 2018). Kedua sistem ini memungkinkan pengguna untuk melakukan analisis yang kompleks dan menghasilkan visualisasi data yang informatif.

8.2.5 *User-Defined Types* (UDT)

User-Defined Types (UDT) adalah tipe data yang didefinisikan oleh pengguna untuk memenuhi kebutuhan spesifik aplikasi. UDT memungkinkan pengguna untuk membuat struktur data yang lebih kompleks dan sesuai dengan domain aplikasi mereka. Proses pembuatan UDT biasanya melibatkan definisi atribut dan metode yang terkait dengan tipe data tersebut (Yaghmazadeh et al., 2018; Guo et al., 2023).

Di Oracle, pengguna dapat membuat UDT menggunakan perintah SQL untuk mendefinisikan tipe data baru yang mencakup atribut dan metode. PostgreSQL juga mendukung pembuatan UDT, memungkinkan pengguna untuk mendefinisikan tipe data yang sesuai dengan kebutuhan aplikasi mereka, termasuk tipe data kompleks seperti structs dan arrays (Yusof & Man, 2017; Khan et al., 2023). Penggunaan UDT dapat memungkinkan pengembang untuk meningkatkan keterbacaan dan pemeliharaan kode, serta meningkatkan efisiensi dalam pengelolaan data.

8.3 Teknik Penyimpanan dan Indeks untuk Complex Data Types

Dalam pengelolaan basis data modern, teknik penyimpanan dan indeks yang efisien untuk complex data types menjadi sangat penting. Dengan meningkatnya volume dan kompleksitas data yang dihasilkan, terutama dalam konteks big data dan Internet of Things (IoT), pemilihan metode penyimpanan dan indeks yang tepat dapat mempengaruhi

kinerja dan skalabilitas sistem basis data. Subbab ini akan membahas berbagai teknik penyimpanan dan indeks yang digunakan untuk complex data types, termasuk penyimpanan khusus, indeks untuk array dan JSON, spatial indexing, serta dampak kompleksitas tipe data terhadap kinerja dan skalabilitas.

8.3.1 Penyimpanan Khusus

Tabel kolom lebar adalah struktur penyimpanan yang dirancang untuk menangani data dengan jumlah kolom yang sangat besar. Dalam model ini, setiap baris dapat memiliki kolom yang berbeda, memungkinkan fleksibilitas dalam penyimpanan data yang tidak terstruktur atau semi-terstruktur. Tabel kolom lebar sangat cocok untuk aplikasi yang memerlukan penyimpanan data dalam format yang bervariasi, seperti data sensor dalam IoT, di mana setiap sensor mungkin menghasilkan jenis data yang berbeda (Al-Obaidi *et al.*, 2022; Jamali *et al.*, 2023).

Penyimpanan terkompresi adalah teknik yang digunakan untuk mengurangi ukuran data yang disimpan, sehingga menghemat ruang penyimpanan dan meningkatkan kecepatan akses data. Dalam konteks complex data types, teknik kompresi seperti Lempel-Ziv (LZ) dan Huffman coding sering digunakan untuk mengompresi data JSON atau array, yang dapat mengurangi overhead penyimpanan dan mempercepat transfer data. Penyimpanan terkompresi juga dapat meningkatkan efisiensi dalam pengolahan data, terutama ketika data yang besar perlu diambil dan dianalisis secara berkala (Hajizadeh & Navimipour, 2017).

8.3.2 Indeks untuk Array dan JSON

PostgreSQL menyediakan dukungan untuk indeks GIN, yang sangat berguna untuk menyimpan dan mengindeks data dalam format array dan JSON. Indeks GIN memungkinkan pencarian yang cepat dan efisien pada kolom yang berisi array atau dokumen JSON, dengan cara mengindeks setiap elemen dalam array atau setiap kunci dalam objek JSON secara

terpisah. Hal ini memungkinkan pengguna untuk melakukan query yang kompleks dan mendapatkan hasil dengan cepat, bahkan pada dataset yang besar (Nazari et al., 2019).

JSONB adalah tipe data di PostgreSQL yang memungkinkan penyimpanan data JSON dalam format biner yang terstruktur. Dengan menggunakan JSONB, pengguna dapat memanfaatkan indeks GIN untuk meningkatkan kinerja query. JSONB indexing memungkinkan pengguna untuk melakukan pencarian berdasarkan kunci dan nilai dalam dokumen JSON, memberikan fleksibilitas dalam pengambilan data yang diperlukan tanpa harus memuat seluruh dokumen ke dalam memori ebadi (Ebadi & Navimipour, 2018; Pina et al., 2022). Hal ini sangat penting dalam aplikasi yang memerlukan akses cepat ke data yang terstruktur, seperti aplikasi web dan layanan API.

8.3.3 Spatial Indexing

Spatial indexing adalah teknik yang digunakan untuk mengelola dan mengakses data geospasial dengan efisien. R-tree adalah salah satu metode yang paling umum digunakan untuk indexing data geospasial, yang memungkinkan penyimpanan dan pencarian data dalam bentuk bounding boxes. R-tree sangat efektif untuk query yang melibatkan pencarian area atau rentang, seperti dalam aplikasi pemetaan dan analisis spasial (Eldahshan et al., 2020; Thatikonda & Rajesh, 2023). Di sisi lain, Quadtree adalah struktur data yang membagi ruang dua dimensi menjadi empat kuadran, yang memungkinkan pencarian yang cepat dan efisien untuk data yang tersebar secara geografis.

Implementasi spatial indexing, seperti R-tree dan Quadtree, dapat ditemukan dalam berbagai sistem basis data geospasial, termasuk PostGIS dan Oracle Spatial. Sistem ini memungkinkan pengguna untuk melakukan query spasial yang kompleks, seperti pencarian titik terdekat, analisis area, dan penghitungan jarak antara objek geospasial. Dengan menggunakan teknik indexing yang tepat, kinerja query dapat ditingkatkan secara signifikan, bahkan ketika bekerja dengan dataset yang sangat besar (Chen, 2021).

8.3.4 Kinerja dan Skalabilitas

Kompleksitas tipe data dapat memiliki dampak signifikan terhadap kinerja query dalam basis data. Data yang lebih kompleks, seperti nested JSON atau array besar, dapat memperlambat waktu respons query jika tidak dikelola dengan baik. Penggunaan teknik penyimpanan dan indeks yang efisien, seperti yang telah dibahas sebelumnya, dapat membantu mengurangi dampak ini dan meningkatkan kinerja keseluruhan system (Behera & Zuber, 2023). Selain itu, pemilihan algoritma query yang tepat juga dapat mempengaruhi kecepatan eksekusi query, terutama dalam konteks data yang kompleks.

Optimasi penyimpanan dan query adalah langkah penting dalam meningkatkan kinerja sistem basis data yang menangani complex data types. Teknik seperti denormalisasi, penggunaan indeks yang tepat, dan penyimpanan terkompresi dapat membantu mengurangi waktu akses dan meningkatkan efisiensi penyimpanan. Selain itu, pemantauan kinerja secara berkala dan penyesuaian terhadap strategi penyimpanan dan indexing dapat membantu menjaga kinerja sistem seiring dengan pertumbuhan volume data (Nurhadi et al., 2022). Dengan pendekatan yang tepat, sistem basis data dapat dioptimalkan untuk menangani kompleksitas data yang terus berkembang, memastikan ketersediaan dan kecepatan akses data yang diperlukan untuk aplikasi modern.

8.4 Querying dan Manipulasi Data

Dalam konteks basis data modern, kemampuan untuk melakukan querying dan manipulasi data yang kompleks sangat penting. Dengan meningkatnya penggunaan complex data types, seperti array, JSON, dan data geospasial, pemahaman yang mendalam tentang teknik querying yang efektif menjadi sangat diperlukan. Subbab ini akan membahas berbagai aspek querying dan manipulasi data, termasuk sintaks query untuk complex types, fungsi dan operasi khusus, geospatial querying, serta integrasi dan transformasi data.

8.4.1 Query Dasar untuk Complex Types

Querying untuk complex data types memerlukan sintaks yang berbeda dibandingkan dengan tipe data sederhana. Dalam PostgreSQL, misalnya, untuk mengakses elemen dalam array, pengguna dapat menggunakan sintaks `arrayindex`, di mana `index` adalah posisi elemen yang ingin diakses. Untuk JSON, PostgreSQL menyediakan operator `->` dan `->>` untuk mengakses nilai dalam objek JSON. Operator `->` mengembalikan nilai dalam format JSON, sementara `->>` mengembalikan nilai sebagai teks (Imam *et al.*, 2018). Dalam MongoDB, proses querying dilakukan dengan menggunakan metode seperti `db.collection.find()` yang memungkinkan pengguna untuk mencari dokumen berdasarkan kriteria tertentu dalam struktur JSON.

Sebagai contoh, jika kita memiliki tabel dengan kolom JSON yang menyimpan informasi pengguna, kita dapat menggunakan query berikut untuk memfilter pengguna berdasarkan usia yang tersimpan dalam objek JSON:

Gambar 8.1. Kode SQL untuk mengambil data spesifik dari database
(Sumber: Dokumentasi pribadi)

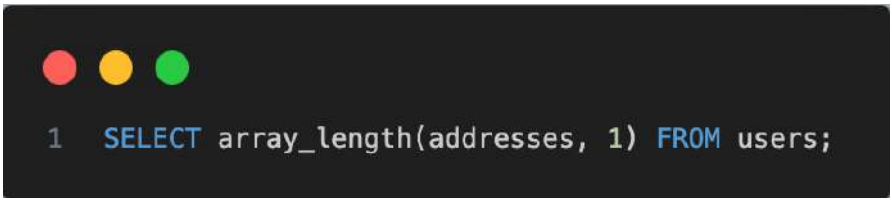
Dalam kasus array, jika kita ingin menemukan semua pengguna yang memiliki lebih dari satu alamat, kita dapat menggunakan query berikut:

Gambar 8.2. Kode SQL untuk mengambil data pada tabel users dari database
(Sumber: Dokumentasi pribadi)

Query ini menunjukkan bagaimana pengguna dapat memanfaatkan operator spesifik untuk melakukan filtering pada data yang kompleks (Lyu *et al.*, 2023).

8.4.2 Fungsi dan Operasi Khusus

PostgreSQL menyediakan berbagai fungsi untuk bekerja dengan array. Salah satu fungsi yang sering digunakan adalah `array_length`, yang mengembalikan jumlah elemen dalam array. Contoh penggunaannya adalah sebagai berikut:



Gambar 8.3. Kode SQL untuk menghitung jumlah alamat setiap pengguna dalam table
(Sumber: Dokumentasi pribadi)

Fungsi ini sangat berguna untuk analisis data yang melibatkan koleksi elemen, seperti menghitung jumlah item dalam daftar atau memverifikasi apakah array kosong (Masrom et al., 2015).

JSON Path adalah fitur yang memungkinkan pengguna untuk menavigasi dan mengambil data dari dokumen JSON dengan cara yang lebih intuitif. Di PostgreSQL, pengguna dapat menggunakan sintaks JSON Path untuk melakukan query yang lebih kompleks. Contohnya, untuk mengambil semua pengguna yang memiliki alamat di kota tertentu, kita dapat menggunakan:

Gambar 8.4. Kode SQL untuk menemukan semua pengguna yang memiliki alamat di kota New York
(Sumber: Dokumentasi pribadi)

Pada SQL Server, fungsi `JSON_VALUE` dan `JSON_QUERY` digunakan untuk mengekstrak nilai dari dokumen JSON. Contoh penggunaannya adalah sebagai berikut:

Gambar 8.5. Kode SQL untuk mengekstrak nilai dan memberikan nama alias city pada hasil ekstraksi
(Sumber: Dokumentasi pribadi)

Penggunaan operasi ini memungkinkan pengguna untuk melakukan manipulasi data JSON dengan lebih efisien dan efektif (Zarkandi, 2021).

8.4.3 Geospatial Querying

Dalam konteks data geospasial, fungsi-fungsi seperti `ST_Distance` dan `ST_Intersects` sangat penting untuk melakukan analisis spasial. `ST_Distance` digunakan untuk menghitung jarak antara dua geometri, sedangkan `ST_Intersects` digunakan untuk menentukan apakah dua geometri saling berpotongan. Contoh penggunaan fungsi ini dalam PostgreSQL dengan PostGIS adalah sebagai berikut:

Gambar 8.6. Kode SQL untuk menghitung jarak antara dua geometri dalam sebuah basis data spasial
(Sumber: Dokumentasi pribadi)

Query ini mengembalikan jarak antara dua area yang saling berpotongan, berguna untuk analisis spasial seperti memahami hubungan geografis, interaksi antar zona, atau dampak tata ruang. Data ini mendukung aplikasi seperti perencanaan perkotaan, manajemen bencana, dan konservasi lingkungan, sehingga

pengambilan keputusan dapat dilakukan lebih akurat dan efisien. (Nurhadi et al., 2022).

Visualisasi data spasial merupakan langkah penting dalam analisis geospasial. Dengan menggunakan alat seperti QGIS atau ArcGIS, pengguna dapat memvisualisasikan hasil query geospasial untuk mendapatkan wawasan yang lebih baik. PostGIS juga menyediakan fungsi untuk menghasilkan data yang dapat digunakan dalam visualisasi, seperti `ST_AsGeoJSON`, yang mengonversi geometri menjadi format GeoJSON yang dapat digunakan dalam aplikasi web (Tong *et al.*, 2023).

8.4.4 Integrasi dan Transformasi Data

Salah satu tantangan dalam bekerja dengan data JSON adalah mengonversi struktur nested menjadi format tabel tradisional. Di PostgreSQL, pengguna dapat menggunakan fungsi `jsonb_to_recordset` untuk melakukan flattening pada data JSON. Contoh penggunaannya adalah sebagai berikut:

```
SELECT * FROM jsonb_to_recordset(user_data) AS x(name TEXT, age INT, address JSONB);
```

Gambar 8.7. Kode SQL untuk mengubah data format JSON ke tabel relasional
(Sumber: Dokumentasi pribadi)

Fungsi ini memungkinkan pengguna untuk mengonversi data JSON yang kompleks menjadi format tabel yang lebih mudah dikelola dan dianalisis (Tolosana *et al.*, 2020).

Ekspor data dari basis data ke format lain, seperti CSV atau XML, juga merupakan bagian penting dari manipulasi data. PostgreSQL menyediakan perintah `COPY` untuk mengekspor data ke format CSV. Contoh perintahnya adalah:

Gambar 8.8. Kode SQL untuk **ekspor data tabel dalam database** ke file CSV (*Comma Separated Values*)
(Sumber: Dokumentasi pribadi)

Dengan cara ini, pengguna dapat dengan mudah memindahkan data dari basis data ke aplikasi lain untuk analisis lebih lanjut atau untuk berbagi dengan pihak ketiga (Asghari & Zarrabi, 2016).

8.5 Studi Kasus Implementasi

Dalam subbab ini, kita akan membahas beberapa studi kasus implementasi yang menunjukkan penerapan complex data types dalam konteks nyata. Kasus-kasus ini mencakup penggunaan data JSON dalam sistem inventaris, analisis data geospasial pada pemetaan transportasi, dan pengelolaan nested data untuk aplikasi streaming. Setiap studi kasus akan menjelaskan desain skema, query yang digunakan, serta teknik optimasi yang diterapkan.

8.5.1 Data JSON dalam Sistem Inventaris

Dalam sistem inventaris, penggunaan data JSON memungkinkan fleksibilitas dalam menyimpan informasi barang yang beragam. Sebagai contoh, skema tabel untuk menyimpan data barang dapat dirancang sebagai berikut:

Gambar 8.9. Kode SQL untuk membuat tabel baru dengan nama inventory
(Sumber: Dokumentasi pribadi)

Di dalam kolom `item_data`, informasi tentang barang dapat disimpan pada format JSON. Penggunaan kolom `item_data` dapat menyimpan informasi barang dalam format JSON, mencakup atribut seperti nama barang, harga, deskripsi, dan properti lain yang dapat bervariasi untuk setiap entri. Pendekatan ini memberikan fleksibilitas dalam mengelola data inventaris tanpa perlu mendefinisikan skema kolom yang rigid. Contoh data JSON yang disimpan dalam kolom ini adalah:

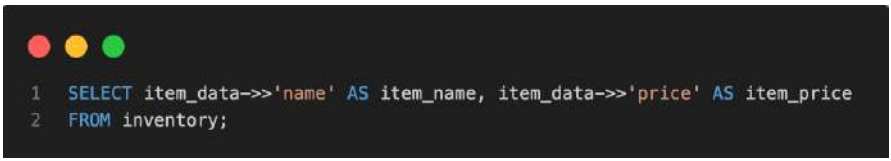
```json
{
    "_comment": "json",
    "name": "Laptop",
    "category": "Electronics",
    "price": 1500,
    "quantity": 10,
    "specifications": {
        "processor": "Intel i7",
        "ram": "16GB",
        "storage": "512GB SSD"
    }
}
```

Gambar 8.10. Kode JSON data tentang sebuah produk
(Sumber: Dokumentasi pribadi)

Desain ini memungkinkan penyimpanan informasi yang kompleks dan terstruktur tanpa memerlukan perubahan skema yang signifikan saat atribut baru ditambahkan (Irzan & Sutriyono, 2021; Suwarno & Nadhia, 2023).

Untuk mengambil data dari struktur JSON yang bersarang, PostgreSQL menyediakan operator dan fungsi yang efisien.

Sebagai contoh, jika kita ingin mengambil nama dan harga barang dari data JSON, kita dapat menggunakan query berikut:



```
SELECT item_data->>'name' AS item_name, item_data->>'price' AS item_price
FROM inventory;
```

Gambar 8.11. Kode SQL untuk ekstrak data spesifik dari kolom JSON di tabel.
(Sumber: Dokumentasi pribadi)

Jika kita ingin mengakses spesifikasi dari barang, kita dapat menggunakan query berikut:

```
SELECT item_data->'specifications'->>'processor' AS processor
FROM inventory;
```

Gambar 8.12. Kode SQL untuk mengambil data spesifik dari JSON di kolom tabel.
(Sumber: Dokumentasi pribadi)

Query ini menunjukkan bagaimana pengguna dapat dengan mudah mengakses data bersarang dalam format JSON, memberikan fleksibilitas dalam pengambilan informasi yang diperlukan (Suwarno & Nadhia, 2023).

8.5.2 Analisis Data Geospasial pada Pemetaan Transportasi

Dalam konteks pemetaan transportasi, analisis data geospasial sangat penting untuk memahami hubungan antar lokasi. PostGIS, sebagai ekstensi untuk PostgreSQL, menyediakan fungsi geospasial yang kuat untuk melakukan analisis ini. Sebagai contoh, kita dapat membuat tabel untuk menyimpan data lokasi sebagai berikut:

```sql
CREATE TABLE locations (id SERIAL PRIMARY KEY,
name VARCHAR(100), geom GEOGRAPHY(Point, 4326));
```

Gambar 8.13. Buat tabel baru dalam database dengan nama locations.
(Sumber: Dokumentasi pribadi)

Setelah tabel dibuat, kita dapat menyimpan data lokasi dalam format geografi. Untuk menghitung jarak antar lokasi, kita dapat menggunakan fungsi `ST_Distance`. Contoh query untuk menghitung jarak antara dua lokasi adalah:

```sql
SELECT a.name AS location_a, b.name AS location_b,
ST_Distance(a.geom, b.geom) AS distance
FROM locations a, locations b
WHERE a.id <> b.id;
```

Gambar 8.14. Kode SQL untuk menghitung jarak antara setiap pasangan lokasi pada tabel locations
(Sumber: Dokumentasi pribadi)

Query ini akan mengembalikan jarak antara semua kombinasi lokasi yang ada dalam tabel, memberikan informasi yang berguna untuk analisis transportasi. Selain itu, query ini sangat berguna dalam berbagai analisis geospasial, seperti perencanaan transportasi, logistik, dan analisis jaringan. Sebagai contoh, jarak antar lokasi yang dihasilkan dapat digunakan untuk mengembangkan model rute terpendek atau mengidentifikasi lokasi strategis untuk pusat distribusi. Dengan demikian, kode ini menjadi alat penting dalam mendukung pengambilan keputusan berbasis spasial, sebagaimana dijelaskan dalam literatur yang relevan (Sutarto et al., 2017; Damayanti & Rassarandi, 2023).

8.5.3 Pengelolaan Nested Data untuk Aplikasi Streaming

Dalam aplikasi streaming, pengelolaan metadata video yang kompleks sangat penting untuk memberikan pengalaman pengguna yang baik. Metadata ini dapat mencakup informasi seperti judul, deskripsi, durasi, dan kategori. Dalam konteks ini, kita dapat menggunakan nested struct untuk menyimpan metadata. Sebagai contoh, skema tabel dapat dirancang sebagai berikut:

```
CREATE TABLE videos (id SERIAL PRIMARY KEY,
title VARCHAR(255), metadata JSONB NOT NULL);
```

Gambar 8.15. Kode SQL untuk **membuat sebuah tabel baru** di dalam database dengan nama **"videos"**
(Sumber: Dokumentasi pribadi)

Di dalam kolom `metadata`, informasi tambahan tentang video dapat disimpan dalam format JSON, termasuk atribut seperti genre, sutradara, dan tahun rilis. Contoh data JSON yang disimpan dalam kolom ini adalah:

```
{
    "_comment": "json",
    "genre": "Action",
    "director": "John Doe",
    "release_year": 2021,
    "cast": [
        {"actor": "Jane Smith", "role": "Lead"},
        {"actor": "Bob Johnson", "role": "Supporting"}
    ]
}
```

Gambar 8.16. Kode JSON memuat data tentang sebuah video
(Sumber: Dokumentasi pribadi)

Desain ini memungkinkan penyimpanan informasi yang terstruktur dan fleksibel, memudahkan pengelolaan metadata video (Irzan & Sutriyono, 2021).

