BASIS DATA 2

Penulis :
Meri Azmi, Irmawati, Yulita Salim, Budi Wijaya Rauf,
Mutia Rahmi Dewi, Lutfi Syafirullah

ISBN 978-623-125-246-3

BASIS DATA 2

Meri Azmi
Irmawati
Yulita Salim
Budi Wijaya Rauf
Mutia Rahmi Dewi
Lutfi Syafirullah

GET PRESS INDONESIA

BASIS DATA 2

Penulis :
Meri Azmi
Irmawati
Yulita Salim
Budi Wijaya Rauf
Mutia Rahmi Dewi
Lutfi Syafirullah

ISBN : 978-623-125-246-3

Editor : Yuliatri Novita., M.Hum
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, Juni 2024

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 Basis Data 2 ini.

Buku Ini Membahas Pengenalan Basis Data, *Database Management System* (DBMS), Basis Data Relasional, Dependensi dan Normalisasi, Data *Definition Language* Dan Data *Manipulation Language,* Studi Kasus Basis Data Penjualan.

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, Juni 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENGENALAN BASIS DATA

Oleh Meri Azmi

1.1 Pendahuluan

Manajemen informasi menjadi kunci sukses dalam perkembangan organisasi pada dunia bisnis saat ini. Salah satu fondasi utama yang mendukung adalah sistem basis data.

1.1.1 Latar Belakang

Pada awalnya, organisasi biasanya menggunakan metode manual atau berbasis berkas untuk menyimpan data, yang mungkin memadai untuk mengelola jumlah data yang relatif kecil. Namun, seiring dengan pesatnya pertumbuhan data, metode ini mulai menunjukkan keterbatasannya. Jumlah data yang semakin besar menjadi tidak efisien dan sulit untuk diatur.

Munculnya sistem basis data memberikan solusi besar bagi organisasi ketika menghadapi masalah ini. Sistem ini memungkinkan organisasi untuk menyimpan, mengelola, dan mengakses data dengan cara yang lebih terstruktur dan efisien. Sebagai tanggapan terhadap peningkatan kompleksitas data, basis data memberikan dasar yang kuat untuk manajemen data yang lebih canggih (Coronel and Morris, 2019).

Dengan menggunakan sistem basis data, organisasi dapat mengatasi masalah yang muncul sebagai akibat dari pertumbuhan eksponensial data. Struktur terorganisir basis data memudahkan pencarian dan analisis data, yang memungkinkan penggunaan data secara lebih efektif. Munculnya sistem basis data menunjukkan transformasi besar dalam pendekatan organisasi terhadap manajemen data di tengah dinamika dan kompleksitas informasi yang terus meningkat. Ini karena inovasi tersebut memungkinkan organisasi memasuki era manajemen data yang lebih canggih di mana informasi bukan hanya merupakan aset, tetapi juga alat

strategis yang membantu pengambilan keputusan, meningkatkan efisiensi operasional, dan memfasilitasi inovasi.

1.1.2 Peran Penting Basis Data dalam Organisasi

Basis data, memainkan peran penting dalam organisasi karena berfungsi sebagai tempat penyimpanan dan pengelolaan data yang digunakan oleh berbagai sistem dan aplikasi (Garcia-Molina et al., n.d.). Berikut adalah beberapa fungsi penting yang dimainkan basis data dalam organisasi:

1. Penyimpanan Data Terpusat
   Basis data menyediakan tempat penyimpanan terpusat untuk data organisasi, yang memungkinkan akses dan pengelolaan data yang relevan untuk berbagai departemen atau unit perusahaan.
2. Akses Data Efisien
   Basis data memungkinkan akses data yang efisien dan cepat. Pengguna dapat mengambil, memperbarui, atau menghapus data dengan mudah tanpa perlu menangani detail teknis penyimpanan data.
3. Integritas Data
   Basis data menggunakan aturan dan batasan, seperti aturan keunikan (seperti kunci utama), aturan referensial (seperti kunci luar), dan batasan integritas lainnya untuk memastikan bahwa data konsisten.
4. Keamanan Data
   Basis data menawarkan mekanisme keamanan untuk melindungi data sensitif, seperti pengendalian akses pengguna, enkripsi data, dan audit trail untuk memantau aktivitas yang terkait dengan basis data.
5. Kemudahan Pengelolaan
   Dengan menggunakan basis data, mengelola data menjadi lebih mudah. Administrator database dapat mengelola backup, pemulihan, dan pemeliharaan lainnya secara efektif, yang meningkatkan ketersediaan dan keandalan sistem.
6. Konsistensi Data
   Melalui penggunaan transaksi, basis data menjamin bahwa data tetap konsisten. Serangkaian operasi dapat dilakukan

secara bersamaan melalui transaksi, entah semuanya berhasil atau tidak satu pun berhasil.

7. Skalabilitas
   Sistem basis data dapat diukur ke atas (skalabilitas vertikal) atau ke samping (skalabilitas horizontal) untuk menangani volume data yang lebih besar. Dengan menggunakan basis data, organisasi dapat dengan mudah mengelola pertumbuhan data.
8. Dukungan untuk Aplikasi Bisnis
   Basis data memberikan infrastruktur yang diperlukan untuk berbagai aplikasi bisnis, termasuk sistem manajemen pelanggan (CRM), sistem manajemen rantai pasokan (SCM), dan aplikasi bisnis lainnya.

1.2 Data Versus Informasi

Basis data dalam organisasi berfungsi sebagai tempat penyimpanan dan pengelolaan data yang berasal dari fakta mentah atau detail yang belum diorganisir. Data ini dapat diubah menjadi informasi yang relevan untuk bisnis atau operasi melalui proses pengolahan. Oleh karena itu, basis data memainkan peran penting dalam mengubah data menjadi informasi yang berguna yang membantu pengambilan keputusan organisasi.

1.2.1 Konsep Dasar Data

Data dapat didefinisikan sebagai representasi fakta atau informasi dalam bentuk mentah atau tidak terorganisir, seperti angka, teks, gambar, suara, atau bentuk lainnya yang mewakili situasi atau peristiwa. Konsep dasar data mencakup pemahaman bahwa data adalah bahan mentah yang tidak memiliki arti atau konteks yang jelas (Elmasri and Navathe, 2016).

Penting untuk diingat bahwa data secara independen belum memungkinkan pemahaman yang cukup. Pengolahan, interpretasi, dan pengorganisasian adalah proses yang diperlukan untuk mengubah data menjadi informasi yang berguna. Organisasi dapat mengelola informasi dengan lebih baik, membuat keputusan yang lebih baik, dan mendukung keberhasilan operasi mereka dengan

memahami konsep dasar data, yang akan menjadi dasar bagi pembentukan informasi yang meningkatkan nilai bagi organisasi.

1.2.2 Perbedaan Data dan Informasi

Data adalah fakta mentah atau detail yang belum diorganisir atau diinterpretasikan; informasi berbeda dengan interpretasi dan konteks, yang memberikan makna pada kumpulan fakta. Data dapat berupa teks, gambar, angka, atau bentuk lainnya yang tidak memiliki definisi yang jelas. Namun, hasil dari pengolahan dan interpretasi data menjadi bentuk yang lebih terstruktur dan bermakna dikenal sebagai informasi. Dengan mengubah data menjadi informasi, kita dapat memperoleh pemahaman yang lebih baik tentang situasi atau konteks tertentu. Proses mengubah data menjadi informasi mencakup pengorganisasian, analisis, dan interpretasi. Setelah diproses, informasi dapat digunakan untuk memberikan wawasan atau membantu dalam pengambilan keputusan. Dengan kata lain, data hanya memiliki nilai terbatas tanpa interpretasi atau konteks yang tepat. Di sisi lain, informasi memberi nilai tambahan melalui pemahaman dan relevansinya dalam konteks tertentu.

1.3 Karakteristik dan Jenis-Jenis Basis Data

1.3.1 Karakteristik Basis Data

Karakteristik utama dari basis data mencakup serangkaian fitur yang dirancang untuk memenuhi kebutuhan fundamental dalam penyimpanan, pengelolaan, dan penyediaan akses efisien terhadap data. Dalam hal ini, basis data tidak hanya berfungsi sebagai penyimpanan pasif; itu juga merupakan kerangka kerja yang kompleks yang digunakan untuk merancang, mengorganisir, dan mengoptimalkan pengelolaan data dalam sistem informasi. Berikut adalah beberapa karakteristik utama basis data:

1. Terorganisir
   Data di basis data disusun dalam struktur tertentu, biasanya dalam bentuk kumpulan file terkait atau tabel. Struktur ini membuat informasi lebih mudah dicari dan dipahami.
2. Terintegrasi

Basis data memungkinkan pengguna melihat dan mengelola data dari sudut pandang yang lebih luas karena data ini diintegrasikan dari berbagai sumber dan organisasi.

3. Berbagi
   Data dapat diakses oleh banyak pengguna secara bersamaan, jadi sistem manajemen basis data (DBMS) membuat kontrol akses untuk memastikan bahwa data tetap konsisten dan aman saat diakses oleh banyak pengguna.
4. Mudah diakses
   Sistem manajemen basis data memiliki antarmuka yang memungkinkan pengguna mengakses dan mengedit data dengan kueri dan perintah.
5. Bebas Redundansi
   Basis data dibuat untuk menghindari atau mengurangi redundansi data, sehingga kemungkinan inkonsistensi data berkurang.
6. Keamanan
   Sistem manajemen basis data menawarkan mekanisme keamanan seperti kontrol akses, autentikasi pengguna, dan enkripsi data untuk mencegah orang yang tidak berhak mengakses data.
7. Integritas Data
   Aturan integritas data, seperti kunci asing dan aturan validasi, digunakan untuk menjaga kualitas data dengan memastikan bahwa data yang disimpan konsisten dan tepat.

1.3.2 Jenis-Jenis Basis Data

Jenis basis data dapat diklasifikasikan berdasarkan berbagai faktor, seperti model data, struktur penyimpanan, dan fungsionalitas. Beberapa kategori basis data yang paling umum termasuk:

1. Basis Data Hierarki
   Basis Data Hierarki adalah jenis basis data yang mengadopsi struktur hirarkis dalam mengorganisir dan menyimpan informasi. Dalam kerangka ini, data diatur secara bertingkat, dengan satu entitas utama yang mendominasi dan memiliki

hubungan hierarkis dengan entitas yang lebih rendah atau terkait di bawahnya. Setiap entitas dalam hierarki memiliki hubungan yang jelas dengan entitas di atasnya atau di tingkat yang lebih tinggi.

Struktur hirarkis ini mencerminkan hubungan "induk-anak", di mana entitas yang mendominasi berperan sebagai induk yang mengontrol atau mengelola entitas yang terkait di bawahnya. Pendekatan ini menciptakan struktur yang terorganisir secara alami, mirip dengan cabang-cabang pohon atau struktur organisasi yang hierarkis.

Sebagai contoh, dalam konteks organisasi, sebuah basis data hierarki dapat digunakan untuk merepresentasikan struktur perusahaan, di mana departemen atau unit bisnis tertentu adalah anak dari divisi yang lebih besar, dan divisi tersebut, pada gilirannya, adalah anak dari perusahaan secara keseluruhan.

Dalam basis data hierarki, akses terhadap data umumnya dilakukan dengan mengikuti jalur hierarkis. Informasi dapat diambil dengan mengikuti jejak dari entitas induk ke anak atau sebaliknya. Meskipun struktur hirarkis dapat memberikan organisasi yang jelas, penggunaan yang terlalu luas dari basis data hierarki sering kali memiliki batasan, terutama dalam hal fleksibilitas ketika berurusan dengan perubahan struktur atau kompleksitas hubungan data.Basis Data Relasional.

Sistem manajemen basis data relasional (RDBMS) seperti MySQL, PostgreSQL, dan Oracle biasanya digunakan untuk menyimpan data dalam tabel yang terhubung satu sama lain melalui kunci relasional.

2. Basis Data Relasional

   Basis Data Relasional (RDBMS) merupakan sistem manajemen basis data yang menerapkan model relasional untuk penyimpanan dan pengelolaan data. Dalam model ini, informasi diorganisir ke dalam tabel yang terdiri dari baris dan kolom. Setiap tabel mencerminkan suatu entitas atau objek, dimana kolom mewakili atribut atau properti, dan baris merepresentasikan catatan atau entitas tersebut.

Kunci relasional, termasuk kunci utama dan kunci asing, memainkan peran kunci dalam membentuk hubungan antar tabel. Kunci utama adalah identifikasi unik untuk setiap baris, sementara kunci asing mengaitkan tabel dengan mengacu pada kunci utama tabel lain. Adanya relasi ini memungkinkan pengguna untuk menggunakan SQL (*Structured Query Language*) dalam mengambil data dari berbagai tabel melalui kueri. Normalisasi, sebagai proses desain, digunakan untuk mengurangi redundansi data dan memastikan integritas data. MySQL, PostgreSQL, Oracle, dan Microsoft SQL Server adalah beberapa contoh RDBMS yang umum digunakan, menyediakan antarmuka yang kuat untuk menyimpan, mengelola, dan mengakses data secara efisien. Model relasional dan RDBMS menjadi pilihan populer dalam berbagai aplikasi dan lingkungan bisnis.

3. Basis Data Objek

   Basis Data Objek adalah suatu sistem manajemen basis data (DBMS) yang dirancang untuk menangani penyimpanan objek kompleks, seperti gambar, musik, dan video, secara efisien dalam suatu basis data. Dalam konsep ini, objek kompleks mencakup data yang lebih dari sekadar nilai teks atau numerik. DBMS ini memungkinkan pengelolaan entitas non-tradisional yang melibatkan struktur data yang lebih kompleks, seperti dokumen multimedia.

   Dalam basis data objek, entitas dapat direpresentasikan sebagai suatu objek yang memiliki atribut dan metode yang terkait dengannya. Objek-objek ini dapat mencakup data multimedia seperti gambar, audio, dan video, dan struktur ini memungkinkan penyimpanan dan pengelolaan informasi yang lebih komprehensif.