Untuk mengoptimalkan pengambilan data dari struktur nested, kita dapat menggunakan query yang efisien. Misalnya, jika kita ingin mengambil semua video dengan genre tertentu, kita dapat menggunakan query berikut:

```
1  SELECT title, metadata->>'director' AS director
2  FROM videos
3  WHERE metadata->>'genre' = 'Action';
```

Gambar 8.17. Kode SQL untuk cari video dengan genre 'Action' dan tampilkan judul serta sutradaranya.
(Sumber: Dokumentasi pribadi)

Query ini memungkinkan pengguna untuk dengan cepat mengambil informasi yang relevan dari data yang bersarang, meningkatkan efisiensi dalam pengelolaan data (Suwarno & Nadhia, 2023).

DAFTAR PUSTAKA

Al-Obaidi, K.M. *et al.* (2022) 'A Review of Using IoT for Energy Efficient Buildings and Cities: A Built Environment Perspective', *Energies*. MDPI. Available at: https://doi.org/10.3390/en15165991.

Anderson, C.B. (2017) 'Data-first Manifesto: Shifting Priorities In Scholarly Communications', in *Information Services and Use*. IOS Press, pp. 335–342. Available at: https://doi.org/10.3233/ISU-170852.

Asghari, P. and Zarrabi, H. (2016) 'A Survey on NoSQL and its Terminology', *IJARCCE*, 5(12), pp. 405–410. Available at: https://doi.org/10.17148/ijarcce.2016.51293.

Behera, A. and Zuber Md (2023) 'Innovations in Frequent Itemset Mining: Challenges And Opportunities', *ACCENTS Transactions on Information Security*, 8(29). Available at: https://doi.org/10.19101/tis.2023.829001.

Suwarno and Nadhia, A.P. (2023) 'Rancang Bangun Sistem Informasi Manajemen Inventaris Berbasis Website Menggunakan Metode SDLC', *CBIS JOURNAL*, 11(02). Available at: http://ejournal.upbatam.ac.id/index.php/cbishttp://ejournal.upbatam.ac.id/index.php/cbis.

Chen, G. (2021) 'The larger the better: Analysis of a Scalable Spectral Clustering Algorithm With Cosine Similarity', in *Frontiers in Artificial Intelligence and Applications*. IOS Press BV, pp. 488–495. Available at: https://doi.org/10.3233/FAIA210280.

Damayanti, T. and Rassarandi, F.D. (2023) 'Pemetaan Perubahan Kondisi Jalan Kota di Kecamatan Sekupang pada Tahun 2017 dan Tahun 2021', *Jurnal Nasional Teknologi dan Sistem Informasi*, 9(1), pp. 1–11. Available at: https://doi.org/10.25077/teknosi.v9i1.2023.1-11.

Ding, L. *et al.* (2020) 'A Framework Uniting Ontology-Based Geodata Integration and Geovisual Analytics', *ISPRS International Journal of Geo-Information*, 9(8). Available at: https://doi.org/10.3390/ijgi9080474.

Ebadi, Y. and Navimipour, N.J. (2018) 'An Energy-aware Method For Data Replication In The Cloud Environments Using A Tabu

Search And Particle Swarm Optimization Algorithm', *Concurrency and Computation: Practice and Experience*, 31(1). Available at: https://doi.org/10.1002/cpe.4757.

ElDahshan, K.A., Alhabshy, A.A. and Abutaleb, G.E. (2020) *A Comparative Study Among the Main Categories of NoSQL Databases*, *Bulletin of Science*. Available at: https://doi.org/10.21608/absb.2020.210374.

Guo, D. and Onstein, E. (2020) 'State-of-the-Art Geospatial Information Processing in NoSQL Databases', *ISPRS International Journal of Geo-Information*. MDPI AG. Available at: https://doi.org/10.3390/ijgi9050331.

Guo, Q. *et al.* (2023) 'Multi-model Query Languages: Taming the Variety of Big Data', *Distributed and Parallel Databases*, 42(1), pp. 31–71. Available at: https://doi.org/10.1007/s10619-023-07433-1.

Hajizadeh, R. and Navimipour, N.J. (2017) 'A Method for Trust Evaluation in The Cloud Environments Using A Behavior Graph and Services Grouping', *Kybernetes*, 46(7), pp. 1245–1261. Available at: https://doi.org/10.1108/K-02-2017-0070.

Heyvaert, P. *et al.* (2019) 'Conformance Test Cases for the RDF Mapping Language (RML)', in *Communications in Computer and Information Science*. Springer Verlag, pp. 162–173. Available at: https://doi.org/10.1007/978-3-030-21395-4_12.

Hu, F. *et al.* (2018) 'Evaluating the Open Source Data Containers For Handling Big Geospatial Raster Data', *ISPRS International Journal of Geo-Information*, 7(4). Available at: https://doi.org/10.3390/ijgi7040144.

Hu, L. and Shu, Y. (2023) *Enhancing Decision-Making with Data Science in The Internet of Things Environments*, *IJACSA) International Journal of Advanced Computer Science and Applications*. Available at: https://doi.org/10.14569/ijacsa.2023.01409120.

Imam, A.A. *et al.* (2018) 'Automatic Schema Suggestion Model for NoSQL Document-Stores Databases', *Journal of Big Data*,

5(1). Available at: https://doi.org/10.1186/s40537-018-0156-1.

Irzan, M.I. and Depa, D.S. (2021) 'Rancang Bangun Sistem Informasi Inventaris Barang Dinas Komunikasi dan Informatika Indragiri Hulu', *Indonesian Journal of Informatic Research and Software Engineering (IJIRSE)*, 1(1), pp. 53–59. Available at: https://doi.org/10.57152/ijirse.v1i1.49.

Jamali, M.-R. *et al.* (2023) 'An Empirical Evaluation of Data Integrity Algorithm Performance in Non-Relational Document Databases', *VAWKUM Transactions on Computer Sciences*, 11(2), pp. 70–82. Available at: https://doi.org/10.21015/vtcs.v11i2.1663.

Khan, W. *et al.* (2023) 'SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review', *Big Data and Cognitive Computing*. MDPI. Available at: https://doi.org/10.3390/bdcc7020097.

Khan, Y. *et al.* (2019) 'One Size Does Not Fit All: Querying Web Polystores'. Available at: https://doi.org/10.1109/access.2018.2888601.

Lyu, E. *et al.* (2023) 'Motion Planning of Manipulator by Points-Guided Sampling Network', *IEEE Transactions on Automation Science and Engineering*, 20(2), pp. 821–831. Available at: https://doi.org/10.1109/TASE.2022.3168542.

Masrom, U.K., Alwi, N.A.N.M. and Mat Daud, N.S. (2015) 'The Role of Task Complexity and Task Motivation in Language Production', *GEMA Online ® Journal of Language Studies*, 15(2). Available at: https://doi.org/10.17576/gema-2015-1502-03.

Mironov, V. *et al.* (2020) *JSON Documents Processing Using Situation-Oriented Databases*, *Acta Polytechnica Hungarica*. Available at: https://doi.org/10.12700/aph.17.8.2020.8.3.

Nazari, E., Shahriari, M.H. and Tabesh, H. (2019) 'Big Data Analysis in Healthcare: Apache Hadoop, Apache Spark and Apache Flink', *Frontiers in Health Informatics*, 8. Available at: https://doi.org/10.30699/fhi.v8i1.180.

Nurhadi, Kadir, R.A. and Surin, E.S.M. (2022) 'Complex SQL-NoSQL Query Translation for Data Lake Management', *Journal of Computer Science*, 18(12), pp. 1179–1188. Available at: https://doi.org/10.3844/jcssp.2022.1179.1188.

Pina, E., Sá, F. and Bernardino, J. (2022) 'NewSQL Databases Assessment: CockroachDB, MariaDB Xpand, and VoltDB', *Future Internet*, 15(1). Available at: https://doi.org/10.3390/fi15010010.

Qin, Y. *et al.* (2016) 'When things matter: A survey on data-centric internet of things', *Journal of Network and Computer Applications*, 64, pp. 137–153. Available at: https://doi.org/10.1016/j.jnca.2015.12.016.

Rmis, A. and Topcu, A. (2020) 'Evaluating Riak Key Value Cluster for Big Data', *Tehnicki vjesnik - Technical Gazette*, 27. Available at: https://doi.org/10.17559/TV-20180916120558.

Shrouf, F. and Miragliotta, G. (2015) 'Energy management based on Internet of Things: Practices and framework for adoption in production management', *Journal of Cleaner Production*, 100, pp. 235–246. Available at: https://doi.org/10.1016/j.jclepro.2015.03.055.

Suroso, F. and Martono, G.H. (2018) *A Brief Study of Comparison between Three Document Databases*, *International Journal of Computer Applications*. Available at: https://doi.org/10.5120/ijca2018918251.

Sutarto, Novianto, A. and Prasetyo, A. (2017) *Pembuatan Basis Data Sistem Informasi Geografis (SIG) Untuk Mendukung Perencanaan Survei dan Pemetaan*. Available at: https://doi.org/10.37875/hidropilar.v3i2.58.

Tampubolon, W., Reinhardt, W. and Tampomas Tobing, S.L. (2021) *NoSQL Standard and Approach for Geospatial Database Collection (Standar dan Pendekatan NoSQL dalam Penyusunan Basis Data Geospasial)*. Available at: https://doi.org/10.24895/sng.2020.0-0.1147.

Thatikonda, V. and Rajesh, H. (2023) 'Building Data-Intensive Applications: Scalability, Performance and Availability', *The Review of Contemporary Scientific and Academic Studies*,

3(10). Available at: https://doi.org/10.55454/rcsas.3.10.2023.004.
Tolosana, R. *et al.* (2020) 'Deepfakes and beyond: A Survey of face manipulation and fake detection', *Information Fusion*, 64, pp. 131–148. Available at: https://doi.org/10.1016/j.inffus.2020.06.014.
Tong, D. *et al.* (2023) 'Sim2Real Neural Controllers for Physics-based Robotic Deployment of Deformable Linear Objects Journal Title XX(X):1-19'. Available at: https://doi.org/10.1177/ToBeAssigned.
Truică, C.O. *et al.* (2021) 'The Forgotten Document-Oriented Database Management Systems: An Overview and Benchmark of Native XML DODBMSes in Comparison with JSON DODBMSes', *Big Data Research*, 25. Available at: https://doi.org/10.1016/j.bdr.2021.100205.
Vodnanský, D. (2021) 'Three metric-based method for data compatibility calculation', *Acta Informatica Pragensia*, 10(1), pp. 38–60. Available at: https://doi.org/10.18267/J.AIP.145.
Widya, P.W., Yustiawan, Y. and Kwon, J. (2018) 'A oneM2M-based query engine for internet of things (IoT) data streams', *Sensors (Switzerland)*, 18(10). Available at: https://doi.org/10.3390/s18103253.
Yaghmazadeh, N., Wang, X. and Dillig, I. (2018) 'Automated Migration of Hierarchical Data to Relational Tables using Programming-by-Example'. Available at: https://doi.org/10.1145/3177732.3177735.
Yusof, M.K. and Man, M. (2017) 'Efficiency of JSON for Data Retrieval in Big Data', *Indonesian Journal of Electrical Engineering and Computer Science*, 7(1), pp. 250–262. Available at: https://doi.org/10.11591/ijeecs.v7.i1.pp250-262.
Yusof, M.K. and Man, M. (2018) 'Efficiency of Flat File Database Approach In Data Storage And Data Extraction For Big Data', *Indonesian Journal of Electrical Engineering and Computer Science*, 9(2), pp. 460–473. Available at: https://doi.org/10.11591/ijeecs.v9.i2.pp460-473.
Sasaki, Y. (2022) 'A Survey on IoT Big Data Analytic Systems: Current and Future', *IEEE Internet Of Things Journal*, 9(2), pp.

1024–1036. Available at: https://doi.org/10.1109/jiot.2021.3131724.
Zarkandi, S. (2021) 'Task-based torque minimization of a 3-Pá R spherical parallel manipulator', *Robotica*, 40(3), pp. 475–504. Available at: https://doi.org/10.1017/S026357472100062X.
Zhai, Z.K. *et al.* (2022) 'Dynamic Updating Method of Geospatial Database with Incremental Data', in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. International Society for Photogrammetry and Remote Sensing, pp. 735–741. Available at: https://doi.org/10.5194/isprs-archives-XLIII-B3-2022-735-2022.

BAB 9
APPLICATION DEVELOPMENT

Oleh Hilman Nuril Hadi

9.1 Pendahuluan

Teknologi saat ini sudah berkembang dengan pesat. Hampir setiap tahun selalu ada teknologi-teknologi baru yang ditemukan dan diterapkan pada aplikasi. Pastinya sering mendengar kata 'aplikasi' di di dunia nyata maupun maya. Istilah 'aplikasi' dapat didefinisikan sebagai program komputer yang digunakan untuk melakukan tugas-tugas tertentu, seperti mengolah data, membuka file, menjalankan perintah, dan sebagainya. Tentunya untuk membuat suatu aplikasi yang baik dan dapat dimanfaatkan dengan optimal oleh pengguna, diperlukan suatu proses pengembangan yang sistematis dan terencana.

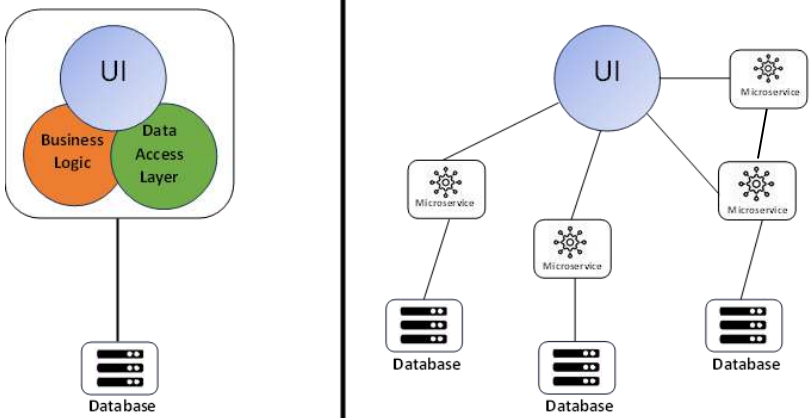
Pengembangan aplikasi (istilah lain *application development*) secara definisi mengacu pada proses untuk menciptakan perangkat lunak yang dirancang untuk memenuhi kebutuhan pengguna atau tujuan spesifik tertentu. Proses pengembangan aplikasi bisa mencakup beberapa tahapan, mulai dari perencanaan, analisis kebutuhan, pemodelan, implementasi, hingga pengujian dan pemeliharaan. Selanjutnya aplikasi itu sendiri dapat berbentuk perangkat lunak flatform desktop, website, atau perangkat *mobile*. Aplikasi tersebut juga biasa sering kali melibatkan integrasi dengan sistem lain, termasuk penggunaan basis data / database yang digunakan untuk menyimpan dan mengelola data yang diperlukan untuk fungsi aplikasi tersebut. Dengan kemajuan teknologi dan meningkatnya permintaan akan solusi digital, pengembangan aplikasi telah menjadi disiplin yang sangat penting dalam industri teknologi informasi saat ini. Penggunaan database yang efisien dan efektif menjadi fokus penting dalam memastikan aplikasi dapat beroperasi dengan baik, menyajikan data secara akurat, dan memberikan kepuasan pengalaman pengguna.

Database umumnya berperan sangat penting dalam pengembangan aplikasi. Penggunaan database secara spesifik berfungsi sebagai pusat penyimpanan dan pengelolaan data yang dibutuhkan untuk operasional aplikasi. Dalam era informasi saat ini, aplikasi sering kali diharapkan untuk menangani data-data dengan volume yang sangat besar, selanjutnya diharapkan dapat diakses dengan cepat dan akurat kepada pengguna. Jika database tidak terstruktur dan terkelola dengan baik, aplikasi tidak akan mampu menyimpan, mengambil, dan memanipulasi data secara efisien. Selain itu, database juga harus mendukung integritas dan konsistensi data melalui berbagai mekanisme seperti transaksi data dan keamanan data yang sangat penting dalam aplikasi yang melibatkan banyak pengguna. Dengan kata lain, desain dan implementasi database yang baik juga menjadi kunci keandalan dan kinerja aplikasi.

9.2 Arsitektur Aplikasi

Bagian ini menjelaskan beberapa model arsitektur yang menggambarkan konsep atau kerangka bagaimana komponen-komponen aplikasi berinteraksi dan berfungsi. Secara teori model arsitektur aplikasi saat ini bisa dibagi menjadi beberapa model diantaranya ada arsitektur monolitic, arsitektur microservices, dan arsitektur serverless. Beberapa arsitektur tersebut memiliki perbedaan dalam penerapannya yang menjadi karakteristik dari setiap modelnya. Pertama adalah **Arsitektur monolithic.** Artitektur tersebut secara teori bisa disebut sebagai pendekatan tradisional dimana semua bagian aplikasi dibangun sebagai satu kesatuan (antarmuka, logika bisnis, dan akses data). Penerapan arsitektur ini dirasakan cukup mudah untuk dikembangkan dan di-*deploy* bagi sebagian kalangan, tetapi model ini dapat menjadi tantangan rintangan saat aplikasi dikelola dan di-*scalable* jika terdapat pengembangan aplikasi lanjutan. Sebagai alternatif, terdapat **arsitektur microservices**. Arsitektur microservices secara konsep membagi aplikasi menjadi serangkaian layanan kecil yang independen dan setiap layanan akan menangani fungsi tertentu. Berbeda dengan arsitektur monolithic, pendekatan ini memiliki manfaat yang memungkinkan tim pengembangan untuk bekerja

secara paralel, meningkatkan fleksibilitas dan kecepatan dalam pengembangan serta memudahkan skalabilitas (Dahri et al., 2022). Akan tetapi disisi lainnya pendekatan ini juga memerlukan pengelolaan komunikasi antar layanan yang lebih kompleks. Gambaran umum mengenai model arsitektur monolithic dan arsitektur microservices direpresentasikan pada gambar 9.1.

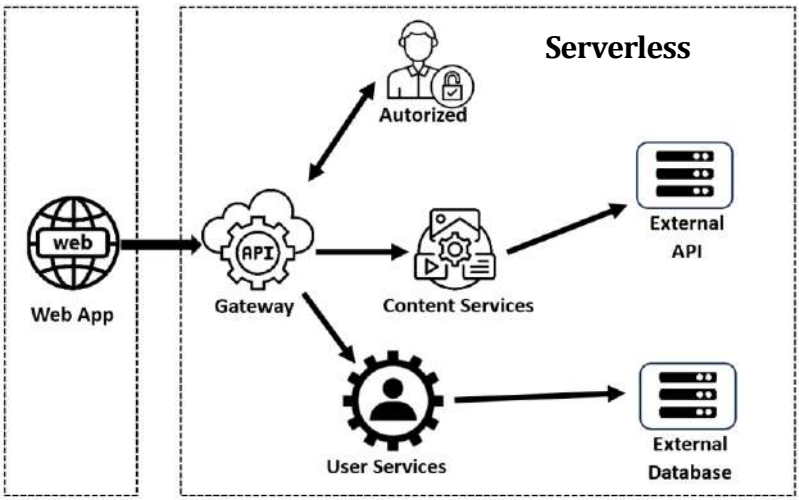


A. Monolithic

B. Microservices

Gambar 9.1. Arsitekur Monolithic (a) dan Microservices (b)

Di sisi lain, terdapat arsitektur yang memungkinkan pengembang untuk membangun dan menjalankan aplikasi tanpa harus mengelola server secara langsung, disebut sebagai **arsitektur serverless**. Arsitektur serverless atau bisa disebut dengan arsitektur tanpa server memanfaatkan penyedia layanan *cloud* menangani semua aspek layanan tanpa perlu mengelola infrastruktur secara fisik (Mulyani and Oktiawati, 2022). Selain itu, arsitektur tersebut juga memungkinkan pengembang fokus pada kode aplikasi dan logika bisnis. Meskipun menawarkan kemudahan dalam pengelolaan dan skalabilitas, arsitektur serverless juga memiliki tantangan dan rintangan seperti dalam pengelolaan status dan latensi yang sangat mungkin terjadi pada fungsi-fungsi yang dijalankan secara *on-demand*. Gambaran umum mengenai model arsitektur monolithic dan arsitektur microservices direpresentasikan pada gambar 9.2.

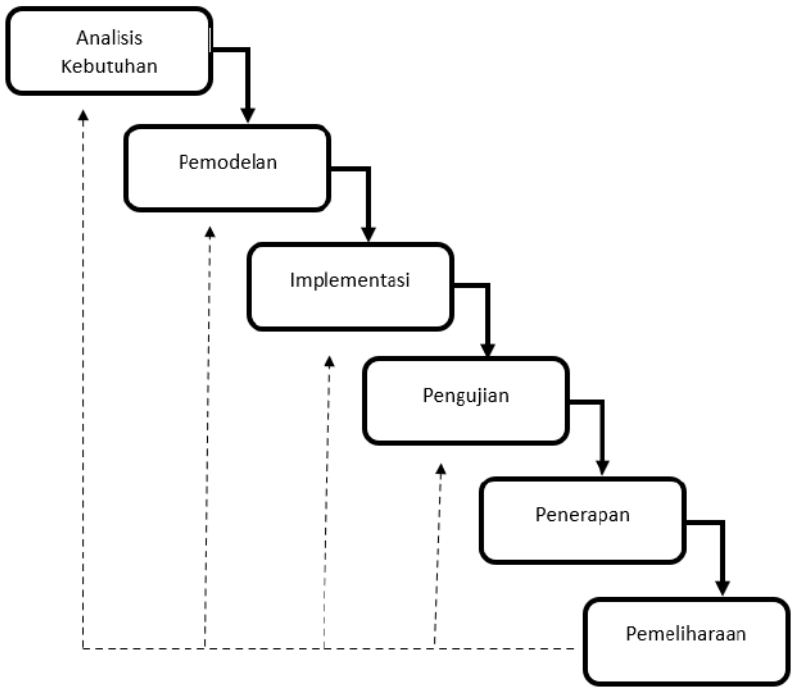


Gambar 9.2. Arsitektur Serverless

Peran database dalam pengembangan aplikasi memang sangat krusial, hal tersebut dikarenakan database berfungsi sebagai sistem penyimpanan terpusat yang mengelola semua data yang diperlukan untuk operasional aplikasi. Database juga memungkinkan aplikasi untuk menyimpan, mengakses, dan memanipulasi data dengan cara yang efisien dan terstruktur, sehingga pengguna dapat melakukan berbagai interaksi dengan aplikasi secara *real-time*. Selain itu, database juga harus mendukung data integritas dan konsistensi data melalui penerapan aturan-aturan seperti validasi & relasi antar tabel dan yang terpenting juga menjaga kualitas data. Dalam konteks aplikasi yang kompleks, database juga menyediakan kemampuan untuk mengelola transaksi, memastikan bahwa perubahan data dilakukan dengan aman dan akurat oleh akun yang telah memiliki otorisasi. Dengan adanya database yang dirancang dengan baik, harapannya aplikasi dapat memberikan pengalaman pengguna yang lebih baik dan juga mampu mendukung analisis data dan pengambilan keputusan yang berbasis informasi.

9.3 Proses Pengembangan Aplikasi

Bagian ini menjelaskan proses pengembangan aplikasi yang mengacu pada siklus hidup pengembangan perangkat lunak / *Software Development Life Cycle* (SDLC). Gambaran umum mengenai beberapa tahapan proses pengembangan direpresentasikan pada Gambar 9.3 dengan mengacu salah satu model SDLC yaitu *waterfall* model.



Gambar 9.3. SDLC Model : Air Terjun (Waterfall)

Proses pengembangan aplikasi terdiri dari beberapa tahapan penting yang saling berkaitan, dimulai dengan **analisis kebutuhan**, di mana pengembang berinteraksi dengan pemangku kepentingan untuk memahami definisi, tujuan, proses bisnis aplikasi dan fitur-fitur yang dibutuhkan. Analisis kebutuhan secara umum merupakan proses yang sangat krusial karena bertujuan untuk mengidentifikasi, mendokumentasikan, dan memvalidasi kebutuhan pengguna dan stakeholder terhadap aplikasi yang akan dikembangkan (Sommerville, 2010). Tujuan utama dari analisis

kebutuhan ini nantinya untuk memastikan bahwa semua kebutuhan yang relevan telah diidentifikasi secara lengkap dan akurat, sehingga dapat menjadi dasar yang kuat untuk desain dan pengembangan aplikasi selanjutnya. Dengan melakukan proses analisis kebutuhan yang baik dapat peningkatan kepuasan pengguna, mencegah pembengkakan biaya pengembangan, memastikan waktu pengembangan tepat waktu, serta pengurangan risiko terjadinya kesalahan atau perubahan signifikan selama proses pengembangan.

Setelah kebutuhan teridentifikasi, tahap selanjutnya adalah tahapan **pemodelan.** Tahapan ini dilakukan untuk merancang arsitektur aplikasi, antarmuka pengguna, dan struktur database yang akan digunakan. Pemodelan perangkat lunak juga menjadi langkah penting dalam siklus hidup pengembangan perangkat lunak yang berfokus pada penciptaan representasi abstrak dari aplikasi yang akan dibangun serta perencanaan detail bagaimana aplikasi tersebut akan diimplementasikan. Secara definisi pemodelan perangkat lunak mencakup pembuatan model konseptual yang menggambarkan struktur, perilaku, dan interaksi berbagai komponen. Di sisi lain, merujuk pada proses mentransformasikan model ini menjadi spesifikasi teknis yang siap untuk diimplementasikan. Manfaat dari pemodelan perangkat lunak juga peningkatan kualitas perangkat lunak, pengurangan risiko kesalahan, dan peningkatan komunikasi antar tim pengembang. Dengan memiliki model yang jelas, pengembang dapat lebih mudah mengidentifikasi dan menyelesaikan masalah potensial sebelum implementasi dimulai, yang pada akhirnya juga menghemat waktu dan biaya. Tahap pemodelan juga berfokus pada pembuatan diagram seperti diagram aliran data (DFD), diagram entitas-relasi (ERD), diagram UML, algoritma, struktur data, dan antarmuka pengguna. Selain itu, tahap pemodelan juga melibatkan pemilihan pola desain dan arsitektur yang sesuai. Dengan beberapa hasil pemodelan tersebut dapat memastikan bahwa aplikasi yang dikembangkan tidak hanya berfungsi dengan baik, tetapi juga mudah dipelihara dan dikembangkan di masa depan.

Tahap selanjutnya adalah **implementasi**, tahapan ini pengembang menulis kode dan membangun aplikasi sesuai dengan

desain yang telah dibuat. Implementasi ini berfokus untuk mentransformasikan desain yang telah dihasilkan pada tahapan sebelumnya menjadi kode program yang dapat dieksekusi oleh perangkat atau komputer. Selain itu, pengembang juga mengimplementasikan database untuk sebagai pusat penyimpanan dan pengelolaan data yang dibutuhkan untuk operasional aplikasi. Tujuan utama dari implementasi adalah untuk merealisasikan semua fitur dan fungsi yang dirancang dengan cara yang efisien dan efektif, memastikan bahwa perangkat lunak beroperasi sesuai dengan kebutuhan pengguna yang telah diidentifikasi sebelumnya. Pada tahap implementasi, pengembang sering kali menggunakan bahasa pemrograman seperti Java, Python, atau JavaScript, serta framework pengembangan seperti React, Angular, atau Django yang memudahkan pembuatan antarmuka pengguna dan logika aplikasi. Selain itu, sistem manajemen database (DBMS) seperti MySQL, PostgreSQL, atau MongoDB digunakan untuk menyimpan dan mengelola data, dengan alat ORM (Object-Relational Mapping) yang membantu dalam interaksi antara aplikasi dan database. Dengan memanfaatkan berbagai alat dan teknologi ini, pengembang dapat meningkatkan efisiensi dan kualitas aplikasi yang dihasilkan.

Setelah aplikasi selesai diimplementasi/dibangun. Tahap selanjutnya adalah tahapan **pengujian.** Tahapan ini memastikan bahwa aplikasi berfungsi dengan baik, bebas dari kesalahan-kesalahan minor ataupun mayor, dan memastikan bahwa aplikasi yang telah dikembangan telah memenuhi semua kebutuhan yang telah ditetapkan di tahap awal. Secara umum pengujian perangkat lunak didefinisikan sebagai proses sistematis untuk mengevaluasi dan memverifikasi bahwa aplikasi perangkat lunak yang telah dihasilkan di tahapan sebelumnya (Hadi et al., 2022). Hal tersebut bertujuan agar aplikasi sesuai dengan spesifikasi yang ditentukan dan memenuhi kebutuhan pengguna. Proses pengujian tersebut mencakup berbagai aktivitas yang bertujuan untuk mendeteksi cacat (*bugs*) dalam perangkat lunak serta memastikan bahwa sistem yang dikembangkan bebas dari kesalahan yang dapat mengganggu fungsionalitas aplikasi (Mili and Tchier, 2015).

Tahapan berikutnya adalah tahapan **penerapan** aplikasi. Tahapan ini akan menerapkan atau memindahkan aplikasi ke

lingkungan produksi / server agar pengguna dapat mengakses dan menggunakan aplikasinya. Penerapan (istilah lain : *deployment*) perangkat lunak adalah proses penyebaran dan instalasi aplikasi ke lingkungan produksi sehingga dapat digunakan oleh pengguna akhir. Tujuan dari penerapan aplikasi ini untuk memastikan bahwa perangkat lunak yang telah selesai dikembangkan dan diuji dapat dioperasikan dengan baik dalam lingkungan nyata dan siap digunakan untuk memenuhi kebutuhan pengguna.

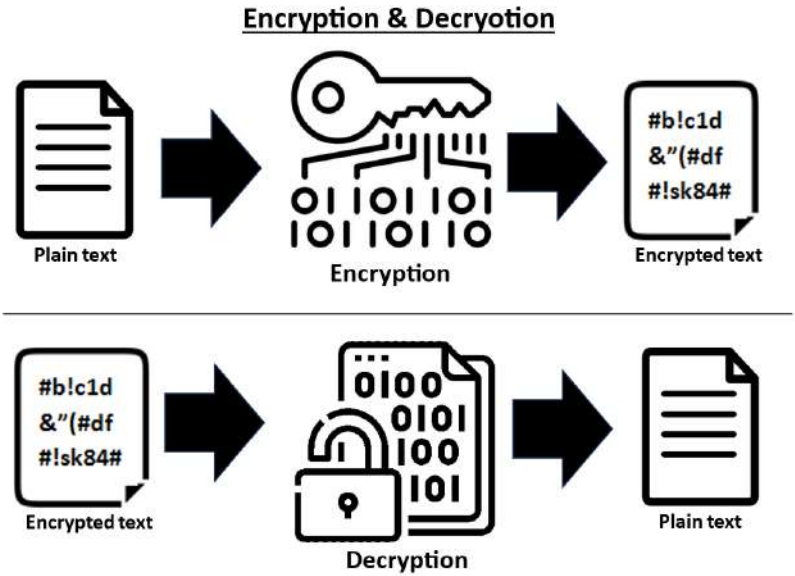
Tahap Terakhir adalah **pemeliharaan,** dimana tahapan ini diperlukan untuk memperbaiki masalah yang mungkin muncul setelah penerapan, melakukan pembaruan, dan penambahan fitur baru berdasarkan respon dari pengguna. Dengan mengikuti tahapan-tahapan ini secara sistematis dan memperhatikan keterlibatan database, pengembang dapat menghasilkan aplikasi yang berkualitas tinggi dan sesuai dengan harapan pengguna, sambil memastikan pengelolaan data yang efektif dan aman.

9.4 Keamanan dalam Pengembangan Aplikasi

Secara umum, keamanan data, secara spesifik data pada database merupakan aspek yang sangat penting dalam pengembangan aplikasi. Database menjadi media penyimpanan informasi sensitif dan penting yang berpengaruh besar terhadap pengguna dan organisasi. Dalam kondisi nyata, database dapat menjadi target penyerangan siber seperti peretasan, pencurian data, atau perusakan data, dimana hal tersebut dapat merugikan reputasi sejumlah pihak, khusunya pemilik aplikasi ataupun organisasi terkait dan juga dapat menyebabkan kerugian finansial. Oleh karena itu, sebagai pengembang sangat disarankan menerapkan berbagai teknik keamanan data, misalnya enkripsi data, manajemen hak akses pengguna yang ketat, dan autentikasi pengguna untuk melindungi informasi yang tersimpan di dalam database aplikasi. Selain itu, pengembang juga dapat melakukan praktik pemantauan dan audit secara rutin membantu dalam mendeteksi aktivitas yang mencurigakan dan berpotensi pada pelanggaran keamanan informasi. Dengan menjamin keamanan data pada database, pengembang tidak hanya melindungi data pengguna aplikasi, tetapi hal tersebut juga dapat memastikan kepercayaan dan kepuasan

pengguna terhadap aplikasi yang telah dikembangkan. Langkah-langkah dalam mewujudkan keamanan yang baik akan menjadi fondasi penting dalam membangun aplikasi yang tidak hanya bagus di sisi fungsional akan tetapi juga pada aspek keamanan dan keandalan.

Teknik pengamanan yang bisa diterapkan pada database diantaranya adalah teknik **enkripsi dan dekripsi data**. Teknik tersebut merupakan proses mengubah data menjadi format yang tidak dapat dibaca tanpa kunci yang sesuai dan bisa berlaku sebaliknya sesuai dengan algoritme yang diterapkan(Seth et al., 2022). Gambaran umum proses enkripsi dan dekripsi bisa dilihat pada Gambar 9. 4.



Gambar 9.4. Ilustrasi proses enkripsi dan dekripsi

Dengan menggunakan teknik enkripsi, data-data sensitif dan bersifat rahasia yang disimpan dalam database dilindungi dari akses yang tidak sah, bahkan jika data dalam database tersebut berhasil diretas akan sulit untuk diketahui. Proses enkripsi dapat diterapkan pada berbagai level penerapan, termasuk pada saat data ditransmisikan antara perangkat aplikasi dan penyimpanan data pada database serta saat data disimpan dalam database itu sendiri.

Penggunaan algoritma enkripsi yang mutakhir dan kunci (*key password*) yang dikelola dengan optimum akan menjadi sangat penting untuk menjaga kerahasiaan data. Dengan demikian, walaupun ada tindakan peretasan berhasil mendapatkan akses fisik ke sistem database, data tersebut tetap tidak dapat diakses tanpa informasi yang diperlukan untuk mendekripsi data tersebut.

Selain teknik enkripsi, **autentikasi** dan **kontrol akses** juga merupakan teknik yang populer dalam meningkatkan keamanan data pada database. Teknik autentikasi secara spesifik bertujuan untuk memastikan bahwa hanya pengguna yang terotorisasi yang dapat mengakses database.

Selain itu teknik tersebut juga yang dapat dilakukan dengan metode seperti penerapan kata sandi (*password*) yang kuat, otentikasi dua faktor (*two factor authentication*) atau penggunaan sertifikat digital. Setelah proses autentikasi berhasil, kontrol akses juga berperan dalam menentukan tingkat akses yang diberikan kepada pengguna. Hal tersebut bisa dilakukan dengan membatasi hak akses berdasarkan peran/aktor yang ada dalam organisasi. Ini bertujuan untuk memastikan bahwa pengguna hanya dapat mengakses data dan fungsi yang relevan dengan tugas mereka. Teknik tersebut juga dapat meminimalisir risiko penyalahgunaan atau kebocoran informasi. Dengan mengimplementasikan teknik-teknik ini secara menyeluruh, organisasi dapat melindungi data mereka dari berbagai ancaman dan memastikan bahwa database yang telah diimplementasikan beroperasi dengan aman.

Salah satu contoh pelanggaran keamanan yang signifikan di Indonesia terjadi pada tahun 2022 ketika data pribadi pengguna aplikasi peduliLindungi bocor (Dirgantara and Prabowo, 2022). Kasus tersebut menunjukkan bahwa walaupun aplikasi tersebut telah dirancang untuk tujuan positif, masih bisa dijadikan target peretasan sehingga dapat menyebabkan kebocoran data yang merugikan semua pengguna dan instansi pemerintah. Pembelajaran dari insiden ini adalah pentingnya penerapan protokol keamanan yang ketat dalam pengembangan aplikasi yang mengelola data pribadi.

9.5 Tren dan Teknologi Terbaru

Pengembangan aplikasi berbasis awan (*cloud*) telah menjadi tren yang semakin populer dalam beberapa tahun terakhir. Pengembangan dengan konsep cloud menawarkan fleksibilitas, skalabilitas, dan efisiensi biaya yang tinggi daripada dengan metode konfensional. Secara konsep, aplikasi yang dikembangkan di-*hosting* di lingkungan cloud, yang memungkinkan pengguna dapat mengakses layanan dan data dari mana saja dengan memanfaatkan koneksi internet. Selain itu pengguna juga dapat menggunakan aplikasinya tanpa perlu menginstal perangkat lunak secara lokal. Pengembang aplikasi saat ini sangat dimudahkan dikarenakan pengembang dapat fokus pada proses pengembangan aplikasi dan inovasi tanpa harus khawatir tentang pemeliharaan infrastruktur / server fisik (Winters et al., 2020). Beberapa penyedia layanan infrastruktur berbasis cloud yang populer saat ini diantaranya : Amazon Web Services (AWS), Microsoft Azure, atau Google Cloud Platform. Selain itu, aplikasi berbasis cloud juga dapat mendukung metode kolaborasi yang lebih baik dikarenakan tim pengembang dapat bekerja secara bersamaan di platform yang sama, mempercepat proses pengembangan dan pengujian. Di sisi lain, terdapat tantangan terkait keamanan, privasi, dan kepatuhan data tetap masih perlu diperhatikan untuk memastikan bahwa data pengguna tetap aman dan terjamin selama berada di lingkungan cloud.

Tren berikutnya terkait pengembangan aplikasi adalah pengembangan aplikasi dengan memanfaatkan teknologi kecerdasan buatan/*Artificial Intelligence* (AI) dan pembelajaran mesin/*Machine Learning* (ML). Dua teknologi tersebut menjadi fokus utama dalam industri teknologi saat ini. Dengan menggunakan algoritma ML, aplikasi dapat menganalisis pola data dengan volume yang cukup besar dan membuat prediksi atau rekomendasi yang lebih akurat. Sebagai contoh penerapan AI dan ML dalam aplikasi, terdapat rekomendasi di beberapa platform e-commerce dan chatbot cerdas yang memudahkan dalam layanani pelanggan. Selain itu, berkaitan dengan kemajuan dalam pemrosesan bahasa alami *Natural Language Processing* (NLP) dan visi komputer/ *Computer Vision*, aplikasi dapat memahami input

pengguna dengan cara yang lebih responsif. Meskipun implementasi AI dan ML menawarkan potensi besar untuk inovasi, pengembang juga dihadapkan pada tantangan seperti kebutuhan data berkualitas tinggi, isu etika, dan transparansi algoritma yang harus diatasi untuk memastikan bahwa aplikasi yang dihasilkan benar-benar bermanfaat dan dapat diandalkan.

Perkembangan terbaru dalam teknologi database saat ini ditandai dengan adopsi database NoSQL, database dalam memori, dan solusi cloud yang semakin meluas. Database NoSQL seperti MongoDB dan Cassandra menawarkan fleksibilitas yang lebih besar untuk menangani berbagai jenis data dan skala besar dan yang sangat penting jika berhubungan dalam konteks big data. Khususnya di Indonesia, salah satu contoh penerapan teknologi tersebut dapat dilihat pada aplikasi e-commerce seperti Tokopedia dan Bukalapak. E-commerce tersebut menggunakan database NoSQL untuk mengelola dan menganalisis data pengguna serta transaksi secara real-time. Selain itu, teknologi database dalam memori seperti Redis juga banyak digunakan oleh perusahaan fintech di Indonesia. Teknologi tesebut untuk menyediakan layanan pembayaran yang cepat dan responsif. Solusi database berbasis cloud, seperti Google Cloud SQL dan Amazon RDS menjadi semakin banyak diadopsi oleh startup dan perusahaan besar di Indonesia karena kemudahan dalam mengelola dan meningkatkan kapasitas penyimpanan data tanpa perlu investasi infrastruktur fisik yang besar. Di sisi lain, Dengan semua inovasi ini, teknologi database terus berkembang untuk memenuhi kebutuhan bisnis yang semakin kompleks dan dinamis di pasar Indonesia.

DAFTAR PUSTAKA

Dahri, F., Hanafi, A.M. El, Handoko, D., Wulan, N., 2022. Implementation of Microservices Architecture in Learning Management System E-Course Using Web Service Method. Sinkron : Jurnal dan Penelitian Teknik Informatika 7, 76–82. https://doi.org/10.33395/sinkron.v7i1.11229

Dirgantara, A., Prabowo, D., 2022. Data PeduliLindungi Bocor, Pemerintah Diminta Tak Saling Lempar Tanggung Jawab Halaman all - Kompas.com [WWW Document]. URL https://nasional.kompas.com/read/2022/11/18/05230361/data-pedulilindungi-bocor-pemerintah-diminta-tak-saling-lempar-tanggung?page=all (accessed 10.18.24).

Hadi, H.N., Aditya, A., Purwiantono, F.E., Listio, S.W., 2022. PENGUJIAN PERFORMA PADA WEBSITE LOMBA NASIONAL. Jurnal Informatika 22, 100–110. https://doi.org/https://doi.org/10.30873/ji.v22i1.3194

Mili, A., Tchier, F., 2015. Software Testing Concepts and Operations. John Wiley & Sons, Inc.

Mulyani, A., Oktiawati, U.Y., 2022. Implementasi Arsitektur Serverless Internet of Things pada Monitoring Cold Chain. Journal of Internet and Software Engineering 3.

Seth, B., Dalal, S., Jaglan, V., Le, D.N., Mohan, S., Srivastava, G., 2022. Integrating encryption techniques for secure data storage in the cloud. Transactions on Emerging Telecommunications Technologies 33, 1–24. https://doi.org/10.1002/ett.4108

Sommerville, I., 2010. Software Engineering - Ninth Edition, 9th ed, Software Engineering. Addison-Wesley Pearson Education, Inc. https://doi.org/10.1111/j.1365-2362.2005.01463.x

Winters, Titus., Manshreck, Tom., Wright, Hyrum., 2020. Software Engineering at Google. Upfront Books.

BAB 10
INTRODUCTION BIG DATA

Oleh Irpan Adiputra Pardosi

10.1 Pengantar Big Data

Di era digital saat ini, data dihasilkan setiap detik dari berbagai sumber, termasuk jaringan sosial, transaksi online, sensor, dan perangkat IoT (*Internet of Things*). Big data mengacu pada kumpulan data yang sangat besar dan kompleks sehingga sulit untuk diproses menggunakan database tradisional dan aplikasi pemrosesan data. Istilah "*Big Data*" tidak hanya mengacu pada volume data yang besar tetapi juga variabilitas, kecepatan, dan kompleksitas data tersebut.

Sejarah Big Data dimulai dengan konsep database dan konsep data warehouse pada tahun 1960an dan 1970an . Namun, konsep modern Big Data lebih erat kaitannya dengan kemunculan Internet dan perkembangan teknologi yang memungkinkan pengumpulan dan analisis data dalam jumlah besar dengan *throughput* tinggi.