   Contohnya, dalam konteks aplikasi seni digital, basis data objek dapat digunakan untuk menyimpan dan mengelola karya seni yang mencakup gambar beresolusi tinggi, catatan audio tentang keterangan seni, dan mungkin juga video dokumentasi pembuatan karya tersebut. Keuntungan utama basis data objek adalah kemampuannya untuk menyimpan dan mengelola berbagai tipe data dalam unit tunggal,

sehingga memudahkan integrasi dan akses data yang kompleks.Penting untuk dicatat bahwa implementasi basis data objek memerlukan pemahaman konsep model objek, serta pengelolaan relasi antar objek dan pewarisan. Beberapa sistem manajemen basis data objek dapat menyediakan fleksibilitas yang lebih besar dalam hal penyimpanan dan pengambilan data yang kompleks, tetapi juga memerlukan pemahaman yang mendalam terkait dengan fitur-fitur tersebut. Sebagai contoh, Object-Relational Database Management Systems (ORDBMS) adalah evolusi dari RDBMS yang mencoba menggabungkan keuntungan dari kedua model, yaitu relasional dan objek.

4. Basis Data NoSQL

   Basis Data NoSQL menghadirkan pendekatan alternatif dalam penyimpanan dan pengelolaan data yang berbeda dari model relasional tradisional yang digunakan oleh Basis Data Relasional (RDBMS). Salah satu keunggulan utama dari basis data NoSQL adalah kemampuannya untuk menangani volume data yang besar dan fleksibilitas dalam hal skema data. Terdapat beberapa jenis basis data NoSQL, termasuk yang berbasis dokumen, grafik, kolom, dan key-value. Basis data berbasis dokumen, seperti MongoDB atau CouchDB, memungkinkan penyimpanan data dalam format dokumen dengan struktur yang dapat berubah. Sementara itu, basis data grafik seperti Neo4j fokus pada representasi dan pencarian hubungan kompleks antar entitas. Basis data kolom, seperti Apache Cassandra atau HBase, menyimpan data dalam bentuk kolom, sangat efisien untuk operasi yang melibatkan pembacaan atau penulisan sejumlah besar data pada kolom tertentu. Basis data key-value, seperti Redis atau Amazon DynamoDB, menyimpan data dalam pasangan kunci-nilai, cocok untuk kebutuhan penyimpanan dan pengambilan data yang sangat cepat. Keseluruhan, basis data NoSQL memberikan solusi yang adaptif dan scalable untuk aplikasi dengan skenario pengelolaan data yang beragam dan memerlukan kinerja tinggi. Pemilihan jenis basis data NoSQL harus disesuaikan dengan karakteristik

khusus aplikasi dan tuntutan data yang dihadapi.Basis Data Terdistribusi

Pada basis data ini, data tersebar di berbagai tempat fisik atau server.

5. Basis Data Time-Series

   Basis Data Time-Series merupakan suatu sistem manajemen basis data yang secara khusus dirancang untuk menyimpan dan mengelola data yang berkaitan dengan peristiwa yang terjadi sepanjang waktu. Karakteristik utama dari basis data ini adalah kemampuannya dalam menyimpan data yang berkembang seiring berjalannya waktu, seperti data sensor, log aktivitas, atau data lain yang dihasilkan dari peristiwa atau proses yang terus-menerus berlangsung.

   Dalam basis data time-series, data disusun berdasarkan urutan waktu, memungkinkan pengguna untuk menyelidiki dan menganalisis perubahan yang terjadi dari waktu ke waktu. Contohnya mencakup data suhu harian, pergerakan harga saham per menit, atau data produksi per jam dalam suatu pabrik.

   Keunggulan dari basis data time-series melibatkan kemampuannya untuk menangani volume data yang besar dan mengoptimalkan penyimpanan serta kueri terhadap data berbasis waktu. Dengan menyediakan indeks waktu dan struktur penyimpanan yang dioptimalkan, basis data time-series memungkinkan pengguna untuk dengan efisien menganalisis tren, mencari anomali, dan membuat proyeksi berdasarkan data historis.

   Beberapa contoh fitur yang umumnya terdapat dalam basis data time-series melibatkan penyimpanan data dalam bentuk kolom-kolom seperti timestamp, nilai, dan mungkin juga tag atau label yang mengidentifikasi sumber atau jenis data. Contoh sistem basis data time-series termasuk InfluxDB dan OpenTSDB.

   Dengan fokus pada kinerja dan efisiensi dalam menyimpan serta menganalisis data berkala, basis data time-series menjadi pilihan yang populer dalam berbagai aplikasi,

termasuk pemantauan jaringan, analisis log, dan sistem IoT (*Internet of Things*).

1.4 Manfaat Penggunaan Basis Data

Penggunaan basis data membawa manfaat signifikan dalam meningkatkan efisiensi dan efektivitas operasional suatu organisasi. Dengan menyimpan data secara terpusat dan terstruktur, proses pengambilan data menjadi lebih cepat dan akurat. Tim yang terlibat dalam kegiatan operasional dapat mengakses informasi yang relevan dengan lebih efisien, mengurangi waktu yang diperlukan untuk pencarian data manual. Selain itu, basis data memungkinkan otomatisasi tugas-tugas rutin, membebaskan sumber daya manusia untuk fokus pada tugas-tugas yang lebih kompleks dan strategis.Penggunaan basis data membawa manfaat signifikan dalam meningkatkan efisiensi dan efektivitas operasional suatu organisasi. Dengan menyimpan data secara terpusat dan terstruktur, proses pengambilan data menjadi lebih cepat dan akurat. Tim yang terlibat dalam kegiatan operasional dapat mengakses informasi yang relevan dengan lebih efisien, mengurangi waktu yang diperlukan untuk pencarian data manual. Selain itu, basis data memungkinkan otomatisasi tugas-tugas rutin, membebaskan sumber daya manusia untuk fokus pada tugas-tugas yang lebih kompleks dan strategis.

1.4.1 Peningkatan Keamanan Data

Data dapat didefinisikan sebagai representasi fakta atau informasi dalam bentuk mentah atau tidak terorganisir, seperti angka, teks, gambar, suara, atau bentuk lainnya yang mewakili situasi atau peristiwa. Konsep dasar data mencakup pemahaman bahwa data adalah bahan mentah yang tidak memiliki arti atau konteks yang jelas.

1.4.2 Peningkatan Keamanan Data

Keamanan data menjadi kritis dalam era informasi saat ini, dan basis data memberikan kerangka kerja yang kokoh untuk melindungi informasi berharga suatu organisasi. Dengan menerapkan kontrol akses dan hak pengguna, basis data

memastikan bahwa hanya pihak yang berwenang yang dapat mengakses, memodifikasi, atau menghapus data tertentu. Selain itu, langkah-langkah keamanan seperti enkripsi data dan pemantauan aktivitas sistem membantu melindungi data dari ancaman internal dan eksternal. Peningkatan keamanan data tidak hanya melibatkan perlindungan terhadap akses yang tidak sah tetapi juga pemulihan data yang andal dalam situasi darurat.

1.4.3 Dukungan Keputusan Berbasis Data

Basis data memainkan peran kunci dalam mendukung proses pengambilan keputusan yang lebih baik dan informasional. Dengan menyediakan akses cepat dan mudah ke data historis dan saat ini, basis data memungkinkan para pemimpin organisasi untuk membuat keputusan yang didukung oleh fakta dan analisis. Sistem manajemen basis data (DBMS) yang canggih juga dapat menyediakan alat analisis yang membantu dalam memahami tren, pola, dan keterkaitan dalam data. Dukungan keputusan berbasis data membantu organisasi untuk menjadi lebih responsif terhadap perubahan pasar, meningkatkan inovasi, dan meraih keunggulan kompetitif.

DAFTAR PUSTAKA

Coronel, C., Morris, S., 2019. Database systems: design, implementation, and management, 13th e. ed. Cengage Learning, Australia ; United States.

Elmasri, R., Navathe, S., 2016. Fundamentals of database systems, Seventh edition. ed. Pearson, Hoboken, NJ.

Garcia-Molina, H., Ullman, J.D., Widom, J., n.d. Database System Implementation.

BAB 2
DATABASE MANAGEMENT SYSTEM (DBMS)

Oleh Irmawati

2.1 Pendahuluan

Database Management System (DBMS) merupakan perangkat lunak yang dirancang khusus untuk mengelola dan menyimpan data secara terstruktur dalam suatu basis data. Basis data adalah kumpulan informasi yang diorganisir sedemikian rupa agar dapat diakses, dikelola, dan diperbarui dengan efisien. DBMS berfungsi sebagai perantara antara pengguna atau aplikasi dengan basis data, bertanggung jawab atas berbagai aspek teknis, termasuk penyimpanan, pengambilan, pembaruan, dan penghapusan data (Elmasri and Navathe, 2016).

Karakteristik utama DBMS melibatkan:

1. Pengelolaan Data:
   Memungkinkan pembuatan, penyimpanan, pengambilan, dan pembaruan data dengan struktur yang terorganisir.
   Data disusun dalam tabel dengan baris dan kolom, menyediakan cara yang terstruktur untuk menyimpan informasi.
2. Pengontrol Akses:
   Mengelola hak akses pengguna dan aplikasi terhadap data, menjaga keamanan dan integritas informasi.
3. Integrasi Data:
   Mendukung integrasi data dari berbagai sumber, memungkinkan penggunaan bersama oleh berbagai aplikasi.
4. Konsistensi dan Integritas Data:
   Menjaga konsistensi data melalui penerapan aturan integritas referensial dan aturan bisnis.

5. Pemulihan Data:
   Menyediakan fitur pemulihan data untuk mengembalikan informasi ke keadaan yang konsisten setelah kegagalan sistem atau kesalahan pengguna.
6. Pemakaian Transaksi:
   Mendukung konsep transaksi, memungkinkan eksekusi serangkaian operasi sebagai satu kesatuan yang tidak dapat dipisahkan, untuk memastikan konsistensi data.
7. Bahasa Query:
   Menyediakan bahasa query untuk menyusun pertanyaan atau pernyataan yang digunakan dalam pengambilan atau manipulasi data.
   Contoh DBMS yang umum digunakan meliputi MySQL, Oracle Database, Microsoft SQL Server, PostgreSQL, dan MongoDB (untuk basis data NoSQL). Pemilihan DBMS tergantung pada kebutuhan spesifik proyek atau organisasi, dengan masing-masing sistem memiliki kelebihan dan kelemahan tertentu.

2.1.1 Peran DBMS dalam Sistem Informasi

DBMS memiliki peran krusial dalam mendukung keberhasilan Sistem Informasi. DBMS tidak hanya berfungsi sebagai penyimpan data, tetapi juga sebagai pengelola, pelindung, dan pengoptimalkan proses data dalam suatu organisasi (Connolly and Begg, 2005). Beberapa peran utama DBMS dalam konteks Sistem Informasi meliputi:

1. Pengelolaan Data:
   a. Penyimpanan Data: DBMS menyediakan mekanisme efisien untuk penyimpanan dan pengelolaan data, termasuk manajemen struktur data dan penyimpanan fisik data di dalam database.
   b. Manipulasi Data: DBMS memungkinkan pengguna untuk dengan mudah melakukan operasi penyisipan, pembaruan, dan penghapusan data.
2. Keamanan Data:
   a. Otentikasi dan Otorisasi: Fitur otentikasi dan otorisasi DBMS memastikan bahwa hanya pengguna yang

berwenang yang dapat mengakses dan memanipulasi data tertentu.
   b. Enkripsi: DBMS menyediakan opsi enkripsi untuk melindungi data dari akses yang tidak sah.
3. Integritas Data:
   a. Integritas Referensial: DBMS menjaga hubungan antar data, memastikan integritas referensial di seluruh database.
   b. Ketatapan Domain: Memastikan bahwa data yang dimasukkan sesuai dengan aturan dan batasan yang ditentukan.
4. Konsistensi Data:
   Transaksi: DBMS mendukung konsep transaksi untuk menjaga konsistensi data selama operasi dan memungkinkan pemulihan jika terjadi kegagalan.
5. Manajemen Kinerja:
   a. Optimasi Query: DBMS mengoptimalkan eksekusi query untuk memberikan kinerja maksimal.
   b. Pemeliharaan Konsistensi dan Keseimbangan: Mengelola aspek kinerja seperti pemeliharaan indeks dan penyeimbangan beban.
6. Kemudahan Pengembangan Aplikasi:
   a. Antarmuka Pemrograman Aplikasi (API): DBMS menyediakan API untuk interaksi antara aplikasi dan database.
   b. Keberlanjutan dan Skalabilitas: DBMS mendukung pengembangan aplikasi yang dapat berkembang dan diukur.
7. Pemulihan Bencana:
   Backup dan Restore: DBMS menyediakan fasilitas untuk backup dan restore data, mendukung pemulihan setelah kegagalan sistem.

Dengan menyatukan semua peran ini, DBMS membentuk fondasi yang kokoh untuk Sistem Informasi yang efisien dan handal.

2.1.2 Sejarah Perkembangan DBMS

DBMS telah mengalami perkembangan yang signifikan sepanjang beberapa dekade, mencerminkan evolusi kebutuhan dan teknologi di dunia pemrosesan data. Berikut adalah sejarah perkembangan DBMS (Bai and Bhalla, 2022):

1. Model Hierarki (Awal 1960-an):
   Pada awalnya, data disimpan dalam model hierarki. Integrated Data Store (IDS), dikembangkan oleh Charles Bachman pada awal 1960-an, adalah contoh awal dari *Hierarchical Database Management System* (HDBMS).
2. Model Jaringan (Akhir 1960-an):
   Model jaringan muncul sebagai perkembangan dari model hierarki. CODASYL DBTG Model menjadi dasar untuk beberapa DBMS jaringan seperti Integrated Database Management System (IDMS).
3. Model Relasional (Awal 1970-an):
   Edgar F. Codd memperkenalkan model relasional pada tahun 1970, membawa revolusi dalam manajemen basis data. IBM menyebarkan konsep ini melalui System R dan SQL (*Structured Query Language*).
4. Proliferasi SQL (Akhir 1970-an - Awal 1980-an):
   SQL menjadi bahasa standar untuk mengelola basis data relasional. Oracle, IBM DB2, dan Microsoft SQL Server adalah beberapa produk yang muncul, membawa DBMS relasional ke pasar secara luas.
5. Pengembangan Teknologi Client-Server (Akhir 1980-an - Awal 1990-an):
   Dengan munculnya teknologi client-server, DBMS seperti Oracle, Sybase, dan Microsoft SQL Server mendukung arsitektur ini, memungkinkan aplikasi berkomunikasi dengan basis data melalui jaringan.
6. Objek-Relasional DBMS (1990-an):
   Untuk mengatasi keterbatasan model relasional, muncul DBMS objek-relasional seperti Oracle 8 dan IBM DB2.
7. Basis Data Terdistribusi (1990-an - 2000-an):
   Perkembangan teknologi jaringan dan internet mendorong

minat pada basis data terdistribusi. Oracle dan IBM DB2 terus berkembang mendukung lingkungan terdistribusi.

8. Basis Data Open Source (2000-an - Sekarang):
   MySQL dan PostgreSQL adalah contoh DBMS open source yang populer. Munculnya teknologi cloud dan pergeseran ke arah skalabilitas horizontal mempengaruhi perkembangan DBMS.
9. NoSQL dan NewSQL (2010-an - Sekarang):
   Dengan meningkatnya kebutuhan akan kinerja tinggi dan skala yang lebih besar, muncul konsep NoSQL dan NewSQL. Contoh NoSQL melibatkan MongoDB, Couchbase, dan Cassandra, sementara NewSQL melibatkan produk seperti Google Spanner dan CockroachDB.

2.2 Komponen-Komponen DBMS

DBMS terdiri dari beberapa komponen utama yang bekerja sama untuk mengelola dan menyimpan data. Berikut adalah komponen-komponen tersebut (Silberschatz, Korth and Sudarshan, 2011):

1. Database:
   Database adalah tempat penyimpanan terstruktur untuk data. Data disimpan dalam tabel, record, dan field sesuai dengan aturan dan struktur yang telah ditentukan.
2. Data *Dictionary*:
   Data *Dictionary* berisi metadata yang memberikan definisi terhadap struktur basis data. Informasi di dalamnya mencakup definisi tabel, atribut, dan hubungan antar entitas.
3. *Query Processor:*
   Query Processor bertanggung jawab untuk menerjemahkan perintah SQL (*Structured Query Language*) menjadi instruksi yang dapat dimengerti oleh DBMS. Selain itu, ia juga mengeksekusi perintah tersebut.
4. Data *Manipulation Language* (DML):
   DML adalah subset dari SQL yang digunakan untuk memanipulasi data dalam basis data. Perintah DML melibatkan operasi seperti INSERT (menambah data),

UPDATE (mengubah data), DELETE (menghapus data), dan SELECT (mengambil data).

5. Transaction Manager:
   Transaction Manager menangani transaksi, yaitu kumpulan perintah SQL yang harus dilaksanakan sebagai satu kesatuan. Hal ini membantu memastikan integritas data dalam lingkungan multi-pengguna.
6. Database Engine:
   Database Engine adalah inti dari DBMS. Ia menangani manajemen penyimpanan fisik data, indeks, caching, dan optimasi query untuk memastikan kinerja dan efisiensi yang optimal.
7. *Concurrency Control:*
   Concurrency Control memastikan bahwa transaksi yang berjalan secara bersamaan dapat beroperasi tanpa menghasilkan hasil yang tidak konsisten atau bertentangan.
8. Security and Authorization:
   Bagian ini menangani mekanisme keamanan dan otentikasi pengguna untuk mengendalikan akses ke data. Ini melibatkan pengaturan hak akses dan perlindungan data.

2.3 Model Data dalam DBMS

Model data dalam DBMS merujuk pada cara data disusun, diorganisir, dan dihubungkan dalam suatu basis data. Model data ini mencakup struktur data, aturan integritas, serta hubungan antara data. Terdapat beberapa jenis model data yang berbeda, yang dipilih tergantung pada kebutuhan spesifik dari suatu aplikasi atau proyek. Beberapa model data yang umum digunakan adalah model hierarki, model jaringan, model relasional, dan model objek (Singh, 2009).

1. Model Data Hierarki:
   a. Struktur data diatur dalam bentuk pohon atau hirarki.
   b. Setiap entitas memiliki satu entitas induk kecuali entitas teratas yang tidak memiliki induk.
   c. Contoh:
      Model Information Management System (IMS).
2. Model Data Jaringan:
   a. Struktur data diatur dalam bentuk grafik jaringan.

b. Hubungan antara entitas diwakili oleh tautan atau relasi.
c. Contoh: Model CODASYL.

3. Model Data Relasional:
   a. Data diorganisir dalam tabel dengan baris dan kolom.
   b. Tabel-tabel ini memiliki hubungan satu sama lain melalui kunci asing dan kunci utama.
   c. Contoh:
      Model relasional yang diimplementasikan dalam sistem seperti MySQL, PostgreSQL, atau Oracle.
4. Model Data Objek:
   a. Data diwakili sebagai objek yang memiliki atribut dan metode.
   b. Penggunaan konsep pemrograman berorientasi objek dalam basis data.
   c. Contoh:
      Model data objek yang diimplementasikan dalam sistem seperti MongoDB atau Couchbase.

Pemilihan model data bergantung pada kebutuhan aplikasi, kompleksitas struktur data, serta kebutuhan untuk mengakses dan mengelola data. Model relasional saat ini menjadi model yang paling umum digunakan dalam industri.

2.4 Bahasa Query

Bahasa query adalah suatu bentuk bahasa komputer yang digunakan untuk mengirimkan perintah atau kueri kepada DBMS dengan tujuan untuk mengambil, memanipulasi, atau mengelola data dalam basis data. Bahasa ini memungkinkan pengguna atau aplikasi untuk berinteraksi dengan basis data, melakukan operasi seperti pengambilan data (query), penyisipan, pembaruan, atau penghapusan data (Winand, 2011).

Dalam konteks DBMS, dua jenis bahasa query yang umum digunakan adalah:

1. SQL (*Structured Query Language*):
   SQL adalah bahasa query yang paling umum digunakan untuk berinteraksi dengan basis data relasional. SQL

memiliki perintah-perintah seperti SELECT (untuk pengambilan data), INSERT (untuk penyisipan data), UPDATE (untuk pembaruan data), DELETE (untuk penghapusan data), dan lainnya.
2. QBE (*Query by Example*): QBE adalah metode query yang memungkinkan pengguna untuk menyusun kueri dengan memberikan contoh tampilan tabel yang diinginkan. Pengguna dapat menentukan kriteria pencarian dan pemfilteran data dengan mengisi formulir atau tabel contoh.

Bahasa query sangat penting dalam pengelolaan basis data, karena memungkinkan pengguna untuk berkomunikasi dengan sistem basis data dan mendapatkan hasil yang diinginkan. Pemahaman bahasa query menjadi keterampilan yang krusial bagi pengembang perangkat lunak, administrator basis data, dan profesional IT lainnya yang bekerja dengan sistem manajemen basis data.

2.5 Transaksi dalam DBMS

Dalam DBMS, transaksi pada unit pekerjaan atau aktivitas yang dapat dilakukan di dalam basis data. Transaksi ini dapat mencakup satu atau lebih operasi database yang dilakukan secara bersamaan dan membentuk suatu tindakan yang konsisten terhadap basis data. Transaksi harus mematuhi prinsip ACID, yang merupakan singkatan dari Atomicity (Atomicitas), Consistency (Konsistensi), Isolation (Isolasi), dan Durability (Daya Tahan) (Foster and Godbole, 2022).

1. *Atomicity* (Atomicitas): Transaksi dianggap sebagai operasi tunggal yang either dilakukan secara keseluruhan atau tidak dilakukan sama sekali. Jika satu bagian dari transaksi gagal, seluruh transaksi harus dibatalkan dan tidak ada perubahan yang diterapkan ke dalam basis data.
2. *Consistency* (Konsistensi): Transaksi harus membawa basis data dari satu keadaan yang konsisten ke keadaan lain yang juga konsisten. Ini berarti bahwa transaksi harus memastikan bahwa integritas referensial dan semua aturan bisnis terpenuhi setelah transaksi selesai.

3. *Isolation* (Isolasi): Transaksi yang berjalan secara bersamaan seharusnya tidak saling mengganggu satu sama lain. Isolasi memastikan bahwa hasil sementara dari suatu transaksi tidak terlihat oleh transaksi lain sampai transaksi tersebut selesai.
4. *Durability* (Daya Tahan): Setelah transaksi selesai, perubahan yang diterapkan ke dalam basis data harus tetap ada bahkan jika terjadi kegagalan sistem atau pemadaman listrik. Durability memastikan bahwa hasil dari transaksi tetap persisten dan dapat diandalkan.

2.6 Keamanan dalam DBMS

Keamanan dalam DBMS pada serangkaian langkah dan kebijakan yang dirancang untuk melindungi data yang disimpan dalam basis data dari potensi ancaman seperti akses tidak sah, perubahan yang tidak diinginkan. Kehilangan atau penyalahgunaan data dalam basis data dapat memiliki konsekuensi serius, terutama karena basis data sering mengandung informasi yang sangat bernilai, termasuk data pribadi, keuangan, dan bisnis (Bertino and Sandhu, 2005).

Beberapa aspek kunci dalam mencapai keamanan DBMS mencakup:

1. Otentikasi dan Otorisasi:
   Melibatkan proses memastikan bahwa hanya pengguna yang sah yang memiliki akses ke basis data, dan mereka memiliki izin yang sesuai untuk melakukan tindakan tertentu. Ini mencegah akses tidak sah dan penggunaan data yang tidak sesuai.
2. Enkripsi:
   Menggunakan teknik enkripsi untuk melindungi data, sehingga bahkan jika akses fisik ke server basis data diperoleh, data tetap aman karena hanya dapat dibaca dengan kunci enkripsi yang benar.
3. Audit dan Pemantauan:
   Mencatat aktivitas pengguna dan perubahan data untuk memungkinkan deteksi dini terhadap aktivitas mencurigakan. Pemantauan ini membantu dalam mengidentifikasi dan menanggapi insiden keamanan.

4. Backup dan Pemulihan: Melakukan pencadangan secara teratur untuk memastikan bahwa data dapat dipulihkan dengan cepat dan efisien dalam situasi kehilangan data atau serangan.
5. Firewall:
   Mengimplementasikan langkah-langkah keamanan jaringan dan menggunakan firewall untuk melindungi akses ke basis data dari ancaman eksternal.
6. Pembaruan dan Pemeliharaan Sistem:
   Menjaga perangkat keras dan perangkat lunak yang mendukung DBMS tetap diperbarui dengan pembaruan keamanan untuk mengatasi potensi kerentanan.
7. Manajemen Kata Sandi:
   Menerapkan kebijakan keamanan yang kuat terkait dengan kata sandi pengguna, termasuk kebijakan kompleksitas dan rotasi kata sandi secara teratur.
8. Proteksi terhadap Serangan Terhadap Basis Data: Mengimplementasikan langkah-langkah perlindungan terhadap serangan umum terhadap basis data, seperti SQL *injection* dan *cross-site scripting*.

DAFTAR PUSTAKA

Bai, Y. and Bhalla, S. (2022) 'Introduction to databases', in *Oracle Database Programming with Java*. Auerbach Publications, pp. 9–66.

Bertino, E. and Sandhu, R. (2005) 'Database security-concepts, approaches, and challenges', *IEEE Transactions on Dependable and secure computing*, 2(1), pp. 2–19.

Connolly, T.M. and Begg, C.E. (2005) *Database systems: a practical approach to design, implementation, and management*. Pearson Education.

Elmasri, R. and Navathe, S.B. (2016) 'Fundamentals of Database Systems 7th ed.' Pearson.

Foster, E. and Godbole, S. (2022) *Database systems: a pragmatic approach*. Auerbach Publications.

Silberschatz, A., Korth, H.F. and Sudarshan, S. (2011) 'Database system concepts'.

Singh, S.K. (2009) *Database systems: Concepts, design and applications*. Pearson Education India.

Winand, M. (2011) 'SQL performance explained', *Development*, 2011, pp. 3–8.

BAB 3
BASIS DATA RELASIONAL

Oleh Yulita Salim

3.1 Pendahuluan

Basis Data Relasional adalah jenis basis data yang mengatur data ke dalam baris dan kolom yang secara kolektif membentuk tabel tempat titik-titik data saling terkait satu sama lain. Basis data relasional diperkenalkan pertama kali oleh Edgar Frank Codd.

Saat ini, basis data relasional digunakan dalam berbagai aplikasi seperti sistem keuangan, penelitian ilmiah ataupun *e-commerce*. Data biasanya terstruktur dalam beberapa tabel, yang dapat digabungkan melalui kunci utama atau kunci asing. Pengidentifikasi unik ini menunjukkan hubungan yang berbeda antar tabel, dan hubungan ini biasanya diilustrasikan melalui berbagai jenis model data. Analis menggunakan queri SQL untuk menggabungkan titik data yang berbeda dan merangkum kinerja bisnis, memungkinkan organisasi memperoleh wawasan, mengoptimalkan alur kerja, dan mengidentifikasi peluang baru.

Basis data relasional biasanya dikaitkan dengan basis data transaksional, yang menjalankan perintah, atau transaksi, secara kolektif. Contoh populer yang digunakan untuk menggambarkan hal ini adalah transfer bank. Jumlah tertentu ditarik dari satu akun, dan kemudian disetorkan ke akun lain. Jumlah total uang ditarik dan disimpan, dan transaksi ini tidak dapat terjadi secara parsial

Beberapa istilah penting yang perlu diketahui dalam basis data relasional adalah:

1. Tabel (Relasi): struktur yang menyimpan data dalam bentuk baris dan kolom. Setiap baris akan mewakili sebuah data dan setiap kolom mewakili attribut atau data tertentu.
2. Kolom (*Field/Attribute*): kolom berisi atribut yang biasa disebut field atau kolom atribut. Setiap kolom memiliki nama dan tipe data yang menyimpan nilai yang terkait dengan entitas dalam baris.

3. Baris (*Data/Record/Tuple*): baris berisi nilai data yang mewakili satu entitas unik dan data untuk setiap kolom dalam tabel.
4. *Primary Key*: satu kolom atau lebih yang berfungsi untuk mengidentifikasi tabel menggunakan kunci utama yang bersifat unik.
5. *Foreign Key*: kolom atau kumpulan kolom yang membentuk kunci unik dari tabel lain yang digunakan untuk mendefenisikan hubungan antara tabel.
6. Normalisasi: proses pengorganisasian data dalam basis data yang berfungsi untuk mengurangi redundansi dan meningkatkan integritas data.
7. SQL (*Structured Query Language*): SQL merupakan bahasa standar yang digunakan untuk mengelola dan memanipulasi basis data relasional. SQL ini dapat membantu melakukan berbagai operasi seperti SELECT (memilih data), INSERT (menyisip data), UPDATE (memperbaharui data), dan DELETE (menghapus data). SQL menjadi standar dari American National Standards Institute (ANSI) pada tahun 1986. Semua mesin basis data relasional yang populer mendukung standar SQL ANSI.