Karakteristik Big Data mencakup 4V, yaitu:

1. **Volume**: Mengacu pada sejumlah besar data yang dihasilkan setiap hari dari berbagai sumber seperti media sosial, sensor, perangkat seluler, dll. Volume data ini seringkali mencapai petabyte atau bahkan exabyte.
2. **Velocity**: Kecepatan transfer data sangat cepat dan juga harus diproses dengan cepat. Hal ini mencakup seberapa sering data dibuat, dipublikasikan, dan diperbarui. Misalnya, data transaksi real-time, log streaming, dan data sensor IoT
3. **Variety**: Keanekaragaman data mengacu pada berbagai jenis data yang ada, mulai dari data terstruktur (database, spreadsheet) hingga data tidak terstruktur (video, teks, audio). Keberagaman ini membuat pemrosesan dan analisis data menjadi lebih kompleks

4. **Veracity**: Akurasi mengacu pada keakuratan dan keandalan data. Dengan banyaknya data yang dihasilkan, penting untuk memastikan bahwa data tersebut akurat, valid, dan dapat diandalkan untuk analisis dan pengambilan keputusan.

Pentingnya Big Data tidak hanya terletak pada jumlah data yang dikumpulkan tetapi juga pada kemampuan memproses dan menganalisis data tersebut untuk mengungkap wawasan yang dapat digunakan dalam pengambilan keputusan dan perubahan baru.

Implikasi Big Data dalam Era Digital

Penggunaan Big Data telah mengubah cara organisasi, bisnis, dan pemerintah beroperasi. Dalam bisnis, Big Data digunakan untuk mengoptimalkan operasi, meningkatkan pengalaman pelanggan, dan mengidentifikasi tren baru. Dalam bidang kesehatan, Big Data membantu dalam penelitian medis dan personalisasi perawatan. Dalam ilmu pengetahuan dan penelitian, Big Data memungkinkan analisis yang lebih mendalam dan pengujian hipotesis kompleks.

Dalam pengantar Big Data ini, kita akan mengeksplorasi bagaimana konsep telah berubah Mengubah cara kita hidup, bekerja, dan berpikir di era digital. Big data, dengan volume, kecepatan, dan variasinya yang luar biasa, memungkinkan kita memahami dunia dengan cara yang belum pernah mungkin dilakukan sebelumnya. Dalam buku "Big Data: A Revolution That Will Transform How We Live, Work and Think", **Viktor Mayer-Schönberger** dan **Kenneth Cukier** menggambarkan Big Data sebagai kekuatan transformatif yang akan berubah, misalnya seperti kesehatan, transportasi dan sains.

Big data tidak hanya terletak pada skalanya yang besar namun juga pada kemampuannya memberikan informasi yang menciptakan nilai baru. Dalam Ilmu Data untuk Bisnis, **Foster Provost** dan **Tom Fawcett** menekankan pentingnya wawasan analitis saat menggunakan Big Data. Mereka menjelaskan bagaimana teknik penambangan data dan pemikiran analitis dapat

membantu bisnis membuat keputusan berdasarkan data yang lebih cerdas.

Sejarah Big Data terkait erat dengan evolusi penyimpanan data, seperti yang dijelaskan **Ralph Kimball** dan **Margy Ross** dalam buku ” The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling” diterbitkan oleh Wiley pada tahun 2013. Buku ini merupakan panduan komprehensif tentang pemodelan dimensional untuk data warehousing dan business intelligence, yang sangat dihargai di bidangnya. Mereka menggambarkan bagaimana teknologi berkembang untuk menyimpan dan menganalisis kumpulan data yang semakin besar dan kompleks. Hal ini memberikan konteks untuk memahami betapa pentingnya big data bagi strategi informasi perusahaan.

McKinsey Global Institute, juga dalam artikelnya “Big Data: The Next Frontier for Innovation, Competition, and Productivity,” menyoroti bagaimana Big Data merupakan katalis bagi inovasi di berbagai industri. Melalui analisis data yang tepat, bisnis dapat mengidentifikasi tren yang belum terlihat, meningkatkan efisiensi, dan menciptakan produk dan layanan baru yang inovatif. Peran ilmuwan data di era Big Data tidak dapat diabaikan. Hal ini sejalan dengan yang dikatakan **Thomas H. Davenpor**t dan **D.J. Patil**, yang dipublikasikan di Harvard Business Review pada Oktober 2012, bahwa data memiliki kemampuan unik untuk menavigasi kompleksitas data besar, kata Pat dalam " Data Scientist: The Sexiest Job of the 21st Century" dari Harvard Business Review. Mereka menggunakan keterampilan statistik, pemrograman, dan kecerdasan bisnis untuk mengekstraksi nilai dari sejumlah besar data.

Big data juga menimbulkan tantangan, termasuk privasi, keamanan, dan pertimbangan etika. Menggabungkan potensi Big Data dengan kebutuhan untuk melindungi data pribadi individu merupakan perhatian utama bagi dunia usaha dan regulator. Diskusi ini menjawab pertanyaan tentang siapa pemilik data, bagaimana data tersebut digunakan, dan bagaimana memastikan manfaatnya lebih besar daripada risikonya. Adopsi big data akan membuka pintu ke dunia baru di mana data adalah sumber daya paling berharga. Dengan menggunakan analitik data besar, kita

dapat mengoptimalkan proses, memprediksi tren masa depan, dan menciptakan solusi inovatif untuk masalah yang kompleks. Saat kita memasuki abad ke-21, pemahaman dan penggunaan Big Data akan terus menjadi kunci keberhasilan organisasi dan kesejahteraan masyarakat secara keseluruhan.

10.2 Ekosistem Big Data

Ekosistem Big Data mengacu pada serangkaian teknologi dan metode komprehensif yang dirancang untuk mengelola, memproses, dan menganalisis data dalam jumlah besar dan kompleks. Ekosistem ini terdiri dari banyak komponen berbeda yang terintegrasi bersama, termasuk infrastruktur, platform analitik, aplikasi, serta pengelolaan dan penyimpanan data.

Komponen Ekosistem Big Data

1. Infrastruktur teknologi Big Data mencakup data sistem penyimpanan dan pemrosesan seperti database Hadoop, data NoSQL, dan database *Massively Parallel Processing*(MPP). Teknologi ini dirancang untuk mendukung pengelolaan data yang besar dan kompleks dengan mengiris dan mendistribusikan data secara paralel untuk pemrosesan yang efisien
2. Platform Analytics mencakup perangkat lunak yang mengintegrasikan dan menganalisis data untuk menciptakan wawasan data yang mendalam . Contoh platform ini termasuk Apache Spark, yang sangat populer dalam pemrosesan analitis karena kecepatan dan efisiensinya. Platform ini membantu bisnis membuat keputusan berdasarkan data, dengan alat seperti dasbor interaktif, laporan analitis, dan visualisasi data (IBM - Amerika Serikat).
3. Aplikasi di ekosistem Big Data menggunakan data yang dianalisis untuk memberikan wawasan yang optimal kepada pengguna akhir. Misalnya, aplikasi dalam layanan kesehatan dapat membantu ahli bedah saraf mendiagnosis dan merencanakan perawatan dengan lebih akurat menggunakan data dari pemindaian otak 3D
4. Manajemen Data mencakup aktivitas yang mengelola data sepanjang siklus hidupnya, dimulai dengan pengumpulan,

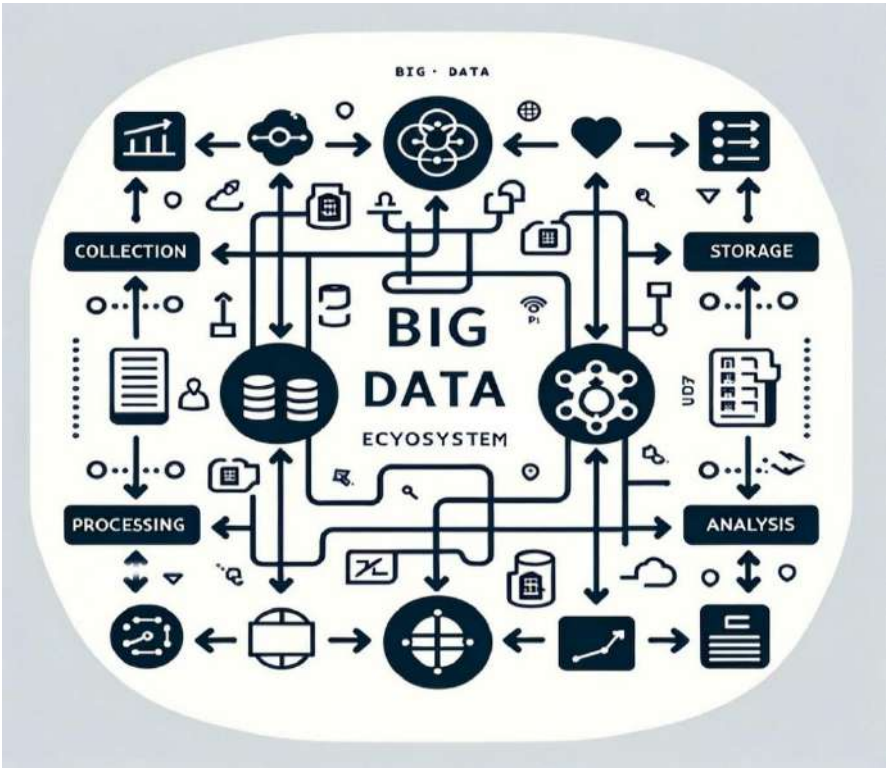
penyimpanan, pemrosesan, dan analisis. Manajemen ini juga melibatkan tugas-tugas seperti keamanan data, pencadangan, dan pemulihan. Sasaran utamanya adalah memastikan integritas, keamanan, dan ketersediaan data selama pengelolaan dan analisis

5. Alat dan layanan pendukung yaitu beragam alat dan layanan mendukung operasi dan analisis di seluruh Ekosistem Big Data, termasuk yang canggih perangkat lunak analisis data dan layanan berbasis cloud yang membantu meningkatkan skala operasi dengan lebih mudah. Misalnya, IBM Analytics menyediakan serangkaian alat yang mendukung analisis data besar dan menghasilkan wawasan untuk berbagai kebutuhan bisnis (IBM - Amerika Serikat).

Tabel 10.1. Teknologi Utama dalam Ekosistem Big Data

Teknologi	**Deskripsi**	**Aplikasi Utama**
Hadoop	Platform terdistribusi yang mendukung pengolahan data besar.	Pengolahan dan analisis data besar
Apache Spark	Mesin komputasi cepat untuk pengolahan data besar secara real-time.	Streaming dan machine learning
MongoDB	Database NoSQL yang mendukung penyimpanan data semi-terstruktur.	Aplikasi web skala besar
Apache Kafka	Platform untuk membangun real-time streaming data pipelines.	Streaming data
TensorFlow	Framework yang dikembangkan oleh Google untuk pembelajaran mesin.	Machine learning dan AI

Sumber: White, T. (2015)



Gambar 10.1. Ekosistem Big Data
(Sumber : White, T. (2015))

10.3 Integrasi Big Data dengan Basis Data Tradisional

Mengintegrasikan big data dengan database tradisional merupakan langkah penting dalam penggunaan teknologi data saat ini. Meskipun database tradisional dan teknologi big data sering dianggap beroperasi di dua dunia yang berbeda, integrasi antara keduanya dapat memberikan manfaat yang signifikan, menggabungkan keandalan dan fitur manajemen transaksi database relasional dengan kemampuan skala besar dan fleksibilitas solusi Big Data seperti seperti NoSQL dan Hadoop.

Pendekatan terintegrasi

1. Federasi dan virtualisasi data:
   Federasi data memungkinkan kueri mengakses data secara bersamaan dari database tradisional dan NoSQL, sehingga menyediakan lapisan data terpadu. Meskipun hal ini memberikan fleksibilitas, masih terdapat tantangan dalam menjaga konsistensi dan integritas data, terutama saat memproses transaksi real-time yang memerlukan integritas data yang tinggi. Selain itu, pengikatan data sering kali mengalami masalah kinerja karena persyaratan metadata yang ekstensif. Di sisi lain, virtualisasi data menawarkan solusi yang lebih fleksibel dengan menyediakan akses terpadu ke berbagai sumber data melalui satu database virtual, meskipun hal ini dapat menambah kompleksitas sistem
2. Integrasi menggunakan Apache Sqoop:
   Apache Sqoop adalah alat yang efektif untuk mengintegrasikan Hadoop dengan database relasional. Sqoop memfasilitasi ekspor dan impor data antara Hadoop dan database relasional menggunakan JDBC. Hal ini memungkinkan bisnis untuk memanfaatkan Hadoop sebagai solusi penyimpanan data yang hemat biaya sambil mempertahankan kemampuan kueri dan transaksi dari database relasional. Sqoop mendukung operasi paralel yang meningkatkan toleransi kesalahan dan mempercepat impor dan ekspor data.
3. Tantangan dan pertimbangan:
   Integrasi data besar menawarkan tantangan seperti mengelola volume data yang sangat besar, volume Data yang besar . harga dan variasi format data. Proses integrasi sering kali memerlukan alat dan teknologi khusus seperti NoSQL dan data lake, yang dirancang untuk menangani tantangan semacam ini dengan lebih efektif dibandingkan sistem manajemen basis data tradisional. Selain itu, memastikan kualitas dan tata kelola data yang baik adalah penting karena volume dan keragaman data yang terlibat.

4. Strategi implementasi:
   Strategi data Integrasi big data melibatkan perencanaan yang cermat, mengidentifikasi data yang tepat, memilih data yang tepat teknologi, dan menetapkan kebijakan tata kelola data yang kuat. Proses integrasi biasanya mencakup penggalian data dari berbagai sumber, pembersihan dan penyiapan data, mengubah data ke dalam format yang sesuai, dan menyimpan data terintegrasi dalam lingkungan yang aman dan dapat diakses.

Dengan pendekatan yang tepat, integrasi antara Big Data dan database tradisional tidak hanya mengatasi keterbatasan masing-masing teknologi namun juga membuka peluang baru dalam analisis data dan bisnis cerdas. Hal ini memungkinkan organisasi untuk lebih memahami dan merespons kebutuhan operasional dan strategis mereka secara lebih efektif.

10.4 Penyimpanan dan Manajemen Big Data

Di sini, strategi penyimpanan data yang efisien dalam konteks Big Data akan dibahas. Referensi yang dapat digunakan adalah buku "Data Lake for Enterprises: Building a Unified Data Architecture for Better Business Insights" oleh Tomcy John, dan dokumentasi resmi dari *Hadoop Distributed File System* (HDFS) dan teknologi penyimpanan Big Data lainnya.

Penyimpanan dan pengelolaan Big Data merupakan aspek penting dalam mengelola data dalam jumlah besar yang dihasilkan oleh bisnis modern. Dengan pertumbuhan data yang eksponensial, baik dalam volume maupun kompleksitas, pendekatan tradisional terhadap penyimpanan dan pengelolaan data sering kali terbukti tidak memadai. Berikut ini ikhtisar strategi dan teknologi untuk penyimpanan dan pengelolaan data besar yang efektif:

10.4.1 Penyimpanan Data Besar

Teknologi Penyimpanan Data Besar seperti Hadoop, Apache HBase, dan platform berbasis cloud seperti Snowflake dan AWS Play. peran penting dalam ekosistem Big Data. Misalnya, Hadoop memungkinkan pemrosesan Big Data terdistribusi di seluruh cluster

komputer, menggunakan model penyimpanan hemat biaya pada perangkat keras komoditas (Simpliearn.com). Snowflake menyediakan platform analitik yang memungkinkan akses real-time ke data yang diskalakan berdasarkan kebutuhan pengguna tanpa memerlukan manajemen infrastruktur yang ekstensif.

10.4.2 Manajemen Big Data

Manajemen Big Data melibatkan pengelolaan, administrasi, dan pengorganisasian Big Data, termasuk proses pengumpulan, pemrosesan, penyimpanan, dan analisis data. Ini bukan hanya tentang teknologi tetapi juga tentang penerapan kebijakan keamanan, aksesibilitas, dan kepatuhan yang efektif untuk memastikan bahwa data dapat diakses oleh pihak yang berwenang pada waktu yang tepat dan dalam format yang tepat.

Organisasi harus mengembangkan strategi pengelolaan data yang mencakup:

1. Identifikasi dan penentuan prioritas data: mengidentifikasi data yang paling penting dan menetapkan kebijakan penyimpanan data
2. Pencadangan dan pemulihan: Menerapkan rencana pencadangan dan pemulihan data yang kuat untuk melindungi data dari ancaman dunia maya dan bencana alam.
3. Kelola berbagai jenis data: Memahami dan mengelola berbagai jenis data, dari data terstruktur hingga tidak terstruktur, untuk memastikan bahwa data disimpan dan diakses seefisien mungkin.

Big data memerlukan arsitektur kuat yang mendukung tugas pengelolaan data seperti pengumpulan, pemrosesan, penyimpanan, serta integrasi dan persiapan data. Hal ini termasuk penggunaan alat manajemen metadata yang besar dan teknik integrasi data real-time untuk memastikan data selalu akurat dan terkini di semua sistem yang terhubung. Dengan meningkatnya laju pertumbuhan data, bisnis yang dapat mengelola data secara efektif akan memperoleh keunggulan kompetitif, memperoleh wawasan yang lebih mendalam, dan meningkatkan keputusan bisnis. Mengelola Big

Data dengan strategi dan teknologi yang tepat adalah kunci sukses ekonomi digital saat ini.

10.5 Analisis Big Data

Analisis data besar adalah proses penting yang memungkinkan bisnis memperoleh wawasan berguna dari kumpulan data besar dan kompleks yang mereka kumpulkan. Proses ini mencakup beberapa langkah penting untuk memastikan data diintegrasikan, dikelola, dan dianalisis secara efektif untuk mendukung pengambilan keputusan berdasarkan data.

10.5.1 Pengelolaan dan integrasi big data

Sebelum analisis dapat dilakukan. Untuk memulainya, data dari berbagai sumber harus diintegrasikan dan dikelola dengan baik. Integrasi data besar melibatkan penggabungan data dari berbagai sumber untuk memberikan informasi yang lengkap dan akurat, siap digunakan untuk analisis. Ini termasuk replikasi data, di mana data disalin dari sumber utama ke lokasi lain untuk memastikan akses cepat dan kinerja tinggi, serta menggunakan teknologi *change data capture* (CDC) untuk memperbarui Pembaruan mereplikasi data secara real-time saat data ditulis. diubah atau dihapus.

Setelah data dikumpulkan, langkah penting berikutnya adalah menyimpan data. Data dapat disimpan di data lake atau gudang data, melalui penggunaan platform seperti Hadoop, yang populer karena kemampuannya memproses data besar dengan cepat berkat model komputasi terdistribusi yang digunakan.

10.5.2 Proses Analisis Data Besar

Analisis data besar melibatkan eksplorasi dan analisis data untuk mengidentifikasi pola, hubungan, dan tren yang bermakna. Proses ini sering kali didukung oleh kecerdasan buatan (AI) dan pembelajaran mesin untuk mengotomatiskan proses dan memberikan rekomendasi yang mendalam, serta melakukan analisis prediktif dan memungkinkan interaksi bahasa.

Teknologi visualisasi data yang digunakan untuk membantu pengguna, analis, dan pemangku kepentingan memahami Big Data

dan berbagi informasi dalam organisasi dengan mudah. Hal ini diperlukan untuk memastikan bahwa data dapat diakses dan dimanfaatkan secara efektif oleh berbagai kelompok dalam organisasi. Analisis data besar adalah proses yang kompleks dan melibatkan beberapa langkah penting untuk mengubah data mentah menjadi wawasan yang berharga dan dapat ditindaklanjuti. Berikut ini ringkasan langkah-langkah detail proses analisis Big Data:

1. Identifikasi tujuan
   Sebelum mulai mengumpulkan data, penting untuk mendefinisikan dengan jelas tujuan analisis. Hal ini mencakup mengidentifikasi pertanyaan bisnis yang ingin Anda jawab dan hasil analisis data yang diharapkan.
2. Pengumpulan Data
   Langkah ini melibatkan pengumpulan data dari berbagai sumber, yang dapat mencakup data internal organisasi, data dari jaringan sosial, sensor, perangkat IoT, dan sumber lainnya. Proses ini juga harus memperhitungkan volume, kecepatan, dan variasi data yang dikumpulkan
3. Integrasi dan pengelolaan data
   Setelah data dikumpulkan, pengelolaan dan integrasi data perlu dilakukan. Hal ini melibatkan penggabungan data dari berbagai sumber ke dalam satu platform terpadu, sering kali menggunakan teknologi seperti Hadoop atau data lake. Proses ini juga mencakup pembersihan data dari kesalahan, duplikat, dan informasi yang tidak lengkap
4. Pengolahan dan Persiapan Data
   Data yang terintegrasi kemudian diolah dan disiapkan untuk dianalisis. Ini termasuk mengubah data ke dalam format yang sesuai dan mengindeksnya untuk memudahkan akses dan analisis. Pada tahap ini, data juga sering dianotasi dengan metadata untuk meningkatkan pencarian dan pengambilan data.
5. Analisis Data
   Pada tahap ini, data dianalisis untuk mengidentifikasi pola, tren, dan hubungan menggunakan statistik, pembelajaran mesin, atau teknik analisis lainnya. Proses ini dapat

mencakup analisis deskriptif, prediktif, dan preskriptif, bergantung pada sasaran yang ditetapkan pada langkah pertama.

6. Memvisualisasikan dan menyajikan hasil
   Hasil analisis data kemudian divisualisasikan menggunakan grafik, tabel, atau representasi visual lainnya untuk membantu pemahaman. Laporan dan dasbor dibuat untuk menyajikan hasil kepada pemangku kepentingan, mendukung pengambilan keputusan berdasarkan data.
7. Pengambilan keputusan dan tindakan
   Langkah terakhir dalam proses analisis big data adalah menggunakan wawasan yang diperoleh dari data untuk membuat keputusan strategis dan operasional. Hal ini dapat mencakup optimalisasi proses, meningkatkan kepuasan pelanggan, atau mengidentifikasi peluang pasar baru.