Fitur utama dari sistem basis data relasional adalah:

a. Model data

Database relasional terdiri dari tabel yang mewakili entitas atau objek dunia nyata. Setiap kolom dalam tabel ini ditunjuk untuk menyimpan atribut tertentu, sementara bidang berisi nilai sebenarnya dari atribut. Baris dan kolom dalam tabel secara kolektif menyimpan nilai-nilai terkait yang berkaitan dengan objek tunggal atau entitas.

Kunci utama dapat ditetapkan ke setiap baris dalam tabel sebagai pengidentifikasi unik. Kunci asing, di sisi lain, membuat koneksi dengan mengacu pada kunci utama di tabel lain. Hubungan antara baris dalam tabel yang berbeda dibuat melalui pasangan kunci primer dan asing. Misalnya, kunci asing ID pelanggan dalam tabel pesanan dapat menunjuk ke baris

yang sesuai dalam tabel pelanggan, yang mencakup semua informasi pelanggan yang relevan.

b. Transaksi

Transaksi database relasional terdiri dari satu atau lebih pernyataan SQL yang dieksekusi secara berurutan untuk membuat unit kerja yang kohesif dan tak terpisahkan. Transaksi mematuhi prinsip atomisitas, yang mengharuskan seluruh operasi berhasil secara keseluruhan atau sepenuhnya dibatalkan. Kegagalan untuk mengeksekusi bagian mana pun dari transaksi mengakibatkan tidak ada komponen individualnya yang diproses.

Dalam kerangka model relasional, transaksi diakhiri dengan pernyataan COMMIT yang menandakan penyelesaian yang berhasil atau pernyataan ROLLBACK yang menunjukkan kegagalan. Sistem manajemen basis data memastikan bahwa setiap transaksi ditangani secara konsisten dan andal, dengan otonomi dan isolasi dari transaksi bersamaan lainnya.

c. Kepatuhan pada ACID

Semua transaksi basis data relasional harus sesuai dengan *Atomic, Consistent, Isolated,* and *Durable* (ACID) untuk memastikan integritas data.

Atomisitas

Konsep Atomisitas mengharuskan keberhasilan pelaksanaan transaksi secara keseluruhan. Jika ada segmen transaksi mengalami kegagalan, setiap modifikasi yang dilakukan dalam transaksi akan dibatalkan.

Konsistensi

Konsistensi menetapkan bahwa data yang dimasukkan ke dalam database relasional dalam transaksi harus mematuhi semua peraturan dan batasan yang telah ditentukan, mencakup batasan, kaskade, dan pemicu.

Isolasi
Isolasi mengharuskan bahwa setiap transaksi beroperasi secara independen. Secara bersamaan, ketika banyak pengguna berusaha untuk memodifikasi data dalam database relasional, mekanisme untuk kontrol konkurensi menghambat pengguna dari mengesampingkan modifikasi satu sama lain.

Daya Tahan
Daya tahan mengharuskan bahwa semua perubahan yang dilakukan pada database relasional menjadi permanen setelah berhasil menyelesaikan transaksi.

3.2 Dasar Basis Data

Sebelum munculnya database relasional, setiap organisasi menggunakan sistem database hierarki yang berbentuk struktur seperti pohon untuk menyimpan tabel data. Awalnya *Database Management System* (DBMS) akan memfasilitasi organisasi sistematis yang memiliki sejumlah besar data. Meskipun demikian, data ini sering terbukti rumit khusus untuk aplikasi tertentu, dan menyajikan keterbatasan dalam hal representasinya. Kendala ini mendorong Edgar F. Codd, seorang peneliti IBM merilis sebuah artikel pada tahun 1970 berjudul "A Relational Model of Data for Large Shared Data Banks" yang menggambarkan konsep model database relasional. Artikel ini menganjurkan kompilasi data berdasarkan hubungan, di mana tupel atau set tupel, yang dikenal sebagai relasi digunakan yang pada akhirnya memungkinkan integrasi data di beberapa tabel.

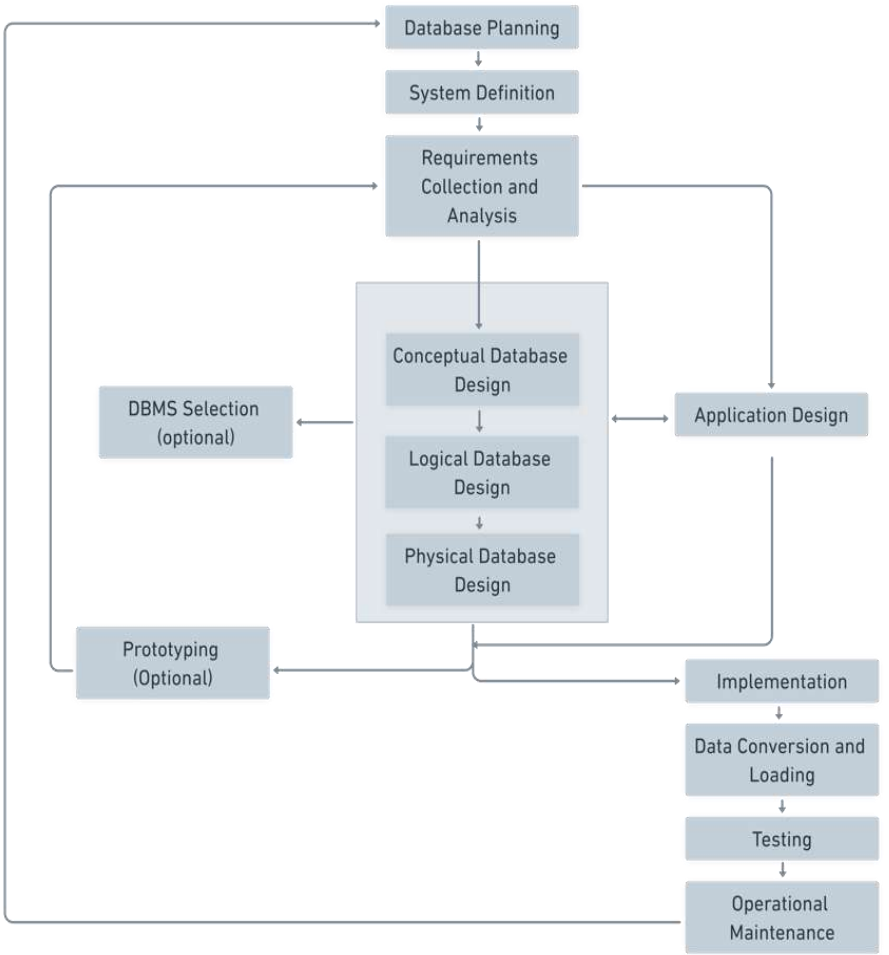
Pada tahun 1973, Laboratorium Penelitian San Jose, yang saat ini diakui sebagai Pusat Penelitian Almaden, memprakarsai Sistem R (R singkatan dari relasional) untuk memvalidasi teori relasional sebagai "implementasi kekuatan industri." Selanjutnya, IBM meluncurkan keluarga database relasional DB2 pada tahun 1983, menandai tahapan kedua dari perangkat lunak manajemen database IBM. Terkenal sebagai salah satu produk IBM yang paling berhasil, DB2 terus mengelola miliaran transaksi setiap hari dalam infrastruktur cloud dan berfungsi sebagai elemen dasar untuk aplikasi pembelajaran mesin.

Sejak awal teknologi komputasi, domain manajemen data dan penyimpanan pada mesin telah berkembang menjadi arena penyelidikan ilmiah yang abadi. Seperti yang dikemukakan oleh Whiteen et al. (2004), data mewujudkan informasi yang tidak diproses mengenai individu, lokasi, kejadian, dan entitas signifikan dalam suatu organisasi. Transformasi data menjadi informasi bergantung pada terjemahannya, sehingga menghasilkan penciptaan pengetahuan baru. Sedangkan basis data (*database*) merupakan sekumpulan logical data yang saling terhubung satu sama lain dan dapat dirancang menjadi suatu informasi yang dibutuhkan dalam sebuah organisasi. Ketika sebuah sistem memungkinkan seorang pengguna untuk mendefenisikan, membuat, dan memelihara basis data dan menyediakan akses kontrol kepada data base maka dapat disebut *Database Management System* (DBMS). Dua bagian utama dari DBMS adalah:

1. *Data Definition Language* (DLL)
   DLL adalah bahasa yang memungkinkan seorang database administrator (DBA) mendeskripsikan nama entitas, atribut dan relasi data yang diminta oleh aplikasi secara bersamaan dengan integritas data dan keamanan datanya.

2. *Data Manipulation Language* (DML)
   DML adalah bahasa yang menyediakan sekumpulan operasi untuk mendukung operasi manipulasi data sederhana pada data yang tersimpan dalam basis data.

Tahapan sebuah basis data di analisis dan dirancang dapat terlihat pada gambar 3.1 dibawah.



Gambar 3.1. Diagram Siklus Basis Data

3.3 Normalisasi Basis Data

Normalisasi basis data merupakan proses pengelompokan atribut data yang membentuk entitas sederhana, efisien, dan mudah beradaptasi. Untuk mengelola.

3.3.1 Tahapan Normalisasi Basis Data

1. *Unnormalized Form* (UNF)
   Ketidakteraturan struktur pada data, ditandai dengan terjadinya kelompok data yang berulang dalam dataset sehingga menimbulkan tantangan selama manipulasi data.
2. *First Normal Form* (1NF)
   Pada normalisasi tahap pertama setiap sel dalam tabel hanya berisi satu nilai dan tidak ada kelompok yang berulang pada kolom yang sama.
3. *Second Normal Form* (2NF)
   Pada bentuk normalisasi kedua, setiap non-key akan tergantung sepenuhnya pada kunci utama (primer), artinya tidak ada atribut non-key yang secara fungsional bergantung pada sebagian kunci utama.
4. *Third Normal Form* (3NF)
   Pada bentuk normalisasi ketiga tabel harus sudah dalam bentuk 2NF dan setiap atribut non-key tidak boleh memiliki ketergantungan transitif pada kunci utama.
5. *Boyce-Codd Normal Form* (BCNF)
   Bentuk normalisasi BCNF bertujuan untuk mengatasi anomali dan overlooping yang tidak dapat diatasi dalam bentuk 3NF.
6. *Fourth Normal Form* (4NF)
   4NF dilakukan dengan tujuan untuk mengatasi terjadinya joint dependent sehingga terjadi pemecahan relasi menjadi dua.
7. *Fifth Normal Form* (5NF)
   Pada 5NF, bentuk normalisasi bertujuan untuk mengatasi terjadinya joint dependent sehingga terjadi pemecahan relasi menjadi dua.

3.4 Desain Basis Data

Desain basis data adalah proses perancangan struktur dan organisasi basis data agar dapat memenuhi kebutuhan bisnis, memastikan efisiensi dalam penyimpanan dan akses data serta untuk menjaga integritas data.

Untuk mendesain basis data, beberapa langkah yang diperlukan adalah:

1. Analisa kebutuhan bisnis, sebelum mendesain database kita perlu memahami kebutuhan bisnis serta mengetahui bagaimana data akan digunakan. Langkah ini memerlukan identifikasi entitas bisnis, atribut terkait dan hubungan antar entitas.
2. Pemodelan konseptual, setelah melakukan analisa terhadap kebutuhan bisnis entitas utama dan hubungan antar entitas di modelkan secara konseptual menggunakan teknik misalnya *Entity Relationship Diagram* (ERD). ERD akan membantu dalam memvisualisasikan struktur data serta hubungan antar entitas dalam lingkup bisnis yang relevan. Pada pemodelan konseptual terdapat tiga tahapan yang harus dikerjakan yaitu:
   a. Penentuan *entity* (entitas) basis data
   b. Pendefenisian relationship (hubungan) antar entitas
   c. Penerjemahan hubungan ke dalam entitas
3. Normalisasi, pada tahap normalisasi perlu dipastikan agar struktur database tidak mengandung anomaly dan redundansi yang tidak diinginkan.
4. Pemodelan logikal, setelah normalisasi lalu desain basis data dikembangkan lebih lanjut menjadi model logis. Model konseptual akan dibentuk kedalam model yang lebih terstruktur dan terstandarisasi, seperti skema relasional dalam bentuk tabel, kolom, dan kunci-kunci.
   Kegiatan pada pemodelan logical terdiri atas dua yaitu membangun sebuah model data logical lokal dari model data konseptual yang memberikan gambaran perusahaan untuk memastikan struktur tabel sudah benar. Dan yang kedua adalah membuat kombinasi model data logical lokal individu ke dalam sebuah model data logical global tunggal yang menggambarkan objek atau perusahaan.
   Hasil dari kegiatan ini adalah kamus data yang berisi semua atribut dan *key* berupa *primary key, alternate key*, dan *foreign key* serta ERD secara keseluruhan.

5. Optimasi kinerja, tahap ini akan melibatkan penyesuaian desain basis data untuk meningkatkan kinerja sistem. Tahap ini meliputi pemilihan indeks yang tepat, pemilihan tipe data yang efisien, dan pengoptimalan struktur query.
6. Implementasi fisik, selanjutnya adalah implementasi ke dalam sistem manajemen basis data (DBMS). Pada tahap ini dilakukan Pembuatan tabel, indeks, dan constraint berdasarkan model logis yang dikembangkan sebelumnya. Tujuan Pembuatan model relational adalah:
   a. Membuat kumpulan table relational dan constraints pada tabel dimana informasi diperoleh dari model data logical.
   b. Mengidentifikasi struktur penyimpanan tertentu dan metode akses terhadap data untuk mencapai hasil optimal dari sistem basis data.
   c. Merancang produksi keamanan untuk sistem.
7. Pengujian dan evaluasi, setelah melalui tahap implementasi, basis data di uji untuk memastikan telah memenuhi kebutuhan bisnis, berkinerja baik, dan menjaga integritas data. Pengujian ini akan melibatkan pengujian fungsionalitas, kinerja, keamanan, dan keandalan basis data.

Desain basis data yang baik dapat meliputi kriteria:

1. Efisiensi: basis data harus dirancang untuk beroperasi secara efisien, baik dalam penyimpanan maupun pengaksesan data.
2. Integritas: basis data harus menjaga integeritas data seperti referensial yaitu hubungan antar data harus didefenisikan dengan jelas dan dijaga agar data tetap konsisten.
3. Konsistensi: struktur, format dan nilai-nilai basis data harus konsisten, yaitu data yang sama memiliki representasi yang seragam di seluruh sistem.
4. Keamanan: desain basis data harus memperhatikan aspek keamanan data, kebijakan akses, enkripsi data, serta pemantauan aktifitas pengguna.
5. Fleksibilitas: basis data harus dapat mengakomodasi perubahan kebutuhan bisnis tanpa mengganggu operasi

yang sedang berjalan. Hal ini dapat mencakup kemampuan menambahkan atau mengubah struktur data tanpa mengganggg aplikasi yang bergantung pada basis data tersebut.

3.5 Perbedaan Basis Data Relasional dan Non-Relasional

Perbedaan basis data relasional dan non-relasional (NoSQL) berkaitan dengan struktur data yang dikandungnya. Model basis data relasional mengatur data ke dalam tabel, indeks, dan tampilan. Struktur tabular ini memudahkan untuk membuat, membaca, mengubah, dan menghapus data yang relevan menggunakan bahasa queri, seperti SQL. Struktur setiap barisnya sama, seperti *spreadsheet*.

Basis data non-relasional tidak menggunakan struktur data tabular. Sebagai gantinya, data dapat disimpan sebagai pasangan nilai-kunci, JSON, grafik, atau hampir semua tipe struktur data lainnya. Banyak basis data non-relasional disebut basis data NoSQL karena data disimpan dan diqueri dengan cara yang tidak memerlukan SQL.

Basis data non-relasional atau NoSQL, dibuat khusus untuk model data tertentu dan memiliki skema yang fleksibel untuk membangun aplikasi modern. Basis data NoSQL dikenal secara luas karena kemudahan pengembangan, fungsionalitas, dan performa dalam skala besar. Beberapa perbedaan Basis Data Relasional dan Non-Relasional antara lain:

1. **Mekanisme penyimpanan data**

   Basis data relasional menyimpan data terstruktur dalam baris dan kolom berbasis aturan. Sebaliknya, basis data NoSQL menyimpan elemen data individual dalam file terpisah.

2. **Struktur fleksibel**

   Basis data relasional menyimpan data dalam format tabular dan mengikuti aturan yang ketat mengenai variasi data dan hubungan tabel. Basis data NoSQL menawarkan fleksibilitas yang lebih besar karena tidak memerlukan data terstruktur. Anda dapat menggunakannya

untuk menyimpan file, video, dan konten tidak terstruktur lainnya.

3. **Mekanisme integritas data**

   Model basis data relasional mengikuti properti ACID yang ketat. Secara tradisional, basis data NoSQL menawarkan model BASE (*Basically Available, Soft state, Eventual consistency)* yang lebih fleksibel. Mereka menjamin ketersediaan tetapi tidak memiliki konsistensi yang kuat. Status basis data dapat berubah dari waktu ke waktu, yang pada akhirnya menjadi konsisten. Basis data NoSQL modern juga menawarkan ACID, konsistensi yang kuat, ketersediaan yang tinggi, dan banyak lagi.

DAFTAR PUSTAKA

https://aws.amazon.com/id/relational-database/
https://www.ibm.com/topics/relational-databases
https://www.oracle.com/database/what-is-a-relational-database/
https://www.dicoding.com/blog/mengenal-apa-itu-relational-database/
https://repository.unikom.ac.id/50882/1/

BAB 4
DEPENDENSI DAN NORMALISASI

Oleh Budi Wijaya Rauf

4.1 Pendahuluan

Dependensi dan normalisasi dalam basis data merupakan topik yang sangat penting dalam desain dan pengelolaan basis data. Dependensi dalam basis data merujuk pada hubungan antara entitas atau atribut dalam basis data, yang dapat mempengaruhi integritas data dan efisiensi operasi basis data (Suprayogi et al., 2016). Sebaliknya, normalisasi adalah proses transformasi data dalam basis data untuk mengurangi redundansi dan anomali data, sehingga memastikan basis data tetap konsisten dan efisien (Nasution et al., 2019).

Tujuan utama dari memahami dan menerapkan dependensi dan normalisasi adalah untuk menciptakan basis data yang optimal. Basis data yang telah dinormalisasi dengan baik akan lebih mudah dikelola dan dikembangkan serta lebih tahan terhadap masalah yang sering muncul dalam basis data yang dirancang dengan buruk. Beberapa manfaat utama dari normalisasi dan pemahaman dependensi meliputi pengurangan redundansi, peningkatan konsistensi, penghapusan anomali, dan peningkatan efisiensi query.

Meski normalisasi membawa banyak manfaat, proses ini juga menghadirkan tantangan. Salah satu tantangan utamanya adalah kebutuhan akan keseimbangan antara normalisasi dan kinerja sistem. Terkadang, normalisasi yang terlalu ketat dapat menyebabkan banyak join dalam query, yang dapat memperlambat kinerja. Oleh karena itu, sering kali diperlukan pendekatan yang pragmatis yang mempertimbangkan kebutuhan khusus dari aplikasi dan lingkungan operasional.

Dengan pemahaman yang baik tentang dependensi dan normalisasi, pengembang basis data dapat merancang sistem yang tidak hanya efisien dan konsisten, tetapi juga fleksibel dan mampu beradaptasi dengan kebutuhan bisnis yang terus berkembang. Bab

ini akan membahas secara mendetail konsep-konsep ini, memberikan panduan praktis untuk implementasi, serta studi kasus untuk memperjelas penerapan dalam situasi nyata.

4.2 Dependensi

Dependensi atau ketergantungan adalah hubungan antara entitas atau antribut dalam basis data. Dependensi penting untuk dipahami dan diidentifikasi agar dapat mengetahui ketergantungan antar atribut sehingga dapat mencegah redudansi data untuk basis data yang efisien.

4.2.1 Dependensi Fungsional

Dependensi fungsional merupakan konsep kunci dalam teori basis data yang menggambarkan hubungan antar atribut dalam sebuah relasi tabel. Atribut yang dikatakan sebagai dependensi fungsional kepada atribut yang lain adalah jika nilai dari atribut tersebut menentukan nilai dari atribut yang lain. Dependensi fungsional dapat diartikan jika terdapat setidaknya dua atribut dalam sebuah tabel, anggap saja atribut A dan atribut B, maka atribut B dapat dikatakan dependensi fungsional terhadap A jika nilai unik dari atribut A menentukan nilai dari atribut B (A -> B). Contoh sederhana dari dependensi fungsional :

Tabel 4.1. Dependensi Fungsional

NIM	Nama	Alamat
012345	Budi Wijaya Rauf	Jalan Nusa
023456	Sujiwo Tejo	Jalan Indah

Pada tabel 4.1, atribut nama dan atribut alamat sangat bergantung terhadap NIM. Dimana ketika nilai dari atribut NIM berbeda maka nilai nama dan alamat pun juga akan berbeda. Inilah yang disebut sebagai dependensi fungsional.

4.2.2 Dependensi Fungsional Penuh

Dependensi fungsional penuh terjadi jika atribut B bergantung secara fungsional penuh dengan himpunan A dan tidak pada salah satu atribut A. Himpunan A merujuk pada beberapa atribut, jika A = {A_1, A_2, ... A_n} maka atribut B dapat dikatakan dependensi fungsional penuh jika :

1. Himpunan A -> B
2. Tidak ada subset dari himpunan A yang dapat mempengaruhi nilai atribut B

Contoh dari dependensi fungsional penuh dapat dilihat pada tabel 4.2 berikut ini.

Tabel 4.2. Dependensi Fungsional Penuh

NIP	KaryaId	Total Gaji
8629101001	01	5000000
2010025001	02	10000000

Dependensi fungsional penuh sebagaimana yang digambarkan pada tabel 4.2 adalah atribut gaji sangat bergantung dengan dua atribut yaitu NIP dan karyaId. Jadi dapat dikatakan NIP, KaryaId -> Total Gaji.

4.2.3 Dependensi Fungsional Parsial

Dependensi fungsional parsial terjadi jika atribut B bergantung pada himpunan atribut A akan tetapi salah satu subset dari himpunan A dapat mempengaruhi nilai dari atribut B. Mari memakai contoh kasus yang sama seperti pada tabel 6.2. Pada tabel tersebut, NIP dan KaryaId memiliki pengaruh terhadap nilai dari total gaji. Namun, bagaimana jika salah satu subset dari himpunan tersebut memiliki pengaruh terhadap total gaji ? NIP menentukan nilai dari gaji pokok sehingga ini dapat dikatakan sebagai dependensi fungsional parsial (NIP -> gajipokok).

4.2.4 Dependensi Fungsional Transitif

Dependensi fungsional transitif dapat dimaknai jika atribut B bergantung pada atribut A, tapi atribut C bergantung pada atribut B, maka secara transitif atribut A mempengaruhi nilai dari atribut C.

Tabel 4.3. Dependensi Fungsional Transitif

NIM	Nama	No. HP
214125	Joko	082345129xxx
203213	Miswar	082442123xxx
242151	Tejo	082299014xxx

Pada tabel 4.3 atribut nama bergantung pada NIM dan nomor hp bergantung pada atribut nama, sehingga NIM -> Nama dan Nama -> No.HP inilah yang disebut sebagai dependensi fungsional transitif.

4.2.5 Dependensi Fungsional *Multivalued*

Dependensi fungsional *multivalued* merupakan ketergantungan dua atau lebih pada atribut lain secara independen. Sebagai contoh pada tabel 4.4 terjadi dependensi *multivalued* 'Kursus' -> 'Materi' dimana atribut 'Kursus' dan subset 'Materi' dapat berbeda secara independen dari 'Pengajar'.

Tabel 4.4. Dependensi Fungsional *Multivalued*

Kursus	Materi	Pengajar
Matematika	Aljabar	Budi
Matematika	Kalkulus	Budi
Matematika	Aljabar	Mita
Matematika	Kalkulus	Mita

4.3 Normalisasi

Normalisasi merupakan teknik yang dapat memproses secara sistematis tabel maupun skema relasi tabel sehingga dapat menciptakan bentuk normalm, teknik ini pertama kali diperkenalkan oleh Codd (1972). Proses ini biasanya melibatkan pembagian tabel yang besar lalu dipecah menjadi tabel kecil. Normalisasi bertujuan untuk membentuk struktur basis data yang efisien dan konsisten sehingga memudahkan dalam pengelolaan dan penggunaan data.

4.3.1 Macam-Macam Anomali

Bentuk tidak normal pada sebuah tabel dikatakan sebagai anomali, tabel dengan anomali menciptakan tabel yang tidak efisien dan konsisten sehingga dapat menyebabkan masalah pengelolaan dan penggunaan data. Anomali ada 3 bentuk yaitu:

1. Anomali Penyisipan (*Insertion Anomaly*)
   Anomali penyisipan terjadi ketika menambahkan data baru ke dalam sebuah basis data akan tetapi data lain yang tidak relevan harus dimasukkan, contoh:

Tabel 4.5. Anomali Penyisipan

NIM	Nama	KodeMataKuliah	NamaMataKuliah
02141	Temon	01	Matematika
01120	Abdel	02	Bahasa Indonesia
02323	Tulus	04	Kewarganegaraan
01234	Mamut	03	Bahasa Inggris

   Pada tabel 4.5, terjadi anomali penyisipan. Ketika pengguna hanya ingin memasukkan mata kuliah, pengguna harus memasukkan NIM dan nama mahasiswa, yang mana tidak relevan dengan mata kuliah. Begitu pun ketika hanya ingin memasukkan mahasiswa, pengguna wajib memasukkan mata kuliah ke dalam tabel.

2. Anomali Penghapusan (*Deletion Anomaly*)
   Anomali penghapusan merupakan anomali yang terjadi ketika menghapus data tertentu akan tetapi data lain yang tidak relevan turut terhapus. Anomali ini menyebabkan informasi hilang dan mengganggu integritas data. Berikut contoh dari anomali penghapusan.

Tabel 4.6. Anomali Penghapusan

NIM	Nama	Kode Mata Kuliah	Nama Mata Kuliah
02323	Tulus	04	Kewarganegaraan
01234	Mamut	03	Bahasa Inggris

Seperti tabel 4.6, ketika hanya ingin menghapus data nama mahasiswa, data untuk matakuliah akan turut terhapus sehingga akan menyebabkan hilangnya informasi. Sebagai contoh, ketika ingin menghapus nama mahasiswa Tulus dari basis data, maka data 'KodeMataKuliah' dan 'NamaMataKuliah' juga akan turut terhapus.

3. Anomali Pembaruan (*Update Anomaly*)
   Anomali pembaruan terjadi ketika perubahan data yang sama di berbagai tempat dalam basis data menyebabkan inkonsistensi data. Hal ini sering terjadi jika data yang sama disimpan di beberapa lokasi yang berbeda.

Tabel 4.7. Anomali Pembaruan

NIM	Nama	KodeMataKuliah	NamaMataKuliah
02323	Tulus	04	Kewarganegaraan
01234	Mamut	03	Bahasa Inggris
02135	Nimbus	04	Kewarganegaraan
12341	Mita	03	Bahasa Inggris

Data pada tabel 4.7 memiliki beberapa persamaan, kolom 'KodeMataKuliah' dan 'NamaMataKuliah' memiliki beberapa data yang sama pada baris yang lain. 'KodeMataKuliah' 04 yang merupakan 'NamaMataKuliah' Kewarganegaraan memiliki data yang sama dengan baris ketiga, sehingga ketika ingin mengganti nama 'NamaMataKuliah' 04 menjadi Pendidikan Pancasila, maka seluruh kolom yang mengandung 'KodeMataKuliah' 04 harus diperbarui.

4.3.2 Bentuk Normalisasi

Normalisasi ditujukan untuk menjaga integritas data sehingga relasi antar tabel dalam sebuah basis data dapat mudah untuk dipahami, dikelola, dan mudah untuk dikembangkan sesuai dengan kebutuhan. Untuk mencapai tujuan yang diinginkan, maka ada beberapa tahapan proses yang perlu untuk diikuti, antara lain adalah :

1. Bentuk Normal Pertama (1st Normal Form/1NF)
   Bentuk normal pertama merupakan tahapan awal dari rentetan proses normalisasi. Ada beberapa syarat yang harus terpenuhi yaitu :
   a. Tidak ada atribut yang duplikat dalam sebuah tabel.
   b. Tidak ada baris yang duplikat.
   c. Nilai dari sebuah data dalam kolom harus atomik (nilai tunggal).