Mengikuti langkah-langkah berikut dalam proses analisis big data dapat membantu organisasi mengelola dan memanfaatkan kumpulan data besar mereka dengan lebih efektif untuk mendapatkan keunggulan kompetitif dan meningkatkan bisnis mereka. efisiensi.

10.5.2 Praktik terbaik analisis data besar

Untuk memastikan keberhasilan analisis data besar, penting bagi bisnis untuk mengadopsi beberapa praktik terbaik:

1. Mengembangkan dan menerapkan arsitektur yang andal: Penting untuk menciptakan arsitektur data yang kuat untuk mendukung misi pengelolaan data
2. Menggunakan metode Pengembangan tangkas: Pendekatan ini memastikan bahwa tim dapat dengan cepat beradaptasi terhadap perubahan kebutuhan bisnis dan teknologi
3. Menerapkan langkah-langkah keamanan data yang kuat: Melindungi data dari akses dan serangan tidak sah sangat penting untuk menjaga integritas dan keamanan data
4. Penggunaan data secara etis: menjamin bahwa data dikumpulkan, disimpan, dan digunakan dengan cara yang mematuhi peraturan dan menghormati privasi pengguna

5. Pemantauan dan optimalisasi berkelanjutan: memantau kinerja sistem data secara terus-menerus dan melakukan penyesuaian untuk meningkatkan kinerja data

Dengan menerapkan strategi ini, organisasi dapat memastikan bahwa mereka dapat mengelola dan menganalisis data besar secara efektif, memungkinkan mereka untuk membuat lebih cerdas keputusan dan mempercepat inovasi.

10.6 Keamanan dan Privasi Big Data

Keamanan dan privasi Big Data merupakan aspek penting yang harus dikelola secara hati-hati untuk memastikan perlindungan data yang efektif dan persyaratan kepatuhan terpenuhi. Berikut beberapa rekomendasi praktik terbaik untuk mengelola keamanan dan privasi big data:

1. Menerapkan enkripsi data
   Enkripsi data adalah salah satu strategi utama untuk melindungi data dari akses tidak sah. Enkripsi harus diterapkan pada data saat istirahat (data at rest) dan data dalam perjalanan (data in transit) untuk memastikan keamanan dan integritas informasi sensitif. Penggunaan teknologi yang dapat digunakan pada tahap ini seperti:
   a. *Encryption Algorithm* seperti *Advanced Encryption Standard* (AES) dapat digunakan untuk enkripsi data. AES adalah standar enkripsi yang kuat dan banyak digunakan dalam industri.
   b. *Key Management Systems*: Alat seperti AWS Key Management Service (KMS) atau Azure Key Vault membantu mengelola kunci enkripsi, memastikan bahwa data tetap aman dan hanya dapat diakses oleh pengguna yang berwenang.
2. Menggunakan Penyimpanan Data yang Aman
   Solusi penyimpanan data yang aman sangat penting untuk melindungi data besar dari ancaman fisik dan virtual. Teknologi seperti penyimpanan cloud yang aman, klasifikasi data, kontrol akses, dan sistem pencegahan kehilangan data dapat meningkatkan keamanan data yang disimpan. Selain

itu, penting untuk membuat cadangan data secara teratur dan memiliki rencana pemulihan bencana untuk mengurangi risiko kehilangan data. Penggunaan teknologi yang dapat digunakan pada tahap ini seperti:

a. *Secure Cloud Storage Solutions*: Layanan seperti Amazon S3 dan Google Cloud Storage menawarkan opsi penyimpanan yang aman dengan dukungan kebijakan keamanan yang kuat dan enkripsi data.
b. *Data Loss Prevention (DLP) Systems:* Alat seperti Symantec DLP atau McAfee Total Protection for DLP dapat digunakan untuk mencegah kehilangan data melalui pemantauan, pendeteksian, dan blokiran pemindahan data sensitif.

3. Kontrol dan manajemen akses

   Kontrol akses memainkan peran penting dalam memastikan keamanan dan integritas data besar. Dengan menentukan hak istimewa dan peran pengguna, bisnis dapat membatasi akses ke data sensitif hanya untuk karyawan yang berwenang. Mekanisme autentikasi yang kuat, seperti autentikasi multifaktor, harus diterapkan untuk mencegah upaya akses tidak sah. Penggunaan teknologi yang dapat digunakan pada tahap ini seperti:

   a. *Access Control Systems*: Alat seperti Microsoft Active Directory atau Okta membantu dalam mendefinisikan dan mengelola hak akses pengguna.
   b. *Multi-factor Authentication* (MFA) Tools: Solusi seperti Google Authenticator atau Duo Security meningkatkan keamanan dengan memerlukan beberapa metode autentikasi untuk mengakses data.

4. Pengujian dan pemantauan rutin

   Pengujian keamanan rutin merupakan bagian integral dari strategi keamanan data besar secara keseluruhan. Melalui audit, bisnis dapat mengidentifikasi kerentanan, mengevaluasi kepatuhan kebijakan keamanan, dan mendeteksi aktivitas tidak sah. Penting juga untuk menerapkan sistem pemantauan dan peringatan secara real-time untuk mengidentifikasi dan merespons insiden

keamanan dengan cepat. Penggunaan teknologi yang dapat digunakan pada tahap ini seperti:

a. *Security Information and Event Management (SIEM) Systems*: Alat seperti Splunk atau IBM QRadar menyediakan fitur pemantauan keamanan real-time dan kemampuan audit untuk mendeteksi dan merespons insiden keamanan.
b. *Vulnerability Management Tools*: Alat seperti Tenable Nessus atau Qualys membantu dalam pemindaian dan penilaian kerentanan untuk memastikan bahwa sistem terlindungi dari ancaman terkini.

5. Privasi dan kepatuhan

   Masalah privasi dan kepatuhan merupakan tantangan penting dalam keamanan data besar. Dengan meningkatnya peraturan perlindungan data, seperti GDPR dan CCPA, bisnis harus memastikan bahwa pengumpulan, pemrosesan, dan penyimpanan data mematuhi persyaratan hukum. Hal ini memerlukan persetujuan eksplisit dari individu sebelum mengumpulkan dan menggunakan data mereka, serta menerapkan kebijakan privasi yang ketat dan praktik penanganan data yang transparan. Penggunaan teknologi yang dapat digunakan pada tahap ini seperti:

   a. *Compliance Management Tools*: Alat seperti OneTrust atau TrustArc dapat membantu organisasi memastikan bahwa kegiatan pengumpulan dan pengolahan data mereka mematuhi peraturan seperti GDPR.
   b. *Privacy Management Software*: Alat seperti WireWheel atau BigID menyediakan platform manajemen privasi yang membantu organisasi dalam memetakan, mengelola, dan melindungi data pribadi pengguna sesuai dengan kebijakan privasi.

Mengadopsi praktik terbaik ini dapat membantu organisasi mengelola risiko yang terkait dengan big data dan memastikan hal tersebut mereka dapat melindungi data berharga dari akses tidak sah atau pelanggaran data. Untuk informasi selengkapnya tentang keamanan dan privasi big data, Anda dapat berkonsultasi dengan

sumber seperti Cloud Security Alliance dan Data Institute, yang memberikan panduan mendetail tentang topik ini.

10.7 Studi Kasus dan Praktik Terbaik

Studi kasus nyata tentang implementasi Big Data dalam berbagai industri dan praktik terbaik dalam pengelolaan dan analisis Big Data akan dibahas. Referensi yang dapat digunakan adalah laporan industri dari perusahaan-perusahaan terkemuka seperti IBM, Microsoft, dan Google, Walmart serta jurnal ilmiah terkait.

Big data telah menjadi topik utama dalam teknologi informasi dan analisis data, memberikan informasi umum tentang berbagai sektor industri. Dengan mengelola dan menganalisis data dalam jumlah besar, beragam, dan terus bertambah, bisnis dapat memperoleh keunggulan kompetitif yang signifikan.

10.7.1 Studi Kasus Big Data

Studi kasus tentang big data seringkali melibatkan penggunaan data besar dalam sektor-sektor seperti kesehatan, keuangan, ritel, dan transportasi. Sebagai contoh:

1. Inovasi layanan kesehatan
   Rumah sakit menggunakan Big Data untuk memantau dan meningkatkan kualitas pelayanan pasien. Misalnya, dengan menganalisis data pasien rawat inap, rumah sakit dapat mengidentifikasi tren yang menyebabkan pasien masuk kembali dan mengembangkan strategi untuk menguranginya. Analisis big data memungkinkan rumah sakit meningkatkan efisiensi operasional dan hasil kesehatan pasien.
2. Optimalisasi rantai pasokan di ritel
   Perusahaan ritel besar Pengecer mengintegrasikan Big Data untuk mengoptimalkan operasi rantai pasokan mereka. Dengan menganalisis data historis dan data transaksi real-time, bisnis dapat memprediksi tren pembelian, mengelola inventaris secara efektif, dan dengan cepat merespons perubahan kebutuhan konsumen.

3. Pengembangan produk berbasis data di industri otomotif
   Industri otomotif menggunakan data besar untuk membangun strategi pengembangan produk baru. Dengan menganalisis data sensor kendaraan dan umpan balik pelanggan, produsen mobil dapat meningkatkan keselamatan, kenyamanan, dan efisiensi kendaraan.

10.7.2 Praktek Terbaik dalam Pengelolaan Big Data

Berikut dijabarkan beberapa praktik terbaik untuk pengelolaan big data dalam berbagai hal:

1. Pengaturan Tata Kelola Data yang Efektif
   Penting untuk menetapkan kebijakan tata kelola data yang kuat untuk memastikan data tetap aman, berkualitas, dan dapat diakses oleh pihak yang berwenang. Tata kelola yang baik meliputi pengaturan privasi, keamanan data, dan kepatuhan terhadap regulasi yang relevan.
2. Mengintegrasikan Teknologi Terbaru
   Pemanfaatan platform dan alat analitik terkini, seperti Hadoop, Spark, dan platform cloud, dapat meningkatkan kemampuan analisis Big Data. Teknologi ini membantu dalam mengelola volume data yang besar, mempercepat analisis, dan memberikan fleksibilitas dalam skalabilitas sumber daya.
3. Pengembangan Keterampilan Analitik Tim
   Perusahaan harus berinvestasi dalam pelatihan dan pengembangan keterampilan analitik tim mereka. Memiliki tim yang terampil dalam statistik, machine learning, dan analisis data adalah kunci untuk memanfaatkan sepenuhnya potensi Big Data.
4. Kerjasama Lintas Fungsi
   Kolaborasi antardepartemen dapat membantu memastikan bahwa wawasan dari Big Data dimanfaatkan secara efektif untuk meningkatkan keputusan strategis dan operasional. Ini juga mendukung inovasi berkelanjutan dan peningkatan produk atau layanan.

Praktik terbaik dan studi kasus dalam Big Data menunjukkan bahwa teknologi ini tidak hanya revolusioner dalam cara data dikumpulkan dan dianalisis, tetapi juga dalam bagaimana ia mengubah keputusan bisnis di semua level. Dengan pendekatan yang tepat, Big Data dapat menjadi katalis yang mendorong inovasi dan efisiensi di banyak industri.

10.8 Tantangan dan Masa Depan Big Data

Saat ini, big data menghadapi beberapa tantangan yang cukup kompleks. Berikut adalah beberapa tantangannya (Katal, 2013):

1. **Kapasitas infrastruktur**: Dengan pertumbuhan data yang eksponensial, banyak organisasi masih kesulitan meningkatkan infrastruktur mereka agar mampu menangani data dalam jumlah besar. Misalnya, perusahaan teknologi harus terus meningkatkan kapasitas server dan teknologi penyimpanan untuk menghindari latensi dalam pemrosesan data.
2. **Integrasi Data *Real-time***: Dengan meningkatnya kebutuhan akan pengambilan keputusan berbasis data waktu nyata, seperti di bidang tanggap bencana atau keamanan siber, organisasi memerlukan Sistem yang dapat secara efektif mengintegrasikan dan menganalisis data masuk tanpa penundaan.
3. **Diversitas Sumber Data**: Meningkatnya penggunaan media sosial, perangkat IoT, dan aplikasi seluler menambah variabilitas dan volume data. Organisasi harus mengembangkan algoritme yang mampu memproses dan mengintegrasikan jenis data yang sangat berbeda ini, seringkali tidak terstruktur dan terdistribusi.
4. **Masalah Privasi dan Keamanan**: Meningkatnya serangan siber dan pelanggaran data memaksa dunia usaha untuk memprioritaskan keamanan dan privasi data. Misalnya, penerapan GDPR di Eropa telah mengubah cara bisnis mengelola dan melindungi data pengguna.
5. **Analisis Data yang Mendalam**: Dibutuhkan alat analisis yang lebih canggih yang dapat mengidentifikasi pola

tersembunyi dalam data besar. Perkembangan teknologi seperti pembelajaran mendalam dan algoritma prediktif menyederhanakan kompleksitas ini namun masih memerlukan sumber daya yang signifikan untuk diterapkan dan dipelihara.

Sedang masa depan big data terlihat sangat menjanjikan dengan beberapa perkembangan yang mungkin terjadi (Mayer, 2013):

1. Teknologi AI dan Machine Learning: Aplikasi AI dan pembelajaran mesin akan semakin meningkatkan kemampuan analisis data besar, memungkinkan perkiraan yang lebih akurat dan personalisasi layanan.
2. Infrastruktur Cloud yang Lebih Efisien: Dengan meningkatnya komputasi awan, penyimpanan data dan biaya pemrosesan diharapkan menjadi lebih terjangkau, membantu lebih banyak organisasi mengadopsi big data.
3. Big Data dan IoT: Integrasi antara IoT (Internet of Things) dan Big Data akan menciptakan volume data yang lebih besar dan wawasan yang lebih mendalam, terutama di industri seperti manufaktur, layanan kesehatan, dan transportasi.
4. Regulasi Data yang Lebih Ketat: Dengan meningkatnya kekhawatiran terhadap privasi, peraturan mengenai penggunaan dan pemrosesan data diperkirakan akan menjadi semakin ketat, sehingga memengaruhi cara organisasi mengelola data.
5. Data sebagai Layanan (DaaS): Model bisnis baru seperti Data sebagai Layanan (DaaS) akan berkembang, memungkinkan bisnis untuk mengakses data dan analisis yang relevan dari layanan penyedia layanan tanpa memerlukan infrastruktur yang luas.

DAFTAR PUSTAKA

Davenport, Thomas H., and D. J. Patil, Data Scientist: The Sexiest Job of the 21st Century Diakses pada 2 April 2024 dari https://www.hbs.edu/faculty/Pages/item.aspx?num=43110

Kimball, R., & Ross, M. (2013). The data warehouse toolkit: The definitive guide to dimensional modeling (3rd ed.). Wiley.

Katal, A., Wazid, M., & Goudar, R. H. (2013). Big Data: Issues, Challenges, Tools, and Good Practices. 2013 Sixth International Conference on Contemporary Computing (IC3), 404-409. https://doi.org/10.1109/IC3.2013.6612229

Mayer-Schonberger, V. and Cukier, K. (2013) Big Data: A Revolution That Will Transform How We Live, Work, and Think. *Houghton Mifflin Harcourt*, Boston.

White, T. (2015). Hadoop: The Definitive Guide (4th ed.). Sebastopol, CA: O'Reilly Media.

BAB 11
DATA ANALYSIS

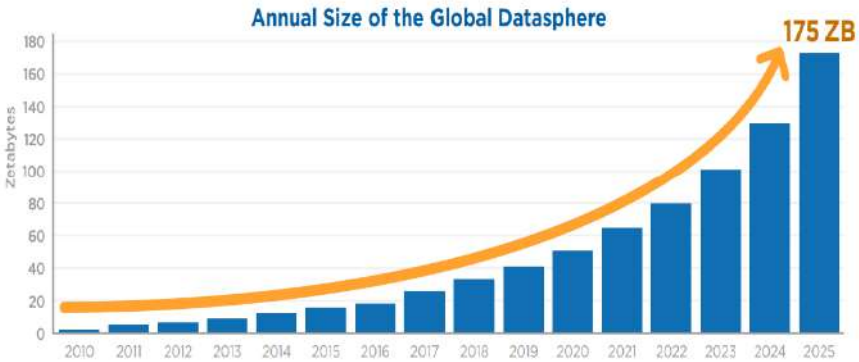
Oleh Jarot Budiasto

11.1 Pendahuluan

Di era digital yang penuh dengan informasi, data telah menjadi aset yang sangat berharga bagi individu, organisasi, hingga pemerintah. Data bukan lagi sekadar kumpulan angka atau teks, melainkan sumber wawasan yang mampu mengungkap pola, tren, dan potensi yang sebelumnya tersembunyi. Dalam konteks ini, analisis data memegang peranan sentral sebagai proses sistematis untuk memahami dan menginterpretasikan data demi mendukung pengambilan keputusan yang lebih baik.

Analisis data didefinisikan sebagai proses pengumpulan, pengorganisasian, pembersihan, transformasi, dan interpretasi data untuk memperoleh informasi yang berguna dan mendukung keputusan strategis (Islam, 2020). Proses ini mencakup berbagai teknik statistik, komputasional, dan visual untuk menemukan pola tersembunyi dan hubungan antar variabel dalam data. Analisis data tidak hanya membantu menjawab pertanyaan "apa yang terjadi", tetapi juga "mengapa hal itu terjadi", "apa yang akan terjadi", dan "apa yang harus dilakukan".

Seiring dengan perkembangan teknologi, analisis data telah mengalami transformasi besar. Pada tahun 2018, volume data global mencapai 33 zettabyte dan diperkirakan akan tumbuh menjadi 175 zettabyte pada tahun 2025 (Rydning, Reinsel and Gantz, 2018). Pertumbuhan ini didorong oleh meningkatnya penggunaan perangkat IoT (*Internet of Things*), aplikasi berbasis AI, dan aktivitas digital lainnya. Di tengah pertumbuhan data yang eksponensial ini, analisis data menjadi alat yang esensial untuk menyaring informasi relevan dari data mentah.



Gambar 11.1. Pertumbuhan Volume Data Global
(Sumber : *IDC DataAge 2025*)

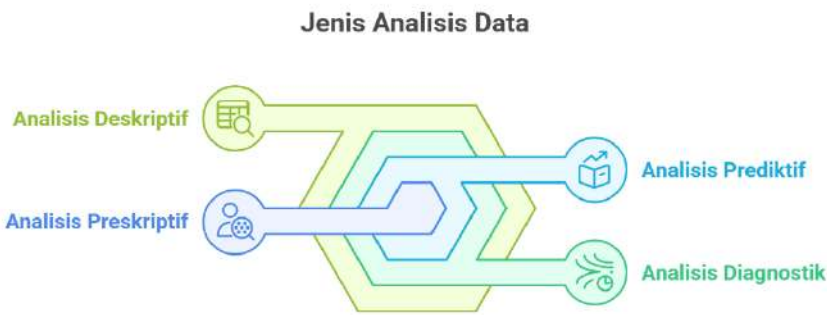
Analisis data telah berkembang pesat seiring dengan kemajuan teknologi. Dahulu, analisis data terbatas pada metode statistik sederhana yang digunakan untuk menyusun laporan. Kini, dengan hadirnya teknologi seperti *machine learning* dan *big data*, analisis data mampu menjawab pertanyaan-pertanyaan yang lebih kompleks, memprediksi kejadian di masa depan, hingga memberikan rekomendasi tindakan yang optimal (L'Heureux *et al.*, 2017).

Sebagai fondasi pengambilan keputusan berbasis data (*data-driven decision making*), analisis data telah diaplikasikan di berbagai bidang seperti kesehatan, keuangan, pendidikan, hingga pemerintahan. Kemampuan untuk menganalisis data tidak hanya menjadi keunggulan kompetitif bagi organisasi tetapi juga menjadi kebutuhan dasar di tengah persaingan global.

Bab ini akan membahas analisis data secara komprehensif, mulai dari jenis-jenis analisis data, metodologi yang digunakan, alat-alat yang mendukung proses analisis, hingga aplikasi praktisnya di berbagai sektor. Dengan pemahaman yang mendalam, pembaca diharapkan dapat mengapresiasi pentingnya analisis data dan mengaplikasikan konsep-konsep yang disajikan untuk menyelesaikan berbagai tantangan di dunia nyata.

11.2 Jenis-Jenis Analisis Data

Analisis data adalah proses sistematis yang digunakan untuk memahami informasi yang terkandung dalam data. Proses ini terdiri dari beberapa jenis analisis, yang masing-masing memiliki fokus, teknik, dan tujuan yang berbeda. Jenis-jenis analisis ini memberikan kerangka kerja yang terstruktur untuk memahami data dan menghasilkan wawasan yang dapat digunakan dalam pengambilan keputusan. Berikut adalah penjelasan setiap jenis analisis data.



Gambar 11.2. Jenis-jenis Data Analisis

11.2.1 Analisis Deskriptif

Analisis deskriptif merupakan langkah pertama dalam proses analisis data. Analisis ini bertujuan untuk memberikan gambaran keseluruhan dari data yang ada tanpa membuat interpretasi yang mendalam. Proses ini membantu dalam memahami data secara umum, seperti mengenali pola, tren, atau distribusi data.

Analisis deskriptif menggunakan berbagai teknik statistik sederhana seperti *mean*, *median*, *modus*, dan standar deviasi. Selain itu, visualisasi data seperti grafik batang, diagram lingkaran, dan histogram sering digunakan untuk memperjelas pola yang ada (Shardt, 2022). Misalnya, grafik batang dapat menunjukkan penjualan per kategori produk, sedangkan histogram membantu memahami distribusi pendapatan pelanggan dalam suatu wilayah.