   Langkah untuk memenuhi syarat 1NF :
   a. Hilangkan atribut yang duplikat.
   b. Buatlah tabel yang terpisah untuk setiap grup data lalu buat atribut relasinya.
   c. Tipe data dari setiap kolom konsisten
   d. Setiap baris memiliki kolom yang unik sebagai identitas (*primary key*)

   Contoh dari tabel yang tidak normal dapat dilihat pada tabel 4.7.

Tabel 4.8. Bentuk Tidak Normal

Nota	Pelanggan	NamaBarang	Qty	Harga	Total
B01	Sutris	Biskuit ;Marimas; Nata de Coco	2 ;1 ; 3	10000;2000 ;15 000	67000
B02	Joko	Rinso;Marjan	1 ;1	8000 ;12000	20000
B03	Prabski	Beras	1	50000	50000

Bentuk tabel 4.8 merupakan bentuk tidak normal karena tidak atomik sehingga tidak memenuhi syarat dari 1NF. Untuk mengubah tabel tersebut agar dapat memenuhi syarat 1NF maka setiap kolom harus atomik, beberapa kolom pun harus dipisah agar menghindari anomali.

Tabel 4.9 . Order

Nota	IdBarang	Pelanggan	Qty	Total
B01	K01	Sutris	2	20000
B01	K05	Sutris	1	2000
B01	K06	Sutris	3	45000
B02	K02	Joko	1	8000
B03	K03	Joko	1	12000
B03	K10	Prabski	1	50000

Tabel 4.10. Barang

IdBarang	NamaBarang	Harga
K01	Biskuit	10000
K02	Rinso	8000
K03	Marjan	12000
K04	Milo	13000
K05	Marimas	2000
K06	Nata de Coco	15000
K10	Beras	50000

Tabel 4.9 dan tabel 4.10 telah memenuhi syarat dari 1NF yaitu tidak ada yang duplikat, setiap data pada kolom bersifat atomik dan telah dibagi menjadi dua tabel untuk menjaga integritas data.

Tabel 4.9 merupakan tabel order yang mana memuat barang pembelian. Pada tabel order terdapat dua *primary key* yaitu 'Nota' dan 'IdBarang' serta *foreign key* yaitu 'IdBarang'. Lalu, tabel 4.9 memuat data barang mulai dari 'IdBarang' yang merupakan *primary key*, 'NamaBarang' dan 'Harga'.

2. Bentuk Normal Kedua (2^{nd} Normal Form/2NF)
   Proses berikutnya dari tahapan-tahapan normalisasi adalah bentuk normal kedua atau 2NF. Proses kedua ini merupakan lanjutan dari bentuk normal pertama, syarat dari bentuk normal kedua yang harus terpenuhi:
   a. Sudah berada dalam bentuk 1NF
   b. Semua atribut dalam sebuah tabel harus bergantung pada atribut kunci (primary key)
   c. Tidak ada dependensi parsial (khusus *composite primary key*)

   Langkah-langkah yang harus dijalankan :
   a. Jika ada dependensi parsial maka pecah menjadi tabel yang berbeda

Tabel 4.11. Order (2NF)

Nota	IdBarang	Qty	Total
B01	K01	2	20000
B01	K05	1	2000
B01	K06	3	45000
B02	K02	1	8000
B03	K03	1	12000
B03	K10	1	50000

Tabel 4.12. Nota

Nota	Pelanggan	AlamatPelanggan	TglOrder	TotalBelanja
B01	Sutris	Jalan Nusa	22-Mei-2024	67000
B02	Joko	Jalan Indah	23-Mei-2024	20000
B03	Prabski	Jalan Sehat	24-Mei-2024	50000

Bentuk normal kedua dengan menggunakan contoh tabel yang sebelumnya, maka tabel order mengalami perubahan. Pada tabel 4.11 order atribut 'pelanggan' berpisah dan masuk ke dalam tabel 4.12 tabel nota. Tabel 4.12 merupakan tabel baru yang terbentuk akibat dari proses 2NF, tabel ini berisi 'Nota' yang merupakan *primary key*, lalu ada 'Pelanggan', 'AlamatPelanggan', 'TglOrder', dan 'TotalBelanja'.

3. Bentuk Normal Ketiga (3rd Normal Form/3NF)
   Selanjutnya adalah bentuk normal ketiga dari tahapan proses normalisasi. Seperti sebelumnya, bentuk normal ketiga atau 3NF juga memiliki syarat yang harus dipenuhi, syaratnya adalah sebagai berikut :
   a. Telah memenuhi syarat 2NF
   b. Tidak ada atribut yang dependensi secara transitif pada kolom lain

Langkah :

a. Hapus atau pisahkan menjadi tabel tersendiri untuk atribut yang masih dependen terhadap atribut selain atribut kunci (*primary key*)
b. Memastikan seluruh atribut bukan kunci hanya bergantung kepada atribut kunci

Pada Tabel 4.12 Nota masih ada dependensi transitif yaitu 'Pelanggan' bergantung kepada 'Nota' namun 'AlamatPelanggan' bergantung kepada 'Pelanggan', sehingga 'Pelanggan' dan 'AlamatPelanggan' perlu dipisah menjadi tabel tersendiri.

Tabel 4.13. Nota (3NF)

Nota	IdPelanggan	TglOrder	TotalBelanja
B01	CS001	22-Mei-2024	67000
B02	CS002	23-Mei-2024	20000
B03	CS003	24-Mei-2024	50000

Tabel 4.14. Pelanggan

IdPelanggan	NamaPelanggan	AlamatPelanggan	No.Hp
CS001	Sutris	Jalan Nusa	08214841xxxx
CS002	Joko	Jalan Indah	08892413xxxx
CS003	Prabski	Jalan Sehat	08127282xxxx

Setelah melewati proses 3NF, maka tabel 4.12 Nota mengalami perubahan yang memisahkan atribut 'Pelanggan' dan 'AlamatPelanggan' menjadi tabel tersendiri, lalu pada tabel 4.12 Nota juga terjadi perubahan dimana ada atribut 'IdPelanggan' setelah atribut 'Nota'.

Bentuk normal dari pertama hingga ketiga merupakan tahapan proses normalisas tingkat dasar yang mana sebenarnya telah mengubah struktur basis data menjadi lebih efisien dan

konsisten. Namun, ada tahapan tingkat lanjut yaitu BCNF, 4NF dan 5NF.

4. Bentuk Normal Boyce Codd
   Bentuk normal Boyce Codd (BCNF) merupakan tahap selanjutnya dari proses normalisasi basis data. Syarat dari BCNF ialah :
   a. Telah memenuhi syarat 3NF
   b. Setiap determinan merupakan kunci kandidat

   Sebagai contoh dari BCNF adalah sebagai berikut.

Tabel 4.15. Proyek

Kursus	Instruktur	Ruangan
Basis Data	Miya	R2
PBO	Budi	R1
Web Dasar	Marjono	R3
Web Lanjut	Marjono	R2
Kecerdasan Buatan	Budi	R3

Tabel 4.14 belum memenuhi syarat BCNF karena ada beberapa instruktur yang memiliki ruangan yang berbeda. Dengan mengasumsikan bahwa 'Instruktur' -> 'Ruangan' maka ini tidak memenuhi BCNF karena instruktur bukan kunci kandidat.

Tabel 4.16. Instruktur

Instruktur	Ruangan
Miya	R2
Budi	R1
Marjono	R3

Tabel 4.17. Kursus

Kursus	Instruktur
Basis Data	Miya
PBO	Budi
Web Dasar	Marjono
Web Lanjut	Marjono
Kecerdasan Buatan	Budi

Setelah memisahkan tabel 4.14 menjadi dua tabel, maka syarat dari BCNF telah terpenuhi karena pada tahap ini, 'Instruktur' -> 'Ruangan' setiap instruktur memiliki ruangan yang berbeda karena 'Struktur' merupakan kunci kandidat yang menentukan ruangan. Tabel 6.16 pun telah memenuhi syarat BCNF karena 'Kursus' merupakan kunci kandidat yang menentukan 'Instruktur'.

5. Bentuk Normal Keempat (4th Normal Form/4NF)
   Tahapan selanjutnya dari BCNF adalah tahap bentuk normal keempat atau 4NF. Tahapan ini akan menghilangkan dependensi *multivalued*. Syarat untuk mencapai 4NF adalah
   a. Telah memenuhi syarat BCNF
   b. Tidak ada dependensi *multivalued*

   Contoh dari tabel yang belum memenuhi syarat 4NF ada pada tabel 4.4, lalu perbaikannya ada pada tabel 4.18 dan 4.19 berikut ini.

Tabel 4.18. Kursus dan Materi

Kursus	**Materi**
Matematika	Aljabar
Matematika	Kalkulus

Tabel 4.19. Kursus dan Pengajar

Kursus	**Pengajar**
Matematika	Budi
Matematika	Mita

6. Bentuk Normal Kelima (5th Normal Form/5NF)
   Bentuk normal kelima (5NF) atau Projection-Join Normal Form (PJNF) merupakan proses terakhir dari tahapan normalisasi. Tahapan ini memastikan tidak ada anomali join, yaitu sebuah anomali yang dapat menghilangkan informasi atau menduplikat informasi ketika disatukan kembali.

DAFTAR PUSTAKA

Arenas, M., Barceló, P., Libkin, L., Martens, W., Pieris, A., Paris, S., & Edinburgh, B. (2022). *Database Theory Querying Data*.

Chavan, H., & Shaikh, S. (2022). *Introduction to DBMS Designing and Implementing Databases from Scratch for Absolute Beginners*. www.bpbonline.com

Elmasri, R., & Navathe, S. B. (2016). *Database Systems SEVENTH EDITION*.

Elvis C Foster, S. V. G. (2023). *Database Systems. A Pragmatic Approach (3rd edition)(2023)[EN]* (Vol. 3). https://doi.org/10.1201/9781003275725

Fikry, M. (2019). *BASIS DATA*.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). *DATABASE SYSTEMS The Complete Book Second Edition*.

BAB 5
DATA DEFINITION LANGUAGE AND DATA MANIPULATION LANGUAGE

Oleh Mutia Rahmi Dewi

5.1 Pendahuluan

Bahasa yang mampu menerjemahkan model konseptual menjadi sebuah model fisik *database* adalah *Structured Query Language* (SQL) (Foster and Godbole, 2016). Saat ini, SQL telah tersedia pada *Database Management System* (DBMS) umumnya. SQL membantu pengguna dalam mendefinisikan struktur *database* dan mengelola data yang ada di dalam *database* (Manual, no date). Pada bagian ini akan dibahas 2 jenis bahasa SQL, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

5.2 Data Definition Language

Data Definition Language (DDL) merupakan perintah untuk mendefinisikan struktur di dalam *database* berupa tabel, *view*, *procedure*, dan *index*. Struktur yang didefinisikan berdasarkan pada hasil perancangan (ERD). Pada bagian ini, akan dibahas cara untuk membuat (*create*), mengubah (*alter*), dan menghapus (*drop*) database dan tabel.

5.2.1 DDL untuk Database

DDL untuk *database* sebagai tahap awal dalam menyusun struktur *database*. Untuk *database*, pengguna dapat membuat atau menghapus *database*.

1. Membuat *Database*

```
CREATE DATABASE <namadatabase>;
```

2. Menghapus *Database*

```
DROP <namadatabase>;
```

5.2.2 Tipe Data

Terdapat tipe data yang digunakan dalam mendefinisikan kolom pada tabel, yaitu tipe data numerik, tipe data karakter, dan tipe data tanggal.

1. Tipe Data Numerik
   Tipe data ini akan menampilkan nilai data berupa angka. Contoh : INT, BIGINT, DEC, FLOAT, DOUBLE
2. Tipe Data Karakter
   Tipe data ini akan menampilkan nilai data berupa karakter atau huruf. Contoh : CHAR dan VARCHAR
3. Tipe Data Tanggal dan Waktu
   Tipe data ini akan menampilkan nilai data berupa tanggal dan waktu sesuai format tertentu. Contoh :

Tabel 5.1. Tipe data dan Format

Tipe Data	**Format**
DATETIME	'YYYY-MM-DD HH :MM:SS'
DATE	'YYYY-MM-DD'
TIMESTAMP	'YYYY-MM-DD HH :MM:SS'
TIME	'HH :MM:SS'
YEAR	YYYY

Keterangan :

YYYY = *Year* (tahun) 4
MM = *Month* (bulan)
DD = *Day* (hari)
HH = *Hour* (jam)
MM = *Minute* (menit)
SS = *Second* (detik)

5.2.3 DDL untuk Tabel

Setelah *database* tersedia, maka pengguna dapat mengelola struktur di dalamnya, seperti membuat tabel, mengelola struktur tabel, dan menghapus tabel.

1. Membuat Tabel
   Membuat tabel menggunakan perintah **CREATE TABLE** dengan *syntax*:

```
CREATE TABLE namatabel
(
namakolom1 tipedata(panjang) [aturan],
namakolom2 tipedata(panjang) [aturan],
namakolom3 tipedata(panjang) [aturan],
namakolompk tipedata(panjang) [aturan],
namakolomfk tipedata(panjang) [aturan],
...,
CONSTRAINT <namaconstraint> PRIMARY KEY (namakolompk),
CONSTRAINT <namaconstraint> FOREIGN KEY (namakolomfk)
REFERENCES tabelrujukan(namakolomrujukan)
);
```

Keterangan:

1. Panjang merupakan batasan panjang berupa jumlah karakter atau digit pada tipe data.
2. Aturan bersifat opsional. Terdapat aturan yang bisa digunakan dalam mendefinisikan tiap kolom, diantaranya yaitu:
   a. **DEFAULT**, aturan ini akan mengembalikan nilai kolom yang telah ditetapkan. Contoh: **DEFAULT '2024-01-01'**
   b. **NOT NULL**, nilai data pada kolom tidak boleh kosong
   c. **AUTO_INCREMENT**, aturan ini berlaku untuk kolom dengan tipe data integer. Nilai data pada kolom yang akan dikembalikan berupa angka yang terurut secara otomatis.
3. Jika ada kolom yang berperan sebagai foreign key, maka terlebih dahulu perlu membuat tabel rujukan yang sudah didefinisikan *primary key*.