Dalam praktiknya, analisis deskriptif digunakan oleh organisasi untuk merangkum informasi dari laporan rutin, seperti total

pendapatan harian atau jumlah pelanggan per bulan. Sebagai contoh, laporan penjualan e-commerce biasanya mencakup data seperti jumlah pesanan, produk terlaris, dan wilayah dengan penjualan tertinggi.

11.2.2 Analisis Diagnostik

Setelah memahami data secara umum melalui analisis deskriptif, langkah selanjutnya adalah mencari tahu penyebab dari pola atau tren tertentu. Analisis diagnostik bertujuan untuk menjawab pertanyaan "mengapa sesuatu terjadi?" dengan menganalisis hubungan antara variabel-variabel yang relevan.

Analisis diagnostik sering menggunakan teknik seperti analisis korelasi untuk melihat hubungan antara dua variabel dan *Root Cause Analysis* (RCA) untuk menggali penyebab utama dari suatu masalah. Sebagai contoh, ketika perusahaan mendeteksi penurunan jumlah pelanggan aktif, analisis diagnostik dapat digunakan untuk menemukan alasan di balik fenomena tersebut, seperti perubahan desain produk atau peningkatan harga yang tidak diterima dengan baik oleh pelanggan.

Pendekatan ini juga sangat berguna dalam industri kesehatan. Misalnya, dokter dapat menggunakan analisis diagnostik untuk mengidentifikasi faktor penyebab utama dari meningkatnya kasus penyakit tertentu dalam populasi.

11.2.3 Analisis Prediktif

Analisis prediktif memanfaatkan data historis untuk memperkirakan kejadian di masa depan. Pendekatan ini sering digunakan dalam pengambilan keputusan strategis, terutama di bidang keuangan, pemasaran, dan logistik. Dengan menggunakan model statistik dan algoritma pembelajaran mesin, analisis prediktif dapat menghasilkan proyeksi yang akurat.

Teknik umum yang digunakan dalam analisis prediktif meliputi regresi linier untuk memodelkan hubungan antara variabel, *time series forecasting* untuk memprediksi tren masa depan, dan algoritma seperti *Random Forest* atau *Neural Networks* untuk menangani data yang kompleks.

Sebagai contoh, perusahaan logistik sering menggunakan analisis prediktif untuk memproyeksikan jumlah pengiriman selama periode liburan. Dengan informasi ini, perusahaan dapat mempersiapkan sumber daya, seperti tenaga kerja dan kendaraan, untuk mengelola peningkatan permintaan.

11.2.4 Analisis Preskriptif

Analisis preskriptif adalah jenis analisis yang bertujuan untuk memberikan rekomendasi tindakan berdasarkan hasil analisis prediktif atau data lainnya. Analisis ini menjawab pertanyaan "apa yang harus dilakukan?" dan membantu organisasi mengambil keputusan yang optimal.

Pendekatan ini menggunakan teknik seperti simulasi untuk memodelkan berbagai skenario dan optimasi untuk mencari solusi terbaik. Contohnya, supermarket dapat menggunakan analisis preskriptif untuk menentukan tingkat stok optimal selama musim liburan, sehingga mengurangi risiko kekurangan atau kelebihan stok.

Selain itu, platform e-commerce seperti Amazon atau Tokopedia sering menggunakan analisis preskriptif untuk merekomendasikan produk kepada pelanggan berdasarkan preferensi mereka, yang meningkatkan peluang pembelian.

Tabel 11.1. Tabel Perbandingan Jenis-Jenis Analisis Data

Jenis Analisis	Fokus Utama	Teknik Utama	Contoh Kasus
Deskriptif	Apa yang terjadi?	Statistik deskriptif, visualisasi	Laporan bulanan, analisis tren
Diagnostik	Mengapa terjadi?	Korelasi, analisis akar masalah	Penyebab penurunan performa
Prediktif	Apa yang akan terjadi?	Regresi, pembelajaran mesin	Peramalan permintaan, deteksi penipuan
Preskriptif	Apa yang harus dilakukan?	Simulasi, optimasi	Strategi pemasaran, manajemen rantai pasok

11.3 Metodologi dalam Analisis Data

Analisis data yang baik membutuhkan pendekatan yang terstruktur untuk memastikan hasil yang dihasilkan akurat, relevan, dan dapat diandalkan. Metodologi analisis data melibatkan serangkaian langkah yang saling terkait, dimulai dari penentuan tujuan hingga visualisasi dan interpretasi hasil. Proses ini tidak hanya membantu menjamin kualitas analisis, tetapi juga memastikan bahwa wawasan yang diperoleh dapat mendukung pengambilan keputusan yang lebih baik (Safitri *et al.*, 2024). Berikut adalah uraian mendalam dari setiap langkah dalam metodologi analisis data:

11.3.1 Menentukan Tujuan

Langkah pertama dalam analisis data adalah menentukan tujuan yang ingin dicapai. Tujuan yang jelas menjadi fondasi dari keseluruhan proses analisis. Tanpa pemahaman yang mendalam tentang masalah yang ingin diselesaikan atau pertanyaan yang ingin dijawab, analisis cenderung tidak terarah dan berpotensi menghasilkan kesimpulan yang tidak relevan.

Misalnya, dalam konteks bisnis, sebuah perusahaan mungkin ingin mengetahui faktor utama yang memengaruhi kepuasan pelanggan berdasarkan hasil survei. Sementara itu, dalam penelitian akademik, tujuan dapat berupa mengidentifikasi hubungan antara dua variabel dalam sebuah dataset. Penentuan tujuan ini juga mencakup pengembangan hipotesis awal atau pertanyaan penelitian yang akan diuji melalui analisis.

11.3.2 Pengumpulan Data

Data adalah bahan mentah dalam analisis, sehingga langkah pengumpulan data menjadi sangat penting. Data dapat diperoleh melalui berbagai sumber, baik secara langsung maupun tidak langsung. Data primer dikumpulkan langsung dari lapangan melalui survei, wawancara, observasi, atau eksperimen. Sementara itu, data sekunder diambil dari sumber yang telah ada, seperti laporan penelitian, jurnal akademik, atau database online.

Pada tahap ini, penting untuk memastikan bahwa data yang dikumpulkan relevan, representatif, dan dapat dipercaya. Misalnya,

dalam analisis penjualan, data transaksi dari sistem *point-of-sale* (POS) dapat digunakan untuk memahami pola pembelian pelanggan. Selain itu, teknologi modern, seperti *web scraping*, dapat digunakan untuk mengumpulkan data secara otomatis dari sumber daring.

11.3.3 Pembersihan Data

Data yang dikumpulkan sering kali tidak sempurna. Data mentah dapat mengandung nilai yang hilang (*missing values*), data yang tidak konsisten, atau bahkan outlier yang dapat memengaruhi akurasi analisis. Oleh karena itu, pembersihan data menjadi langkah krusial dalam metodologi ini.

Pembersihan data mencakup identifikasi dan penanganan masalah seperti duplikasi data, data yang tidak lengkap, atau kesalahan pencatatan. Misalnya, nilai yang hilang dapat diisi menggunakan metode tertentu, seperti mean, median, atau interpolasi, tergantung pada sifat data. Selain itu, outlier yang mencurigakan harus dievaluasi untuk menentukan apakah perlu dihapus atau disesuaikan.

11.3.4 Transformasi Data

Setelah data dibersihkan, langkah berikutnya adalah mengubah data mentah menjadi format yang lebih sesuai untuk dianalisis. Transformasi data melibatkan berbagai proses, seperti normalisasi, encoding, dan agregasi. Langkah ini membantu menyusun data agar lebih terorganisir dan siap untuk diaplikasikan pada teknik analisis tertentu.

Normalisasi, misalnya, dilakukan untuk menyamakan skala data sehingga setiap variabel memiliki bobot yang setara. Sementara itu, encoding digunakan untuk mengonversi data kategorikal menjadi data numerik agar dapat dianalisis secara statistik atau menggunakan algoritma pembelajaran mesin. Agregasi data sering diterapkan untuk merangkum data berdasarkan kategori tertentu, seperti menghitung rata-rata penjualan bulanan.

11.3.5 Analisis Data

Setelah data siap, proses analisis dilakukan untuk menggali wawasan dari data tersebut. Teknik yang digunakan bergantung pada tujuan dan jenis data yang tersedia. Analisis dapat bersifat deskriptif, diagnostik, prediktif, atau preskriptif, dengan setiap jenis memiliki fokus yang berbeda.

Misalnya, analisis deskriptif digunakan untuk menjelaskan apa yang terjadi dengan meringkas data menggunakan statistik sederhana seperti rata-rata dan standar deviasi. Analisis prediktif, di sisi lain, menggunakan algoritma pembelajaran mesin seperti regresi linear atau pohon keputusan untuk memprediksi kejadian di masa depan. Dalam beberapa kasus, analisis kualitatif juga dilakukan untuk memahami pola atau tema dari data non-numerik, seperti tanggapan survei terbuka.

11.3.6 Validasi Hasil

Setelah analisis selesai, langkah berikutnya adalah memastikan bahwa hasil yang diperoleh dapat dipercaya dan konsisten. Validasi hasil adalah proses mengevaluasi keakuratan dan keandalan metode atau model yang digunakan. Salah satu teknik umum dalam validasi adalah *cross-validation*, di mana dataset dibagi menjadi data pelatihan dan pengujian untuk mengukur performa model.

Selain itu, metrik evaluasi seperti akurasi, *precision*, *recall*, dan *mean squared error* (MSE) digunakan untuk menilai sejauh mana model dapat memprediksi atau menjelaskan data. Misalnya, dalam analisis klasifikasi penyakit daun padi, akurasi model dapat menjadi metrik utama untuk menentukan efektivitas metode yang digunakan.

11.3.7 Visualisasi dan Interpretasi

Hasil analisis tidak hanya harus akurat, tetapi juga mudah dipahami oleh pemangku kepentingan. Oleh karena itu, visualisasi data menjadi langkah penting dalam menyajikan temuan-temuan utama. Grafik, tabel, dan dasbor interaktif adalah beberapa alat yang dapat digunakan untuk mengkomunikasikan hasil secara efektif.

Sebagai contoh, tren penjualan dapat divisualisasikan menggunakan grafik garis untuk menunjukkan pola perubahan dari waktu ke waktu. Selain itu, interpretasi hasil dalam bentuk narasi membantu menjelaskan temuan secara lebih mendalam, memberikan konteks, dan merekomendasikan tindakan berdasarkan data.

11.4 Alat untuk Analisis Data

Dalam proses analisis data, berbagai alat digunakan untuk membantu dalam pengumpulan, pemrosesan, analisis, dan visualisasi data. Alat-alat ini dirancang untuk memenuhi kebutuhan yang beragam, mulai dari analisis sederhana hingga kompleks, serta mencakup berbagai platform seperti spreadsheet, bahasa pemrograman, perangkat lunak statistik, alat visualisasi, alat big data, dan basis data. Pemilihan alat yang tepat sangat bergantung pada kebutuhan analisis, kompleksitas data, dan keahlian pengguna. Berikut ini adalah beberapa alat yang umum digunakan dalam analisis data.

11.4.1 Alat Spreadsheet

Alat Spreadsheet seperti Microsoft Excel dan Google Sheets adalah perangkat dasar yang sering digunakan untuk analisis data sederhana. Excel menyediakan fitur untuk melakukan pemrosesan data dasar, menghitung statistik sederhana, dan membuat grafik. Contohnya, Excel sering digunakan untuk perhitungan laba-rugi atau analisis tren penjualan. Sementara itu, Google Sheets, dengan kemampuan berbasis cloud, memungkinkan kolaborasi tim secara real-time dan sering digunakan untuk pelacakan data atau proyek bersama.

11.4.2 Alat Statistik

Untuk analisis yang lebih mendalam, alat statistik seperti SPSS dan SAS sering menjadi pilihan. SPSS dikenal karena kemampuannya dalam melakukan analisis statistik deskriptif maupun inferensial, dan digunakan secara luas dalam penelitian sosial atau analisis data demografi. SAS, di sisi lain, menawarkan kemampuan untuk analisis statistik, prediksi, dan manajemen data

yang lebih kompleks, sehingga sering digunakan dalam industri keuangan dan kesehatan.

11.4.3 Bahasa Pemrograman untuk Analisis Data

Bahasa pemrograman seperti Python dan R menyediakan fleksibilitas tinggi untuk manipulasi data, analisis, dan visualisasi. Python, dengan pustaka seperti pandas, NumPy, dan Matplotlib, memungkinkan pengguna untuk melakukan analisis data yang kompleks, seperti prediksi berbasis pembelajaran mesin. Python juga sering digunakan dalam industri untuk analisis data penjualan atau visualisasi data interaktif. Di sisi lain, R lebih fokus pada analisis statistik dan sangat populer di kalangan peneliti. Paket-paket seperti ggplot2 untuk visualisasi data dan dplyr untuk manipulasi data membuat R menjadi pilihan ideal untuk analisis data survei atau analisis statistik mendalam.

11.4.4 Alat Visualisasi Data

Dalam menyajikan data secara visual, alat visualisasi data seperti Tableau, Power BI, dan D3.js sangat membantu. Tableau memungkinkan pembuatan dasbor interaktif yang efektif untuk memvisualisasikan performa penjualan atau analisis KPI perusahaan. Power BI, yang terintegrasi dengan berbagai sumber data, sering digunakan untuk pelaporan keuangan dan pelacakan penjualan. Untuk kebutuhan visualisasi berbasis web, D3.js menjadi alat yang kuat karena dapat menghasilkan grafik dinamis dan interaktif.

11.4.5 Alat Basis Data

Terakhir, alat basis data memainkan peran penting dalam menyimpan dan mengelola data. SQL (*Structured Query Language*) adalah alat utama untuk mengakses dan mengelola data dalam basis data relasional, seperti melacak transaksi pelanggan. Untuk data yang tidak terstruktur, basis data NoSQL seperti MongoDB atau Cassandra menjadi pilihan karena kemampuannya menyimpan data dalam format seperti JSON, yang banyak digunakan dalam aplikasi web dan data media sosial.

11.4.6 Alat Big Data

Pada skala yang lebih besar, alat big data seperti Hadoop dan Apache Spark digunakan untuk memproses dan menganalisis data dalam jumlah besar. Hadoop memungkinkan penyimpanan dan pemrosesan data yang terdistribusi, menjadikannya ideal untuk analisis data log pada perusahaan besar. Apache Spark, dengan kemampuan pemrosesan data waktu nyata, sering digunakan untuk analisis data streaming, seperti mengidentifikasi tren pengguna secara langsung.

11.5 Aplikasi Analisis Data

Analisis data telah menjadi elemen kunci di berbagai industri, memungkinkan pengambilan keputusan berbasis data yang lebih cepat, akurat, dan efisien. Pada bagian ini, kita akan membahas penerapan analisis data di berbagai sektor, mencakup sektor kesehatan, keuangan, pemasaran, e-commerce, pendidikan, dan pemerintahan.

11.5.1 Kesehatan

Analisis data dalam sektor kesehatan mendukung pengambilan keputusan berbasis bukti untuk meningkatkan layanan pasien dan efisiensi operasional. Pemanfaatan data yang dihasilkan dari rekam medis elektronik, perangkat medis, dan data genomik memberikan wawasan yang kaya untuk diagnosis, perawatan, dan penelitian kesehatan.

1. **Prediksi Penyakit**: Model pembelajaran mesin membantu memprediksi risiko penyakit berdasarkan faktor genetik dan riwayat medis pasien (Ho *et al.*, 2019).
2. **Manajemen Rumah Sakit**: Analisis data membantu rumah sakit dalam mengoptimalkan alokasi tempat tidur dan jadwal dokter. Sebuah studi menunjukkan bahwa analisis prediktif dapat meningkatkan efisiensi operasional secara signifikan hingga 20% dengan mengurangi biaya operasional hingga 20% dan meningkatkan efisiensi proses hingga 15% (Saleh *et al.*, 2024).
3. **Pemantauan Penyakit**: Dengan analitik real-time, penyebaran penyakit menular seperti COVID-19 dapat dipetakan dan

diprediksi, memungkinkan tindakan cepat dalam pengendalian wabah (Wieczorek *et al.*, 2020).

11.5.2 Keuangan

Dalam industri keuangan, analisis data membantu mengelola risiko, mendeteksi penipuan, dan memberikan wawasan strategis untuk investasi.

1. **Penilaian Risiko Kredit**: Data nasabah, seperti riwayat pembayaran dan penghasilan, digunakan untuk mengembangkan model penilaian kredit. Algoritma berbasis machine learning meningkatkan akurasi dalam mengidentifikasi calon peminjam berisiko tinggi (Attigeri, Pai and Pai, 2017).
2. **Deteksi Penipuan**: Sistem berbasis data analitik dapat mendeteksi pola transaksi mencurigakan secara otomatis. Contohnya, bank menggunakan algoritma deteksi outlier untuk mengidentifikasi aktivitas penipuan dalam waktu nyata (Torres and Ladeira, 2020).
3. **Peramalan Pasar**: Analisis sentimen media sosial dan data historis memungkinkan peramalan harga saham yang lebih akurat (Kordonis, Symeonidis and Arampatzis, 2016).

11.5.3 Pemasaran

Analisis data telah merevolusi cara pemasaran dilakukan dengan memungkinkan pemahaman yang lebih mendalam tentang perilaku konsumen.

1. **Segmentasi Pelanggan**: Dengan analisis data demografis dan perilaku, perusahaan dapat mengelompokkan pelanggan ke dalam segmen tertentu untuk kampanye pemasaran yang lebih efektif (An *et al.*, 2018).
2. **Personalisasi**: Data riwayat pembelian dan pencarian online digunakan untuk merekomendasikan produk yang relevan, meningkatkan pengalaman pelanggan. Penelitian menunjukkan bahwa personalisasi berbasis data meningkatkan tingkat konversi hingga 25% (Choppadandi, 2023).

3. **Analisis Kampanye**: Analisis ROI dari kampanye pemasaran digital memungkinkan perusahaan untuk mengalokasikan anggaran secara lebih efisien (Almestarihi *et al.*, 2024).

11.5.4 E-commerce

Dalam e-commerce, analisis data memberikan wawasan mendalam tentang preferensi pelanggan dan mendukung pengambilan keputusan operasional.

1. **Sistem Rekomendasi**: Algoritma seperti collaborative filtering digunakan untuk merekomendasikan produk berdasarkan pembelian sebelumnya. Amazon, misalnya, menggunakan algoritma *item-to-item collaborative filtering* yang memungkinkan rekomendasi *real-time* dan dapat diskalakan untuk dataset besar. Algoritma ini membandingkan item yang dibeli dan dinilai oleh pengguna dengan item serupa untuk menghasilkan rekomendasi berkualitas tinggi (Santhosh, Cheriyan and Sindhu, 2021).
2. **Manajemen Inventaris**: Dengan menganalisis data penjualan historis menggunakan algoritma pembelajaran mesin, perusahaan dapat memprediksi permintaan produk secara akurat untuk menghindari kelebihan atau kekurangan stok (Mehmood *et al.*, 2023).
3. **Analisis Penjualan**: Data transaksi digunakan untuk mengidentifikasi pola pembelian, yang membantu perusahaan merancang strategi promosi yang efektif (Tanusondjaja, Nenycz-Thiel and Kennedy, 2016).

11.5.5 Pendidikan

Di sektor pendidikan, analisis data mendukung pengembangan sistem pembelajaran yang lebih adaptif dan efisien.

1. **Analisis Kinerja Siswa**: Dengan menggunakan algoritma jaringan saraf tiruan, kinerja siswa dalam lingkungan pembelajaran daring berhasil diprediksi secara akurat dengan tingkat akurasi sebesar 80,47%. Prediksi ini difokuskan pada data kehadiran, penggunaan kursus yang diarsipkan, dan durasi waktu yang dihabiskan untuk mengakses konten (Aydoğdu, 2019).

2. **Pengembangan Kurikulum**: Dengan menganalisis kebutuhan pasar tenaga kerja, institusi pendidikan dapat merancang kurikulum yang relevan (Januzaj *et al.*, 2024).
3. **Pembelajaran Adaptif**: Sistem pembelajaran berbasis data dapat menyesuaikan materi sesuai dengan tingkat pemahaman siswa, seperti yang diterapkan dalam platform e-learning modern (Colchester *et al.*, 2017).

11.5.6 Pemerintah

Pemerintah menggunakan analisis data untuk mendukung pengambilan kebijakan berbasis bukti dan meningkatkan efisiensi layanan publik.

1. **Pembuatan Kebijakan**: Data demografis dan ekonomi dianalisis untuk merancang kebijakan publik yang lebih efektif. Contohnya, big data dapat meningkatkan kualitas dan efektivitas layanan publik serta menggunakan sumber daya secara lebih efisien dalam merancang kebijakan publik baru (Azzone, 2018).
2. **Perencanaan Perkotaan**: Analisis data mobilitas penduduk membantu pemerintah merancang infrastruktur transportasi yang efisien (Metz *et al.*, 2024).
3. **Pengawasan Lingkungan**: Data *real-time* dari sensor digunakan untuk memonitor kualitas udara dan air, serta untuk mendeteksi perubahan lingkungan (Onay *et al.*, 2021).

11.6 Kesimpulan

Analisis data telah menjadi elemen krusial dalam pengambilan keputusan modern, baik di sektor bisnis, pemerintahan, maupun penelitian akademik. Kemampuannya untuk mengungkap pola tersembunyi, memprediksi tren, dan memberikan wawasan strategis menjadikan analisis data tidak sekadar alat bantu, melainkan fondasi utama dalam pengambilan keputusan berbasis data (*data-driven decision-making*). Bab ini telah mengupas berbagai elemen penting terkait analisis data, mulai dari jenis-jenis analisis, metodologi, alat, hingga aplikasinya dalam dunia nyata.