2. Mengubah Struktur Tabel

   Mengubah struktur tabel dengan menggunakan perintah **ALTER TABLE**. Struktur tabel yang dapat diubah yaitu nama kolom, tipe data kolom, panjang tipe data, aturan kolom, *constraint*, dan nama tabel.

   a. Menambah kolom baru

```
ALTER TABLE namatabel
ADD namakolom tipedata(panjang) [aturan];
```

b. Mengubah struktur kolom

```
ALTER TABLE namatabel
CHANGE namakolomlama namakolombaru tipedata(panjang) [aturan];
```

c. Menghapus kolom

```
ALTER TABLE namatabel
DROP namakolom;
```

d. Menambah constraint baru

Menambah primary key:

```
ALTER TABLE namatabel
ADD CONSTRAINT <namaconstraint> PRIMARY KEY (namakolompk);
```

Menambah foreign key:

```
ALTER TABLE namatabel
ADD CONSTRAINT <namaconstraint> FOREIGN KEY namakolompk REFERENCES tabelrujukan(namakolomrujukan);
```

e. Menghapus constraint

```
ALTER TABEL namatabel
DROP CONSTRAINT namaconstraint;
```

f. Mengubah nama tabel

```
ALTER TABLE namatabel
RENAME namatabelbaru;
```

3. Menghapus Tabel

Menghapus tabel dengan menggunakan perintah **DROP TABLE** dengan *syntax* :

```
DROP TABLE namatabel;
```

5.3 Data Manipulation Language

Data Manipulation Language (DML) merupakan perintah untuk mengelola dan memanipulasi isi data pada tabel di dalam *database*. Pada bagian ini, akan dibahas cara untuk menyisipkan data (*insert*), mengubah data (*update*), menghapus data (*delete*), dan menampilkan data (*select*).

5.3.1 Operator Kondisi pada SQL

Operator ini membantu saat melakukan proses mengubah data, menghapus data, dan menampilkan data. Jika menggunakan operator, maka hanya nilai-nilai data tertentu yang akan diproses sesuai dengan kondisi yang telah ditetapkan. Kondisi didefinisikan dalam klausa **WHERE** pada SQL. Terdapat beberapa jenis operator, yaitu :

1. Operator Perbandingan
   Operator ini berfungsi dalam membandingkan nilai data pada kolom dengan nilai tertentu. Terdapat beberapa notasi di dalam operator perbandingan, yaitu :
   4. Sama dengan (=)
   5. Tidak sama dengan (!= atau <>)
   6. Lebih kecil (<)
   7. Lebih besar (>)
   8. Kecil sama dengan (≤)
   9. Besar sama dengan (≥)

2. Operator Logika
   Operator ini digunakan jika terdapat lebih dari 1 kondisi yang terjadi. Terdapat 2 notasi, yaitu :
   10. **AND**, akan menampilkan data jika semua kondisi yang dipisahkan **AND** dan bernilai benar (TRUE)
   11. **OR**, akan menampilkan data jika salah satu kondisi yang dipisahkan oleh **OR** bernilai benar (TRUE)

3. Operator IN dan NOT IN
   Operator ini digunakan untuk menampilkan data yang termasuk (IN) atau tidak termasuk (NOT IN) dalam sebuah himpunan data. Dengan *syntax* :

   WHERE namakolom **IN/NOT IN** (nilaidata1, nilaidata2, …, nilaidatan) ;

4. Operator LIKE dan NOT LIKE
   Operator ini hanya berlaku untuk tipe data karakter, digunakan untuk melakukan pencarian data berdasarkan karakter tertentu. Dalam memasukkan nilai pencarian dibantu dengan

karakter *underscore* (_) untuk mewakili satu karakter dan persen (%) untuk mewakili semua karakter. Contoh :

'A_ ' : mencari kata yang berawalan A, diikuti oleh 1 karakter. Contoh : 'Ai', 'An', Al'.

'A%' : mencari kata yang berawalan A diikuti oleh karakter dengan jumlah yang tidak terbatas. Contoh : 'Angga', 'Ainun', 'Amanda'.

Syntax yang digunakan sebagai berikut :

WHERE namakolom **LIKE** 'nilai pencarian';

5. Operator BETWEEN dan NOT BETWEEN
   Operator ini digunakan untuk menampilkan data yang berada di dalam jangkauan (BETWEEN) atau di luar jangkauan (NOT BETWEEN) tertentu. Dengan *syntax* :

WHERE namakolom **BETWEEN/NOT BETWEEN** nilaijangkauan1 **AND** nilaijangkauan2;

6. Operator IS NULL
   Operator ini digunakan untuk menampilkan data yang tidak memiliki nilai atau NULL. Dengan menggunakan synatx :

WHERE namakolom **IS NULL**;

5.3.2 Menyisipkan Data

Menyisipkan data dilakukan dengan menggunakan perintah **INSERT** untuk menambahkan data baru ke dalam tabel tertentu. Perintah **INSERT** dapat dilakukan untuk menambahkan 1 atau lebih data ke dalam semua atau beberapa kolom saja. Dengan menggunakan *syntax* :

INSERT INTO namatabel(namakolom1, namakolom2, namakolom3, namakolomn) **VALUES** (nilaidata1, nilaidata2, nilaidata3, …, nilaidatan), (nilaidata1, nilaidata2, nilaidata3, …, nilaidatan), (nilaidata1, nilaidata2, nilaidata3, …, nilaidatan) ;

5.3.3 Mengubah Data

Mengubah data dilakukan dengan menggunakan perintah **UPDATE** untuk mengubah nilai data pada kolom tertentu dengan *syntax* :

```
UPDATE namatabel
SET namakolom = nilaidatabaru
[WHERE kondisi];
```

5.3.4 Menghapus Data

Menghapus data dilakukan dengan menggunakan perintah **DELETE** untuk menghapus data pada tabel berdasarkan kondisi tertentu dengan *syntax* :

```
DELETE FROM namatabel
[WHERE kondisi];
```

5.3.5 Menampilkan Data

Terdapat beberapa perintah yang membantu dalam menampilkan data, diantaranya yaitu :

1. **SELECT,** digunakan untuk memanggil data dari kolom tertentu
2. **FROM,** digunakan untuk menyebutkan tabel sumber tempat kolom berasal
3. **WHERE,** digunakan untuk mendefinisikan kondisi sesuai dengan permintaan
4. **ORDER BY,** digunakan untuk mengurutkan data yang ditampilkan. Bisa dimulai dari data pertama (*ascending*) dengan menambahkan keterangan **ASC** atau dari data terakhir (*descending*) dengan menambahkan keterangan **DESC**.
5. **LIMIT,** digunakan untuk membatasi jumlah data yang ingin ditampilkan.

Query untuk menampilkan data bisa berasal dari 1 tabel atau lebih.

1. Menampilkan data pada satu tabel

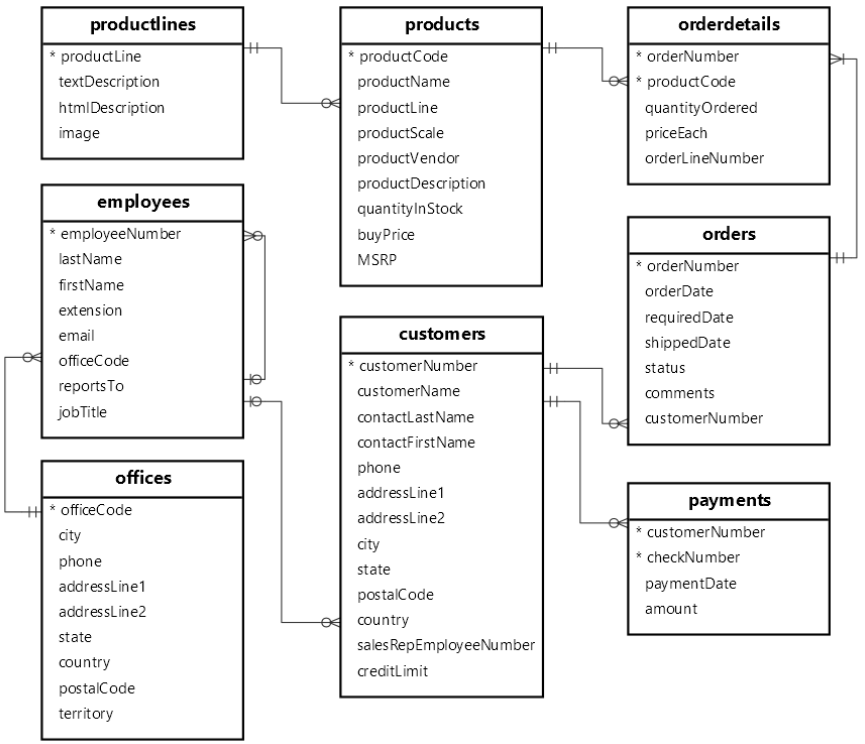
Dengan menggunakan *syntax* :

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel
```

[**WHERE** kondisi]
[**ORDER BY** namakolom **ASC/DESC**]
[**LIMIT** jumlahbaris] ;

Keterangan : perintah di dalam kurung siku ([ ]) bersifat opsional

2. Menampilkan data lebih dari satu tabel
   Menampilkan data lebih dari satu tabel dilakukan dengan menggunakan perintah **JOIN**. Perintah **JOIN** mampu mengolah data yang bersumber dari banyak tabel melalui kolom yang berelasi atau disebut sebagai *foreign key*. Syarat dalam menjalani perintah **JOIN** adalah adanya relasi antar tabel. Jika tidak ada relasi secara langsung, maka perlu mencari tabel yang menjembatani relasi tersebut. Perhatikan skema *database* classicmodels pada Gambar 7.1 berikut.

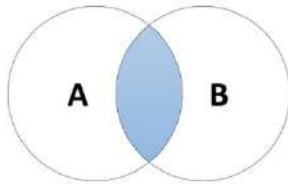


Gambar 5.1. Skema *database* classicmodels
(Sumber : MySQLTUTORIAL, 2017)

Jika ingin menampilkan data dari tabel products dan orderdetails, maka diperlukan kolom productcode sebagai relasinya. Tetapi jika ingin menampilkan data yang berasal dari tabel products dan customers, maka perlu mencari tabel yang menjembatani 2 tabel tersebut. Berdasarkan skema pada Gambar 5.1, tabel orderdetails dan orders menjadi tabel yang menjembatani tabel products dan customers. Setelah itu perlu menemukan kolom-kolom relasi antara tabel products, orderdetails, orders, dan customers. Berdasarkan Gambar 5.1, yang menjadi kolom relasi adalah productcode yang menghubungkan tabel products dan orderdetails, ordernumber yang menghubungkan tabel orderdetails dan orders, dan customernumber yang menghubungkan tabel orders dan customers. Dengan demikian, perintah **JOIN** dapat dijalankan.

Terdapat beberapa jenis **JOIN,** diantaranya yaitu :

1. INNER JOIN yang diilustrasikan pada Gambar 5.2, akan mengembalikan nilai data yang beririsan antar 2 tabel.



Gambar 5.2. Ilustrasi INNER JOIN

Operasi INNER JOIN dilakukan dengan menggunakan beberapa *syntax* sebagai berikut:

Syntax 1:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1, namatabel2
WHERE namatabel1.kolomrelasi = namatabel2.kolomrelasi;
```

Syntax 2:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1 JOIN namatabel2
ON namatabel1. kolomrelasi = namatabel2. kolomrelasi;
```

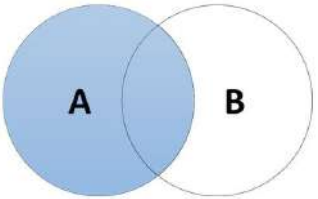
Syntax 3:

```
SELECT namakolom1, namakolom2, …, namakolomn
```

```
FROM namatabel1 JOIN namatabel2 USING (kolomrelasi) ;
```

Keterangan: pada *syntax* 3, perlu dipastikan tabel 1 dan tabel 2 memiliki nama kolom relasi yang sama.

2. LEFT OUTER JOIN yang diilustrasikan pada Gambar 5.3, akan mengembalikan seluruh baris dari tabel disebelah kiri yang dikenai kondisi **ON** dan hanya baris dari tabel disebelah kanan yang memenuhi kondisi *join*.

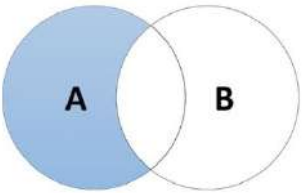


Gambar 5.3. Ilustrasi LEFT OUTER JOIN

Operasi LEFT OUTER JOIN dilakukan dengan menggunakan *syntax*:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1 LEFT JOIN namatabel2
ON namatabel1.kolomrelasi = namatabel2.kolomrelasi;
```

3. LEFT OUTER JOIN WITHOUT INTERSECTION yang diilustrasikan pada Gambar 5.4, merupakan variasi dari *left outer join*. Pada join ini akan mengambil data dari tabel sebelah kiri yang dikenai kondisi **ON** yang juga memenuhi kondisi *join* tanpa data dari tabel sebelah kanan yang memenuhi kondisi *join*.

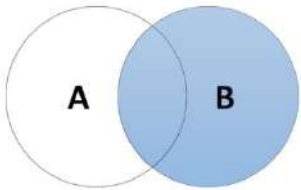


Gambar 5.4. Ilustrasi LEFT OUTER JOIN WITHOUT INTERSECTION

Operasi LEFT OUTER JOIN WITHOUT INTERSECTION dilakukan dengan menggunakan *syntax*:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1 LEFT JOIN namatabel2
ON namatabel1.kolomrelasi = namatabel2.kolomrelasi
WHERE namatabel2.kolomrelasi IS NULL;
```

4. RIGHT OUTER JOIN yang diilustrasikan pada Gambar 5.5, akan mengembalikan semua baris dari tabel sebelah kanan yang dikenai kondisi **ON** dengan data dari tabel sebelah kiri yang memenuhi kondisi *join*.

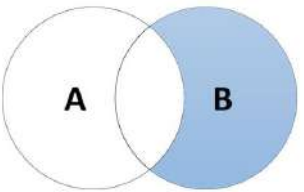


Gambar 5.5. Ilustrasi RIGHT OUTER JOIN

Operasi RIGHT OUTER JOIN dilakukan dengan menggunakan *syntax*:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1 RIGHT JOIN namatabel2
ON namatabel1.kolomrelasi = namatabel2.kolomrelasi;
```

5. RIGHT OUTER JOIN WITHOUT INTERSECTION yang diilustrasikan pada Gambar 5.6, merupakan variasi dari *right outer join*. Pada *join* ini akan mengambil data dari tabel sebelah kanan yang dikenai kondisi **ON** yang juga memenuhi kondisi *join* tanpa data dari tabel sebelah *kanan* yang memenuhi kondisi *join*.