11.6.1 Transformasi Peran Analisis Data

Analisis data telah berkembang jauh dari sekadar alat statistik menjadi salah satu pilar utama dalam pengambilan keputusan strategis. Proses ini tidak hanya bertujuan untuk menjawab pertanyaan dasar seperti "Apa yang terjadi?" tetapi juga menyentuh level yang lebih kompleks seperti "Mengapa ini terjadi?", "Apa yang akan terjadi selanjutnya?", dan "Apa langkah terbaik yang harus diambil?"

Dengan perkembangan teknologi seperti kecerdasan buatan (*Artificial Intelligence*/AI) dan pembelajaran mesin (*Machine Learning*), analisis data kini menjadi lebih dinamis dan proaktif. Alat-alat ini memungkinkan analisis prediktif dan preskriptif yang memberikan wawasan strategis dan bahkan menyarankan tindakan yang paling optimal berdasarkan data historis dan tren saat ini.

11.6.2 Pentingnya Proses Sistematis dalam Analisis Data

Keberhasilan analisis data sangat bergantung pada metodologi yang digunakan. Setiap langkah dalam proses, mulai dari pengumpulan hingga visualisasi data, memainkan peran penting dalam memastikan hasil yang valid dan dapat diandalkan:

1. **Pengumpulan Data**: Kualitas data yang buruk dapat menghasilkan kesimpulan yang salah. Oleh karena itu, data yang relevan, akurat, dan terkini sangat penting untuk menghasilkan analisis yang bermakna.
2. **Pembersihan dan Transformasi Data**: Sebagian besar waktu dalam analisis data dihabiskan untuk membersihkan data mentah, menangani data yang hilang, atau mengatasi anomali. Proses ini menjadi dasar bagi analisis yang dapat dipercaya.
3. **Analisis dan Interpretasi**: Menggunakan teknik yang sesuai dengan jenis data dan tujuan analisis memastikan bahwa wawasan yang dihasilkan dapat langsung digunakan untuk pengambilan keputusan.

11.6.3 Pengaruh Alat dan Teknologi

Teknologi telah memainkan peran signifikan dalam mempercepat dan menyederhanakan analisis data. Alat seperti Python, R, dan Tableau telah diakui secara luas karena

kemampuannya dalam menangani berbagai kebutuhan analisis data. Selain itu, platform big data seperti Apache Spark memberikan kemampuan untuk memproses data dalam skala besar secara efisien, memungkinkan analisis *real-time* di berbagai sektor industri. Namun, keberhasilan penggunaan alat ini tidak hanya bergantung pada teknologinya, tetapi juga pada kompetensi pengguna. Oleh karena itu, pengembangan keterampilan tenaga kerja di bidang analisis data menjadi prioritas utama bagi organisasi yang ingin tetap kompetitif (Johnson *et al.*, 2021).

11.6.4 Aplikasi Nyata dan Implikasi Strategis

Dalam berbagai sektor, analisis data telah membuktikan dampaknya:

1. **Kesehatan**: Analisis data membantu dalam diagnosis penyakit, pengembangan obat baru, dan prediksi penyebaran penyakit menular.
2. **Keuangan**: Digunakan untuk mendeteksi penipuan, mengelola risiko, dan meningkatkan efisiensi operasional.
3. **E-commerce dan Marketing**: Meningkatkan pengalaman pelanggan melalui personalisasi, mengoptimalkan strategi pemasaran, dan mengelola rantai pasok.
4. **Pemerintahan**: Membantu perencanaan kota pintar, mengelola sumber daya publik, dan membuat kebijakan berbasis bukti.

Setiap aplikasi ini menunjukkan bagaimana analisis data tidak hanya membantu organisasi mencapai efisiensi tetapi juga memberikan nilai tambah bagi masyarakat secara keseluruhan.

11.6.5 Tantangan dan Masa Depan

Meskipun manfaat analisis data sangat besar, tantangan tetap ada. Beberapa di antaranya adalah:

1. **Kualitas Data**: Data yang tidak lengkap atau tidak akurat dapat merusak hasil analisis.
2. **Privasi dan Etika**: Penggunaan data pribadi harus mematuhi regulasi yang ketat untuk melindungi hak individu.

3. **Kesenjangan Keterampilan**: Tidak semua organisasi memiliki tenaga ahli yang kompeten untuk menganalisis data secara efektif.

Di masa depan, tren seperti analisis waktu nyata (*real-time analysis*), analitik prediktif berbasis AI, dan pengintegrasian teknologi *blockchain* dalam pengelolaan data diprediksi akan mendominasi. Oleh karena itu, organisasi harus bersiap untuk beradaptasi dengan perubahan ini, baik dari segi teknologi maupun sumber daya manusia.

11.6.5 Refleksi Akhir

Sebagai penutup, analisis data bukan hanya alat untuk pengambilan keputusan, tetapi juga strategi transformasional yang memungkinkan organisasi untuk menjadi lebih responsif, proaktif, dan inovatif. Dengan memanfaatkan kekuatan analisis data, organisasi tidak hanya dapat memahami masa lalu dan memprediksi masa depan tetapi juga menciptakan langkah-langkah nyata untuk membangun keberlanjutan di era digital.

DAFTAR PUSTAKA

Almestarihi, R. *et al.* (2024) 'Measuring the ROI of paid advertising campaigns in digital marketing and its effect on business profitability', *Uncertain Supply Chain Management*, p. Available at: https://doi.org/10.5267/j.uscm.2023.11.009.

An, J. *et al.* (2018) 'Customer segmentation using online platforms: isolating behavioral and demographic segments for persona creation via aggregated user data', *Social Network Analysis and Mining*, 8, p. Available at: https://doi.org/10.1007/s13278-018-0531-0.

Attigeri, G., Pai, M. and Pai, R. (2017) 'Credit Risk Assessment using Machine Learning Algorithms', *Advanced Science Letters*, 23, pp. 3649–3653. Available at: https://doi.org/10.1166/ASL.2017.9018.

Aydoğdu, Ş. (2019) 'Predicting student final performance using artificial neural networks in online learning environments', *Education and Information Technologies*, 25, pp. 1913–1927. Available at: https://doi.org/10.1007/s10639-019-10053-x.

Azzone, G. (2018) 'Big data and public policies: Opportunities and challenges', *Statistics & Probability Letters*, 136, pp. 116–120. Available at: https://doi.org/10.1016/J.SPL.2018.02.022.

Choppadandi, A. (2023) 'Enhancing Customer Experience in E-Commerce Through AI-Powered Personalization: A Case Study', *Tuijin Jishu/Journal of Propulsion Technology*, p. Available at: https://doi.org/10.52783/tjjpt.v43.i4.6018.

Colchester, K. *et al.* (2017) 'A Survey of Artificial Intelligence Techniques Employed for Adaptive Educational Systems within E-Learning Platforms', *Journal of Artificial Intelligence and Soft Computing Research*, 7, pp. 47–64. Available at: https://doi.org/10.1515/jaiscr-2017-0004.

Ho, D. *et al.* (2019) 'Machine Learning SNP Based Prediction for Precision Medicine', *Frontiers in Genetics*, 10, p. Available at: https://doi.org/10.3389/fgene.2019.00267.

Islam, M. (2020) 'Data Analysis: Types, Process, Methods, Techniques and Tools', *International Journal on Data Science*

and Technology, p. Available at: https://doi.org/10.11648/J.IJDST.20200601.12.

Januzaj, Y. *et al.* (2024) 'A Textual Content Analysis Model for Aligning Job Market Demands and University Curricula through Data Mining Techniques', *International Journal of Interactive Mobile Technologies (iJIM)*, p. Available at: https://doi.org/10.3991/ijim.v18i14.47901.

Johnson, M. *et al.* (2021) 'Impact of Big Data and Artificial Intelligence on Industry: Developing a Workforce Roadmap for a Data Driven Economy', *Global Journal of Flexible Systems Management*, 22, pp. 197–217. Available at: https://doi.org/10.1007/s40171-021-00272-y.

Kordonis, J., Symeonidis, S. and Arampatzis, A. (2016) 'Stock Price Forecasting via Sentiment Analysis on Twitter', *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, p. Available at: https://doi.org/10.1145/3003733.3003787.

L'Heureux, A. *et al.* (2017) 'Machine Learning With Big Data: Challenges and Approaches', *IEEE Access*, 5, pp. 7776–7797. Available at: https://doi.org/10.1109/ACCESS.2017.2696365.

Mehmood, U. *et al.* (2023) 'Vending Machine Product Demand Prediction Using Machine Learning Algorithms', *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6. Available at: https://doi.org/10.1109/ISNCC58260.2023.10323888.

Metz, Y. *et al.* (2024) 'Interactive Public Transport Infrastructure Analysis through Mobility Profiles: Making the Mobility Transition Transparent', *ArXiv*, abs/2407.10791, p. Available at: https://doi.org/10.48550/arXiv.2407.10791.

Onay, A. *et al.* (2021) 'Real Time Air and Water Quality Monitoring based on Distributed Sensor Network', *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 118–123. Available at: https://doi.org/10.1109/UBMK52708.2021.9558881.

Rydning, D.R.-J.G.-J., Reinsel, J. and Gantz, J. (2018) 'The digitization of the world from edge to core', *Framingham: International Data Corporation*, 16, pp. 1–28.

Safitri, S.T. *et al.* (2024) *Sistem Pendukung Keputusan*. wawasan Ilmu.

Saleh, H.H. *et al.* (2024) 'Enhancing Business Operations Efficiency thorough Predictive Analytics', *Journal of Ecohumanism*, p. Available at: https://doi.org/10.62754/joe.v3i5.3932.

Santhosh, N., Cheriyan, J. and Sindhu, M. (2021) 'An Intelligent Exploratory Approach for Product Recommendation Using Collaborative Filtering', *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pp. 232–237. Available at: https://doi.org/10.1109/ACCESS51619.2021.9563330.

Shardt, Y. (2022) 'Introduction to Statistics and Data Visualisation', *Statistics for Chemical and Process Engineers*, p. Available at: https://doi.org/10.1007/978-3-319-21509-9_1.

Tanusondjaja, A., Nenycz-Thiel, M. and Kennedy, R. (2016) 'Understanding Shopper Transaction Data', *International Journal of Market Research*, 58, pp. 401–419. Available at: https://doi.org/10.2501/IJMR-2016-026.

Torres, R.A.L. and Ladeira, M. (2020) 'A proposal for online analysis and identification of fraudulent financial transactions', *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 240–245. Available at: https://doi.org/10.1109/ICMLA51294.2020.00047.

Wieczorek, M. *et al.* (2020) 'Real-time neural network based predictor for cov19 virus spread', *PLoS ONE*, 15, p. Available at: https://doi.org/10.1371/journal.pone.0243189.

BAB 12
INFORMATION RETRIEVAL

Oleh Tri Kustanti Rahayu

12.1 Pendahuluan

Pada era digital ini, jumlah informasi yang tersedia terus meningkat secara eksponensial. Dari teks, gambar, hingga video, data terus-menerus dihasilkan melalui aktivitas manusia dan teknologi. Namun, data yang melimpah ini tidak akan bernilai jika tidak dapat diakses atau digunakan secara efisien. Di sinilah Information Retrieval (IR) memainkan peran penting. IR adalah teknologi yang memungkinkan kita untuk menemukan informasi yang relevan dari kumpulan data yang besar dan kompleks.

Sebagai bagian penutup dalam buku ini, sub bab ini dirancang untuk memberikan wawasan mendalam tentang bagaimana IR berfungsi dan mengapa teknologi ini sangat penting dalam pengelolaan informasi di dunia modern. Pembaca akan diajak untuk memahami IR sebagai teknologi yang melampaui fungsi database tradisional, yang umumnya berfokus pada pengelolaan data terstruktur.

Mengapa Information Retrieval Penting?

Seiring perkembangan teknologi, jenis data yang perlu dikelola tidak lagi terbatas pada angka atau teks dalam tabel yang terorganisir. Sebagian besar data saat ini tidak terstruktur, seperti dokumen teks, gambar, video, atau rekaman suara. IR dirancang untuk menangani data seperti ini, berbeda dengan database relasional yang lebih cocok untuk data terstruktur.

Bayangkan ketika Anda mencari artikel tertentu di perpustakaan digital atau menemukan produk di e-commerce. Anda tidak perlu tahu lokasi spesifik data tersebut; cukup masukkan kata kunci, dan teknologi IR akan membantu Anda menemukan informasi yang sesuai. Dalam konteks ini, IR

menjadi jembatan antara kebutuhan pengguna dan data yang ada.

Perkembangan Sistem Pencarian di Era Internet hingga Saat Ini

Ketika internet mulai berkembang pada akhir 1990-an, kebutuhan untuk mengelola dan menemukan informasi dari miliaran halaman web menjadi tantangan besar. Sistem pencarian tradisional yang dirancang untuk koleksi terbatas di perpustakaan digital tidak lagi memadai untuk menangani skala dan kompleksitas data web [1].

Pada tahap awal internet, direktori web seperti Yahoo! digunakan untuk mengatur dan menemukan halaman web secara manual. Namun, pendekatan ini segera digantikan oleh mesin pencari berbasis algoritma yang lebih efisien. Salah satu tonggak sejarah penting adalah pengenalan algoritma **PageRank** oleh Sergey Brin dan Larry Page (1998). Algoritma ini tidak hanya mengevaluasi relevansi konten berdasarkan kata kunci tetapi juga memperhitungkan jumlah dan kualitas tautan yang menuju ke halaman tertentu [2]. PageRank menjadi inti dari Google, yang dengan cepat mendominasi pasar mesin pencari.

Kemajuan di Era Web 2.0

Dengan munculnya Web 2.0 pada awal 2000-an, IR menghadapi tantangan baru, seperti penanganan data yang dihasilkan pengguna, termasuk blog, media sosial, dan ulasan. Mesin pencari mulai mengintegrasikan pendekatan baru untuk menangani jenis data ini:

1. **Pengindeksan Real-Time:** Pencarian informasi pada media sosial seperti Twitter dan Facebook memerlukan algoritma yang mampu memproses data secara real-time [1].
2. **Pemrosesan Bahasa Alami (NLP):** Teknologi NLP mulai digunakan untuk memahami query yang lebih kompleks, termasuk pertanyaan berbentuk percakapan [3].

Perkembangan Teknologi di Era Big Data dan AI

Di era *Big Data*, IR berkembang pesat dengan adopsi teknologi pembelajaran mesin dan kecerdasan buatan. Mesin pencari modern seperti Google dan Bing menggunakan *machine learning* untuk memahami konteks query dan preferensi pengguna. Berikut beberapa perkembangan signifikan:

1. **RankBrain:** Diperkenalkan oleh Google pada tahun 2015, RankBrain adalah algoritma berbasis *machine learning* yang dirancang untuk memahami maksud pencarian yang kompleks dan memberikan hasil yang lebih relevan [3].
2. **Model Transformer:** Model seperti BERT (Bidirectional Encoder Representations from Transformers) digunakan untuk memahami konteks query dengan lebih baik, memungkinkan hasil pencarian yang lebih akurat bahkan untuk query dengan struktur kalimat yang rumit [3]
3. **Pencarian Multimedia:** Dengan meningkatnya penggunaan video dan gambar, sistem IR kini mencakup pengindeksan dan pencarian data multimedia, seperti gambar berbasis metadata atau video menggunakan algoritma pengenalan visual [4].

Tren Terkini dalam Information Retrieval

1. **Pencarian Berbasis Suara dan Konversasional:** Sistem seperti Siri, Alexa, dan Google Assistant mengubah cara pengguna berinteraksi dengan mesin pencari. Pengguna dapat mengajukan pertanyaan melalui suara, dan IR bekerja untuk memberikan jawaban yang relevan dalam bentuk percakapan [1].
2. **Pencarian Personal:** Mesin pencari saat ini memanfaatkan data pengguna, seperti riwayat pencarian dan lokasi, untuk memberikan hasil yang lebih dipersonalisasi [2].
3. **Pencarian Semantik:** Dengan pendekatan pencarian semantik, IR tidak hanya mencari kata kunci tetapi juga memahami makna dan hubungan antar konsep dalam query [5].

Perkembangan IR sejak era katalog fisik hingga teknologi berbasis kecerdasan buatan mencerminkan bagaimana kebutuhan pencarian informasi telah berubah secara signifikan. Dari Boolean

Model hingga algoritma canggih seperti BERT, IR terus berkembang untuk menghadapi tantangan dari dunia data yang terus meluas. Seiring berjalannya waktu, IR tidak hanya menjadi alat pencarian tetapi juga komponen penting dalam eksplorasi dan pengelolaan informasi di era digital.

Hubungan IR dengan Database

Meskipun IR lebih sering diasosiasikan dengan data tidak terstruktur, teknologi ini tidak sepenuhnya terpisah dari database. Sebaliknya, IR sering bergantung pada database untuk menyimpan dan mengelola koleksi data besar. Database menyediakan kerangka kerja untuk menyimpan informasi, sedangkan IR bertugas mencari dan menyajikan data yang relevan berdasarkan permintaan pengguna.

Information Retrieval (IR) dan database adalah dua bidang teknologi yang sering dianggap terpisah, tetapi sebenarnya saling melengkapi. Perbedaan mendasar terletak pada jenis data yang ditangani dan metode pencarian yang digunakan. Database berfokus pada data terstruktur, seperti tabel relasional dengan atribut yang jelas, dan menggunakan kueri berbasis logika formal seperti SQL untuk menghasilkan hasil yang pasti. Sebaliknya, IR menangani data tidak terstruktur atau semi-terstruktur, seperti dokumen teks, gambar, atau video, menggunakan algoritma berbasis relevansi, seperti TF-IDF atau model probabilistik, untuk menyajikan hasil pencarian yang diranking berdasarkan relevansi. Selain itu, tujuan utama database adalah manajemen transaksi dan penyimpanan data dengan konsistensi tinggi, sementara IR berfokus pada menemukan dan menyajikan informasi yang relevan dari koleksi data besar.

Hubungan antara IR dan database dapat diibaratkan sebagai mesin dan bahan bakarnya: database menyimpan data terorganisasi, sementara IR mengakses dan memanfaatkan data tersebut secara efektif, terutama untuk data tidak terstruktur. Dalam aplikasi modern seperti mesin pencari atau e-commerce, database menyimpan informasi dasar, seperti stok dan harga produk, sedangkan IR memproses query pengguna untuk

mencocokkannya dengan deskripsi atau ulasan produk yang tidak terstruktur. Kolaborasi ini menjadi fondasi berbagai teknologi yang kita gunakan saat ini.

12.2 Dasar-Dasar *Information Retrieval*

Information Retrieval (IR) merupakan teknologi yang mendasari banyak sistem pencarian modern, mulai dari mesin pencari internet hingga perpustakaan digital. IR bertujuan untuk menemukan informasi yang relevan berdasarkan kebutuhan pengguna dari kumpulan data yang sangat besar. Dalam sub bab ini, kita akan mengeksplorasi definisi IR serta komponen-komponen utama yang menjadi fondasinya.

12.3 Definisi Information Retrieval

Menurut Manning, Raghavan, dan Schütze (2008) dalam buku *Introduction to Information Retrieval*, IR adalah proses menemukan dokumen yang relevan untuk memenuhi kebutuhan informasi pengguna dari koleksi dokumen yang besar. IR berfokus pada pengelolaan data tidak terstruktur, seperti teks, gambar, atau video, berbeda dengan database tradisional yang menangani data terstruktur seperti angka dalam table [1]. IR dirancang untuk memenuhi kebutuhan pengguna melalui pencarian, penyaringan, dan peringkat informasi berdasarkan relevansi. Dalam buku ini, mereka menyoroti pentingnya algoritma dalam memberikan hasil yang relevan dengan cepat dan akurat di tengah pertumbuhan data yang eksponensial. IR telah menjadi fondasi penting dalam berbagai aplikasi modern, termasuk mesin pencari, sistem rekomendasi, dan *chatbot*.

Dalam pengertian lain tentang IR, IBM mendefinisikan IR sebagai teknologi yang membantu pengguna mengakses informasi yang relevan, bahkan ketika data tersedia dalam format yang beragam, seperti teks, gambar, atau video. IR tidak hanya mencakup pengambilan data yang sesuai dengan query, tetapi juga pemahaman terhadap konteks query itu sendiri, sehingga mampu memberikan jawaban yang relevan dalam berbagai skenario penggunaan [6].

12.4 Pencarian pada Data Terstruktur

Data terstruktur merujuk pada data yang terorganisasi dalam format tabel atau hierarki yang jelas, seperti database relasional. Data ini terdiri dari elemen-elemen yang mudah diidentifikasi, seperti nama, tanggal, atau nilai numerik. Pencarian pada data terstruktur dilakukan menggunakan bahasa query seperti SQL (*Structured Query Language*). Misalnya:

```
SELECT * FROM pelanggan WHERE kota = 'Merauke';
```

Query ini akan mengembalikan hasil yang spesifik dan presisi, karena atribut dan relasi antar data sudah didefinisikan dengan jelas.

Keunggulan pencarian pada data terstruktur adalah akurasi tinggi, performa yang cepat, dan kemudahan integrasi dengan sistem lain [7]. Namun, pendekatan ini terbatas pada data dengan pola terorganisir, sehingga kurang efektif untuk data tidak terstruktur.

12.5 Pencarian pada Data Tidak Terstruktur

Berbeda dengan data terstruktur, data tidak terstruktur mencakup informasi tanpa format tetap, seperti dokumen teks, e-mail, gambar, video, atau audio. Pencarian pada data ini memerlukan pendekatan IR dengan teknik khusus seperti pengindeksan teks, algoritma berbasis statistik, atau pembelajaran mesin.

Contohnya, ketika Anda mengetik "tips belajar Python" di Google, sistem IR akan memproses query tersebut dengan memindai dokumen relevan di seluruh web, menganalisis kata kunci, dan memberikan peringkat hasil pencarian berdasarkan relevansi. Sistem ini tidak hanya mencari berdasarkan kata yang cocok, tetapi juga mempertimbangkan konteks query.

Pada data tidak terstruktur membutuhkan pendekatan berbeda karena sifat data yang kompleks dan tidak memiliki atribut tetap seperti pada data terstruktur [1].

Berikut adalah perbedaan utama antara data terstruktur dan tidak terstruktur.

Tabel 12.1. Perbedaan utama antara data terstruktur dan tidak terstruktur

Aspek	Data Terstruktur	Data Tidak Terstruktur
Definisi	Data yang terorganisasi dalam format tabel atau hirarki yang jelas.	Data yang tidak memiliki format atau struktur yang tetap.
Format	Tabel, database relational, atau spreadsheet.	Dokumen teks, gambar, video, audio, atau e-mail.
Contoh	Nama, tanggal, angka, ID Pelanggan.	Artikel berita, postingan media sosial, file multimedia.
Pencarian	Menggunakan bahasa query sperti SQL.	Menggunakan teknik Information retrieval (IR), seperti pengindeksan teks atau pembelajaran mesin.
Kemudahan Analisis	Mudah dianalisis menggunakan alat database tradisional.	Memerlukan teknik khusus seperti Natural Language Processing (NLP) untuk analisis.
Relevansi	Cocok untuk data dengan pola atribut terdefinisi dengan jelas.	Cocok untuk menangani data yang tidak memiliki struktur tetap.
Kecepatan Pencarian	Cepat karena data sudah terorganisasi dengan baik.	Lebih lambat karena membutuhkan pengolahan data yang kompleks.

12.6 Tujuan dan Fungsi *Information Retrieval* (IR)

Information Retrieval (IR) memiliki tujuan utama untuk membantu pengguna menemukan informasi yang relevan dari kumpulan data yang besar dan kompleks. Dengan IR, pengguna tidak perlu mengetahui secara persis lokasi data yang mereka cari; cukup dengan memberikan query atau kata kunci, sistem IR akan menyaring dan menampilkan informasi yang sesuai.

12.6.1 Tujuan *Information Retrieval*

Adapun tujuan dikembangkannya Information Retrieval antara lain:

1. Menyediakan Informasi yang Relevan; IR dirancang untuk memprioritaskan informasi yang paling relevan berdasarkan kebutuhan pengguna, sering kali melalui proses peringkat. Relevansi menjadi kunci utama dalam menentukan keberhasilan sistem IR [1].
2. Mengelola Data Tidak Terstruktur; IR fokus pada data tidak terstruktur, seperti teks, gambar, atau video, yang sulit diakses dengan metode pengelolaan database tradisional.
3. Meningkatkan Aksesibilitas Informasi; Dengan IR, informasi dapat diakses lebih cepat dan efisien, bahkan dari koleksi dokumen yang sangat besar.

12.6.2 Fungsi *Information Retrieval*

Adapun fungsi yang ingin dicapai melalui penerapan IR ini antara lain:

1. Pencarian Informasi; Fungsi utama IR adalah membantu pengguna mencari informasi yang mereka butuhkan. Pencarian ini melibatkan proses pencocokan antara query pengguna dan data dalam koleksi.
2. Pengelompokan dan Pengorganisasian; IR juga membantu mengelompokkan informasi berdasarkan tema atau kategori tertentu, mempermudah pengguna menemukan pola atau tren dalam data.
3. Penarikan Kesimpulan dari Data; Selain pencarian, IR dapat digunakan untuk mengekstrak wawasan dari data besar, seperti analisis teks untuk memahami sentimen dalam ulasan pelanggan.

12.7 Komponen Utama dalam Sistem IR

Sistem *Information Retrieval* (IR) bekerja melalui beberapa komponen utama yang saling terintegrasi untuk menyediakan informasi yang relevan sesuai kebutuhan pengguna. Berikut adalah penjelasan tentang komponen-komponen tersebut:

1. **Pengindeksan Dokumen.**
   Indexing dalam IR merupakan proses penting yang bertujuan untuk menyusun representasi data dalam format yang dapat diproses dengan mudah oleh mesin pencari. Dalam konteks ini, *indexing* membantu memetakan dokumen ke dalam format yang memungkinkan sistem IR memahami relevansi antara query pengguna dan dokumen dalam database. Selain itu, pengindeksan dokumen tidak hanya tentang pencatatan istilah, tetapi juga penghitungan frekuensi kemunculan istilah dalam dokumen untuk menentukan bobotnya dalam konteks tertentu.
   Salah satu metode populer dalam proses *indexing* adalah *Term Frequency-Inverse Document Frequency* (TF-IDF).
2. **Pemrosesan Query**
   Pemrosesan query adalah di mana sistem berusaha memahami maksud pencarian pengguna untuk mencocokkannya dengan data yang relevan. Proses ini melibatkan analisis query yang dimasukkan pengguna dan, jika diperlukan, melakukan transformasi atau perluasan query untuk meningkatkan akurasi hasil pencarian.
   Langkah pertama dalam pemrosesan query adalah *tokenization*, di mana query dipecah menjadi elemen-elemen kecil, seperti kata-kata individu. Setelah itu, sistem dapat menerapkan *stemming* atau *lemmatization* untuk mengidentifikasi bentuk dasar dari kata-kata tersebut, misalnya mengubah "mobil-mobil" menjadi "mobil".
   Selanjutnya, sistem IR sering menerapkan *query expansion* untuk memperkaya konteks query. Dalam kasus pencarian dengan kata "mobil", sistem dapat secara otomatis menambahkan sinonim atau kata yang berkaitan, seperti "kendaraan" atau "otomotif". Pendekatan ini didukung oleh penggunaan *thesaurus* atau model berbasis semantik seperti WordNet, untuk memperluas cakupan pencarian tanpa mengabaikan relevansi.
   Pemrosesan query ini merupakan kunci untuk meningkatkan presisi dan cakupan hasil pencarian, dengan mempertimbangkan relevansi semantik dan struktur

dokumen (Manning et al, 2008). Selain itu penelitian lain juga menekankan pentingnya teknik *natural language processing* (NLP) dalam memahami maksud query yang kompleks, terutama dalam pencarian berbasis teks bebas [8].

3. Fungsi Pencocokan

Fungsi pencocokan bertujuan untuk menghubungkan query pengguna dengan dokumen yang relevan berdasarkan teknik-teknik tertentu. Proses pencocokan tidak hanya bergantung pada kesesuaian kata kunci, tetapi juga mempertimbangkan relevansi semantik, struktur dokumen, dan konteks query. Tiga model pencocokan yang umum digunakan dalam IR adalah sebagai berikut:

a. Model Boolean

Model Boolean menggunakan logika sederhana berbasis operator seperti AND, OR, dan NOT untuk menyaring dokumen. Dalam model ini, query dipecah menjadi serangkaian kondisi yang harus dipenuhi agar sebuah dokumen dianggap relevan. Misalnya, pencarian "kamera AND digital" hanya akan mengembalikan dokumen yang mengandung kedua istilah tersebut secara bersamaan. Meskipun model ini mudah dipahami, kelemahannya terletak pada hasil pencarian yang kaku karena tidak ada tingkat relevansi yang dihitung [9].

b. *Vector Space Model* (VSM)

VSM merepresentasikan dokumen dan query dalam bentuk vektor dalam ruang multidimensi. Setiap dimensi mewakili sebuah istilah, dengan bobot yang ditentukan oleh metode seperti *Term Frequency-Inverse Document Frequency* (TF-IDF). Proses pencocokan dilakukan dengan menghitung kesamaan antara vektor query dan vektor dokumen menggunakan ukuran seperti *cosine similarity*. Dokumen dengan nilai kesamaan tertinggi dianggap paling relevan. Keunggulan model ini adalah kemampuannya menangani relevansi bertingkat, sehingga memberikan

hasil yang lebih fleksibel dibandingkan model Boolean [1].

4. **Model Probabilistik**
   Model ini menggunakan pendekatan berbasis peluang statistik untuk menilai relevansi dokumen terhadap query. Konsep dasarnya adalah menghitung probabilitas bahwa sebuah dokumen relevan, berdasarkan distribusi istilah dalam dokumen dan query. Model ini juga memperhitungkan umpan balik relevansi dari pengguna, sehingga semakin lama digunakan, sistem dapat meningkatkan akurasi pencocokan. Salah satu implementasi populernya adalah model BM25, yang sangat efektif untuk berbagai aplikasi IR modern [10].
5. Perangkingan
   Setelah dokumen yang relevan ditemukan, sistem IR akan mengurutkan hasil pencarian berdasarkan tingkat relevansi. Proses ini memastikan bahwa pengguna menerima dokumen yang paling sesuai di bagian atas daftar hasil pencarian. Algoritma seperti *PageRank* dan model pembelajaran mesin sering digunakan untuk menentukan peringkat ini. Proses ini memastikan bahwa dokumen yang paling relevan dengan query pengguna ditempatkan di urutan teratas, meningkatkan efisiensi dan kenyamanan pencarian [2].

12.8 Kesimpulan

Perkembangan *Information Retrieval* (IR) dari era katalog fisik hingga teknologi berbasis kecerdasan buatan menunjukkan transformasi besar dalam cara manusia mengakses dan mengelola informasi. Awalnya, IR hanya berfungsi sebagai alat bantu sederhana untuk menemukan dokumen berdasarkan kata kunci melalui pendekatan seperti *Boolean Model*. Namun, seiring dengan meningkatnya kompleksitas dan volume data, IR berkembang menjadi sistem yang lebih canggih dengan dukungan algoritma berbasis pembelajaran mesin dan pemrosesan bahasa alami (NLP), seperti BERT.

Kemajuan ini tidak hanya memengaruhi kecepatan dan akurasi pencarian informasi, tetapi juga mengubah IR menjadi elemen strategis yang mendukung berbagai sektor, seperti pendidikan, bisnis, dan penelitian. IR kini mampu memahami konteks pencarian yang lebih kompleks, menyediakan hasil yang relevan secara semantik, dan mendukung pengambilan keputusan berbasis data.

Dengan kemajuan teknologi seperti kecerdasan buatan, IR terus berkembang dari sekadar alat pencarian informasi menjadi komponen integral dalam eksplorasi, analisis, dan pengelolaan data di era digital. Transformasi ini mencerminkan adaptasi IR terhadap tantangan dunia modern yang kaya data, sekaligus menjadi bukti pentingnya IR dalam mendukung kebutuhan informasi manusia yang semakin dinamis dan personal.

DAFTAR PUSTAKA

C. D. Manning, P. Raghavan, and H. Schutze, *An Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press, 2009. [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

S. Brin and P. Lawrence, "The anatomy of a large-scale hypertextual Web search engine," *Comput. Networks ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998.

J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

W. B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*. Addison-Wesley, 2010.

G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing," *Commun. ACM*, 1975, doi: 10.1145/361219.361220.

IBM, "What is Information Retrieval?," IBM Research Publications. [Online]. Available: https://www.ibm.com/

S. B. Ramez, Elmasri. Navathe, *Fundamentals of Database System (7th ed.)*. Pearson, 2015.

A. W, A. K, and S. A, "Enhanced Query Processing in IR Systems Using NLP Techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 3, pp. 45–52, 2020.

B.-Y. Ricardo and R.-N. Berthier, *Modern Information Retrieval*. Addison-Wesley, 1999.

K. Sparck Jones, S. Walker, and S. E. Robertson, "Probabilistic model of information retrieval: Development and comparative experiments. Part 2," *Inf. Process. Manag.*, 2000, doi: 10.1016/S0306-4573(00)00016-9.

BIODATA PENULIS

Ahmad Jurnaidi Wahidin, M.Kom.
Dosen Program Studi Teknologi Informasi
Universitas Bina Sarana Informatika

Penulis lahir di Kalibata, Lampung, dan tumbuh besar di daerah tersebut. Setelah menyelesaikan pendidikan formal, penulis memulai karier di industri penerbangan yang memberikan wawasan luas dalam dunia profesional. Selanjutnya, penulis melanjutkan pendidikan ke jenjang sarjana dan berhasil menyelesaikan program magister pada tahun 2018. Minat yang besar terhadap dunia literasi membawa penulis menekuni bidang penulisan, menghasilkan karya dalam bentuk book chapter dan jurnal ilmiah yang mencerminkan dedikasi terhadap pengembangan ilmu pengetahuan.

BIODATA PENULIS

Nisa Miftachurohmah, S.Kom., M.Si.
Fakultas Teknologi Informasi Universitas Sembilanbelas November Kolaka

Penulis lahir di Malang tanggal 24 April 1989. Penulis adalah dosen Program Studi Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Sembilanbelas November Kolaka. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika di Universitas Islam Negeri Maulana Malik Ibrahim Malang dan melanjutkan S2 pada Jurusan Matematika, Bidang Komputasi di Institut Teknologi Sepuluh Nopember Kolaka. Penulis menekuni bidang Rekayasa Perangkat Lunak sebagai minat riset.

BIODATA PENULIS

Daniel Alfa Puryono, S.Kom., M.Kom
Dosen Program Studi S-1 Sistem Informasi - STMIK AKI Pati

Lahir di kaki gunung Sapto Renggo Jepara tahun 1980. Lulus Sarjana Komputer, Jurusan Sistem Informasi dari UNAKI Semarang dan memperoleh gelar Magister Komputer dari Universitas Diponegoro Semarang. Pernah menjadi guru TIK SMP Negeri 4 Kembang Jepara selama 3 tahun dan guru TKJ di SMK Gajah Mada Pati selama 3 tahun. Kemudian pada tahun 2007 hingga saat ini menjadi Dosen di STMIK AKI Pati. Serta Dosen di Fakultas Sain dan Teknologi Universitas Terbuka. Selain sebagai dosen penulis juga aktif melakukan peneliti, beberapa kali mendapatkan hibah penelitian dan pengabdian masyarakat dari Kemendikbud. Hasil karya penelitian tersebut sampai saat ini ada 31 artikel ilmiah, 5 Hak cipta dan 2 buku yang ter-index di SINTA. Selain itu penulis juga aktif sebagai reviewer di beberapa jurnal penelitian maupun jurnal pengabdian kepada masyarakat. Hingga saat ini penulis juga masih aktif sebagai Asseor uji kompetensi keahlian siswa SMK jurusan TKJ. Pada tahun 2017-2022 penulis juga dipercaya oleh pemerintah Kabupaten Pati menjadi anggota Dewan Riset Daerah (DRD). Selain itu pada tahun 2018 juga dipercaya menjadi Ketua Dewan Pakar KRENOVA Pati Inovation Award dan Dewan Pakar Inovasi Daerah hingga saat ini.

BIODATA PENULIS

Hanes, S.Kom., M.Kom.
Dosen Program Studi Sistem Informasi
Fakultas Informatika Universitas Mikroskil

Penulis lahir di Medan tanggal 02 Mei 1991. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Informatika, Universitas Mikroskil. Menyelesaikan pendidikan S1 pada Jurusan Sistem Informasi dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis menekuni bidang Sistem Informasi dan sukan mengeksplorasi hal-hal baru.

BIODATA PENULIS

Rajif Agung Yunmar, S.Kom., M.Cs.
Dosen Program Studi Teknik Informatika
Fakultas Teknologi Industri Institut Teknologi Sumatera

Penulis adalah dosen di Progran Studi Teknik Informatika, Institut Teknologi Sumatera (ITERA), dengan fokus penelitian pada keamanan siber dan keamanan *mobile*. Penelitiannya menitikberatkan pada identifikasi serta mitigasi ancaman keamanan terhadap perangkat *mobile* dan jaringan komputer. Penulis juga memiliki berbagai sertifikasi keamanan siber yang diakui secara luas di dunia industri, diantaranya: Certified Ethical Hacker (CEH), Huawei HCIA-Security, Certified Incident Handler (ECIH), dan Cisco CCST Cybersecurity.

BIODATA PENULIS

Salsalina Br Sembiring, S.Kom., M.TI.
Dosen Program Studi Sistem Informasi
Fakultas Informatika Universitas Mikroskil

Penulis lahir di Bandar Meriah tanggal 2 Desember 1986. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Informatika, Universitas Mikroskil. Menyelesaikan pendidikan S1 pada Program Studi Sistem Informasi di Universtas Mikroskil dan melanjutkan S2 pada Program Studi Teknik Informatika Universitas Bina Nusantara.

BIODATA PENULIS

Hasanudin Jayawardana, S.T., MMSI.
Dosen Jurusan Sistem Informasi
Fakultas Teknik Universitas Musamus

Penulis lahir di Merauke pada tanggal 11 Mei tahun 1980. Menempuh pendidikan Sarjana (S1) pada Program Studi Teknik Elektro, kosentrasi Elektronika, Institut Teknologi Nasional (Malang) dan menyelesaiannya pada tahun 2005, sempat menjadi tenaga pengajar di salah satu Sekolah Teknik Menengah di Merauke. Kemudian melanjutkan ke jenjang Strata-2 (S2) pada tahun 2008 di Universitas Gunadarma (Jakarta) pada bidang Manajemen Sistem Informasi, dan berhasil memperoleh gelar Magister pada tahun 2011.

Saat ini, penulis menjadi Dosen Tetap di Universitas Musamus Merauke pada jurusan Sistem Informasi, dengan bidang keahlian Manajemen dan Strategi pengembangan Sistem Informasi. Penulis memiliki pengalaman mengajar pada beberapa mata kuliah, seperti Analisis dan Desain Sistem Informasi, Sistem Informasi Manajemen, Pengantar Teknologi Informasi, Sistem Basis Data, Sistem Operasi serta Sistem Pendukung Keputusan.

BIODATA PENULIS

Ir. Ridho Surya Kusuma, S.T., M.Kom.
Dosen Program Studi PJJ Informatika
Fakultas Teknologi dan Ilmu Kesehatan
Universitas Siber Muhammadiyah

Penulis lahir di Pekanbaru tanggal 08 Desember 1996. Penulis adalah dosen tetap pada Program Studi PJJ Informatika Fakultas Fakultas Teknologi dan Ilmu Kesehatan, Universitas Siber Muhammadiyah. Penulis Menyelesaikan pendidikan S1 pada Jurusan Teknik Elektro, melanjutkan S2 pada Jurusan Magister Teknik Informatika, memperkuat bidang keteknikan melalui profesi insinyur, dan sedang melanjutkan S3 untuk gelar "Doctor of Philosophy" bidang spesialis Malware Protection di Malaysia. Penulis memiliki pengalaman luas dalam penelitian, layanan masyarakat, dan penulisan publikasi ilmiah baik di tingkat nasional maupun internasional, bidang minat penelitian meliputi pembelajaran mesin, jaringan komputer, analisis malware, kecerdasan buatan dalam keamanan siber, forensik digital, blockchain, robotika, dan teori informasi.

BIODATA PENULIS

Hilman Nuril Hadi, S.Kom., M.Kom.
Dosen Program Studi Informatika
STIKI Malang

Penulis lahir di Situbondo pada bulan Juli tahun 1992. Saat ini penulis merupakan dosen tetap Program Studi Informatika STIKI Malang. Penulis menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika di STIKI Malang pada tahun 2015. Selanjutnya penulis melanjutkan program S2 pada Jurusan Ilmu Komputer di Universitas Brawijaya dan lulus di tahun 2019. Pengajaran dan penelitian penulis berfokus pada bidang Rekayasa Perangkat Lunak.

BIODATA PENULIS

Irpan Adiputra Pardosi, S.Kom., M.TI.
Dosen Program Studi Teknik Informatika
Fakultas Informatika Universitas Mikroskil

Penulis lahir di Parsoburan tanggal 16 Mei 1986. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Informatika, Universitas Mikroskil. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Magister Teknik Infrormatika. Penulis menekuni bidang penelitian pengolahan citra, kecerdasan buatan dan pengembangan aplikasi.

BIODATA PENULIS

Ir. Jarot Budiasto, S.T., M.T.
Dosen Program Studi Sistem Informasi
Fakultas Teknik Universitas Musamus

Penulis lahir di Magelang pada 4 Maret 1981. Menyelesaikan pendidikan Sarjana (S1) di Program Studi Teknik Informatika, Universitas Atma Jaya Yogyakarta, pada tahun 2006, dan melanjutkan pendidikan Magister (S2) di bidang Teknik Informatika di universitas yang sama, lulus pada tahun 2008. Pada tahun 2022, penulis memperoleh gelar Insinyur dari Universitas Muslim Indonesia.

Sejak tahun 2008, penulis aktif berkarier sebagai dosen di Program Studi Sistem Informasi, Universitas Musamus, dengan keahlian utama di bidang *Data Analytics* dan *Machine Learning*. Selain berkecimpung di dunia akademik, penulis juga aktif menulis dan telah menerbitkan beberapa buku, di antaranya Sistem Pendukung Keputusan dan Belajar Augmented Reality Dasar pada Pemrograman Mobile dengan Unity 3D.

Melalui karya dan dedikasinya, penulis berkomitmen untuk terus berkontribusi dan berupaya menciptakan dampak positif yang signifikan dalam pengembangan ilmu pengetahuan dan teknologi di Indonesia.

BIODATA PENULIS

Tri Kustanti Rahayu, S.Kom., M.Kom.
Dosen Program Studi Sistem Informasi
Fakultas Teknik Universitas Musamus Merauke

Tri Kustanti Rahayu, lahir di Mojokerto pada tahun 1982, adalah seorang dosen dan penulis yang memiliki minat besar pada dunia teknologi dan literasi. Ia menyelesaikan pendidikan Sarjana (S1) di Universitas Dr. Soetomo Surabaya pada tahun 2005, kemudian melanjutkan studi Magister (S2) di Universitas Diponegoro, Semarang, dan meraih gelar pada tahun 2018. Hobinya menulis dan traveling menjadi bagian penting dalam hidupnya, memberikan inspirasi dalam setiap karya dan penelitiannya. Bidang keilmuan yang menjadi favoritenya adalah Natural Language Processing (NLP) dan Computer Vision, yang mencerminkan dedikasinya untuk mengembangkan inovasi berbasis teknologi demi kemajuan ilmu pengetahuan.