Gambar 5.6. Ilustrasi RIGHT OUTER JOIN WITHOUT INTERSECTION

Operasi LEFT OUTER JOIN WITHOUT INTERSECTION dilakukan dengan menggunakan *syntax*:

```
SELECT namakolom1, namakolom2, …, namakolomn
FROM namatabel1 RIGHT JOIN namatabel2
ON namatabel1.kolomrelasi = namatabel2.kolomrelasi
WHERE namatabel1.kolomrelasi IS NULL;
```

DAFTAR PUSTAKA

Foster, E. C. and Godbole, S. (2016) *Database systems: A pragmatic approach*, *Database Systems: A Pragmatic Approach*. doi: 10.1007/978-1-4842-1191-5.

Manual, M. R. (no date) 'MySQL 8.0 Reference Manual'.

MySQLTUTORIAL (2017) 'MySQL Sample Database'. Available at: https://www.mysqltutorial.org/mysql-sample-database.aspx%0Ahttp://www.mysqltutorial.org/mysql-sample-database.aspx.

BAB 6
DATABASE PENJUALAN

Oleh Lutfi Syafirullah

6.1 Pendahuluan

Perkembangan teknologi khususnya dalam bidang e-commerce, telah mengubah lanskap perdagangan secara signifikan. Ini memberikan berbagai manfaat bagi pelaku usaha dan konsumen. Beberapa manfaat yang telah Anda sebutkan seperti peningkatan penjualan, efisiensi waktu dan tenaga, serta penghematan biaya memang merupakan dampak positif yang signifikan. Pemanfaatan e-commerce memungkinkan pelanggan untuk berbelanja secara online melalui jaringan internet tanpa harus dating secara langsung ke toko fisik. Ini memberi kemudahan dan kenyamanan bagi konsumen, yang pada gilirannya dapat meningkatkan volume penjualan. Selain itu, platform e-commerce juga memungkinkan penargetan pasar yang lebih luas, karena pelanggan dari berbagai daerah atau negara dapat dijangkau dengan lebih mudah.

Tidak hanya itu, sistem e-commerce juga dapat membantu dalam pengelolaan inventaris, pemrosesan pembayaran, dan analisis data penjualan. Sistem e-commerce mempercepat proses bisnis secara keseluruhan untuk dapat melayani kebutuhan pelanggan secara cepat dan efisien. Seperti halnya dengan semua teknologi, sistem e-commerce memiliki permasalahan yang perlu diperhatikan dan diselesaikan seperti keamanan data, persaingan yang ketat, dan perubahan perilaku konsumen. Namun, dengan pengelolaan yang tepat, manfaat e-commerce dapat dioptimalkan untuk meningkatkan kinerja bisnis dan memberikan nilai tambah bagi pelanggan.

Studi kasus pengembangan database pada system informasi penjualan berbasis website ini dilakukan pada toko Alvita Collection.

Tujuannya mengembangkan sistem penjualan ini adalah untuk memasarkan produk secara luas yang memberikan keuntungan bagi pemilik bisnis maupun konsumen.
Manfaat dari sistem penjualan ini adalah :

1. **Membantu Pemilik dalam Promosi dan Pemasaran.**
   Pemilik dengan mudah mempromosikan produk mereka kepada khalayak yang lebih luas, tanpa terbatas oleh batasan geografis dengan berbagai strategi pemasaran online untuk menjangkau lebih banyak pelanggan potensial. Selain itu, pemilik juga dapat dengan mudah melacak dan menganalisis data penjualan untuk memahami tren pasar dan menyesuaikan strategi pemasaran mereka.
2. **Mempermudah Pelanggan dalam Pembelian Online.**
   Pelanggan dengan mudah dapat melakukan pembelian produk secara nyaman. Mereka dapat menjelajahi katalog produk, membaca deskripsi produk, melihat ulasan dari pelanggan lain, dan melakukan pembayaran secara online. Ini memberikan pengalaman belanja yang lebih praktis dan efisien bagi pelanggan.
3. **Membantu Kasir dalam Pencatatan Transaksi.**
   Dengan sistem ini pencatatan yang dilakukan oleh kasir atau petugas penjualan yang sebelumnya dikerjakan secara manual saat ini dapat dikerjakan terotomatisasi, data transaksi dapat dicatat secara rapih dan akurat. Ini membantu mengurangi kesalahan manusia dalam pencatatan data dan memastikan keamanan data transaksi yang tersimpan. Dengan demikian, proses operasional menjadi lebih efisien dan efektif.

Dengan memanfaatkan sistem informasi penjualan berbasis website, baik pemilik bisnis, pelanggan, maupun kasir dapat merasakan manfaat yang signifikan dalam menjalankan dan mengelola bisnis mereka.

6.2 Landasan Teori

6.2.1 Sistem informasi

Sistem dapat didefinisikan sebagai kumpulan beberapa elemen yang saling bekerjasama satu dengan yang lain dalam mencapai suatu tujuan tertentu. Masing-masing elemen dalam sistem berkontribusi terhadap keseluruhan tujuan sistem tersebut. Interaksi antara elemen-elemen ini memungkinkan sistem untuk beroperasi dan mencapai tujuan yang ditetapkan (Mei Prabowo, 2019).

Sistem secara umum terdiri dari masukkan (**input**), pengelolaan (*process*), keluaran (*output*). Terdapat beberapa karakteristik yang dapat mencirikan sebuah sistem, dan keberadaan karakteristik-karakteristik ini menunjukkan bahwa suatu entitas dapat dianggap sebagai sistem. Beberapa karakteristik tersebut antara lain (Tata Sutabri, 2012):

1. Komponen Sistem (*Components*).
   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan.
2. Batasan Sistem (*Boundary*). Batasan Sistem digunakan sebagai satu kesatuan yang tidak dapat dipisahkan. Ruang lingkupnya membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkup luar.
3. Lingkup Luar Sistem (*Environtment*). Lingkup luar sistem adalah seluruh bentuk yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem.
4. Penghubung Sistem (*Interface*). Penghubung sistem ialah media yang menghubungkan sistem dengan subsistem lainnya.
5. Masukkan Sistem (*Input*). Masukkan sistem adalah energi yang dimasukkan dalam sistem, dapat berupa sinyal (signal input) dan pemeliharaan (maintenance input).
6. Keluaran Sistem (*Output*). Keluaran sistem adalah hasil energi yang telah diolah dan diklasifikasikan menjadi keluaran yang berguna.
7. Pengolahan Sistem (*Procces*). Suatu proses yang akan mengubah sebuah masukkan menjadi keluaran.

8. Sasaran Sistem (*Objective*). Sasaran sistem ialah apa yang dimiliki oleh sistem yang pasti dan bersifat deterministik.

Informasi adalah pesan atau makna yang diterima dari sebuah proses komunikasi. Dari pengertian ini menjelaskan bahwa informasi merupakan data yang telah diolah sehingga berguna dan dapaat dimanfaatkan bagi penggunanya dalam menentukan sebuah keputusan. Kualitas informasi menurut Teguh Wahyudi, kualitas informasi dinilai dari tiga hal, yaitu (Agus Rusmana, 2014):

1. Relevansi.
   Kualitas informasi sering kali dinilai berdasarkan relevansinya, seberapa baik informasi tersebut membantu secara signifikan bagi pengguna dalam menarik sebuah kesimpulan atau membuat sebuah keputusan. Penting untuk diingat bahwa tingkat relevansi informasi dapat bervariasi antara individu, tergantung pada peran dan kebutuhan mereka dalam suatu konteks tertentu.
2. Akurasi.
   Salah satu karakteristik penting dari informasi yang berkualitas. Informasi dikatakan akurat jika tidak terdapat bias atau penyesatan, dan maksudnya jelas dipahami oleh pemakainya. Informasi yang lengkap adalah informasi yang menyajikan gambaran menyeluruh dan komprehensif tentang suatu topik atau masalah. Ketika informasi terpecah-pecah atau tidak lengkap, ini dapat menghambat kemampuan seseorang untuk membuat keputusan yang tepat atau memecahkan masalah dengan efektif. Sebaliknya, informasi yang lengkap memberikan pemakai pemahaman yang lebih baik tentang situasi atau masalah yang dihadapi dengan tepat. Kebenaran informasi sering kali dapat dinilai dari kesesuaiannya dengan perhitungan-perhitungan atau metodologi yang digunakan dalam proses pembuatannya. Proses pembuatan informasi yang baik seringkali melibatkan langkah-langkah seperti pengumpulan data, analisis, interpretasi, dan verifikasi.

3. Tepat waktu.
   Informasi yang berkualitas sering kali diukur oleh kemampuannya untuk diciptakan dalam waktu yang tepat, sehingga mampu memainkan peran penting dalam menentukan nilai atau harga sebuah informasi. Informasi yang tersedia secara tepat waktu dapat memberikan nilai tambah yang signifikan, karena memungkinkan para pengambil keputusan untuk merespons situasi atau peluang dengan cepat. Sebaliknya, informasi yang tidak tersedia dalam waktu yang cukup dapat mengurangi nilai atau relevansinya dalam konteks tertentu. Oleh karena itu, dalam merancang sistem informasi atau proses pengolahan data, penting untuk mempertimbangkan kecepatan dan ketepatan waktu agar informasi yang dihasilkan dapat digunakan untuk membuat keputusan yang tepat dalam waktu yang sesuai.

Sistem Informasi adalah suatu cara tertentu untuk menyediakan informasi yang dibutuhkan oleh organisasi untuk beroperasi dengan cara yang sukses, dan untuk organisasi bisnis dengan cara yang menguntungkan(Agus Rusmana, 2014).

6.2.2 Penjualan

Penjualan adalah proses menjual barang atau jasa kepada pelanggan dengan tujuan mendapatkan keuntungan. Ini melibatkan serangkaian kegiatan, strategi, dan interaksi antara penjual dan pembeli untuk mencapai kesepakatan transaksi yang saling menguntungkan. Dengan memahami konsep dan definisi terkait dengan penjualan, perusahaan dapat mengembangkan strategi penjualan yang efektif, membangun hubungan yang baik dengan pelanggan, dan mencapai kesuksesan dalam aktivitas penjualan.

6.2.3 Basis Data

Basis data dapat didefinisikan dan dimaknai sebagai suatu kumpulan data yang saling berhubungan dan terorganisir dengan baik, yang disimpan secara terpusat dan dapat diakses, dikelola, serta diperbarui dengan efisien. Data dalam basis data merujuk

pada fakta-fakta mengenai objek, orang, atau entitas lainnya yang relevan dengan konteks yang diinginkan. Data dapat dinyatakan dalam berbagai bentuk, termasuk nilai numerik, urutan karakter, atau simbol-simbol lainnya(Hendra Jatnika, 2013).

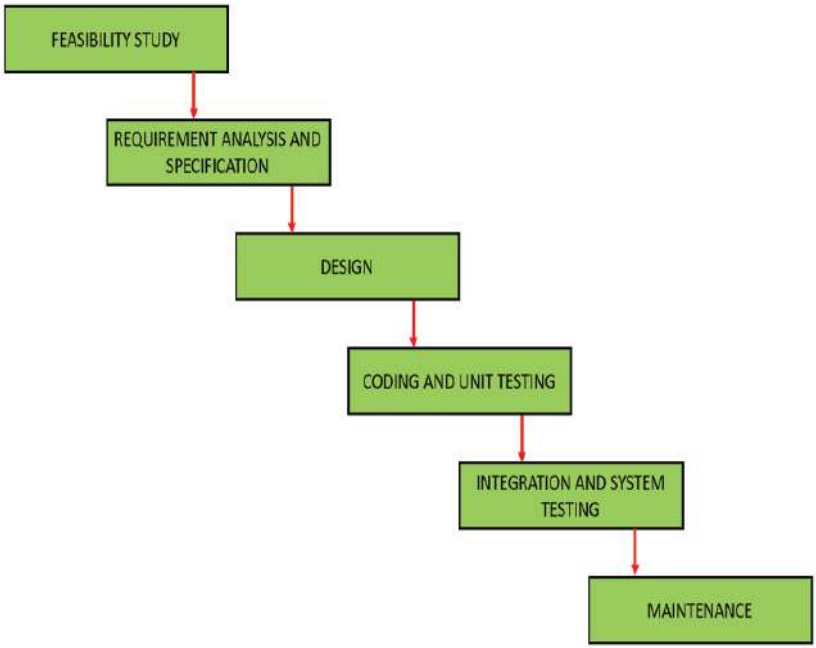
Perancangan basis data melibatkan dua aspek utama: perancangan secara fisik dan perancangan secara logis. Perancangan logis berkaitan dengan desain struktur data, relasi antar entitas, serta model data yang digunakan dalam basis data. Ini meliputi pemodelan data dengan menggunakan diagram ER (Entity-Relationship), penentuan atribut-atribut yang diperlukan untuk setiap entitas, serta pemetaan hubungan antar entitas. Perancangan fisik berkaitan dengan implementasi perancangan logis mencakup pembuatan tabel, indeks, relasi antar tabel, dan aspek teknis lainnya yang diperlukan untuk penyimpanan dan manipulasi data (Ridwan Setiawan, 2018).

6.2.4 Pengembangan Perangkat Lunak

Metode yang digunakan dalam mengembangkan Sistem Informasi Penjualan Berbasis Website adalah metode waterfall.

SDLC (*Software Development Life Cycle*) atau Siklus Hidup Pengembangan Sistem adalah pendekatan sistematis untuk merencanakan, merancang, mengembangkan, dan memelihara perangkat lunak atau sistem informasi. Konsep ini mencakup serangkaian tahapan atau fase yang dijalani selama proses pengembangan. Salah satu metode dalam SDCL yang sering dijumpai adalah Waterfall (Mei Prabowo, 2019).

Proses Pengembangan Sistem Informasi Penjualan Berbasis Website pada Alvita Collection yang akan dibangun menggunakan metode waterfall. Model ini dipilih karena sistem yang akan dibangun menggunakan tahapan metode yang runtun atau tersusun sistematis.



Gambar 6.1. Classical Waterfall
(Sumber: Metodologi Pengembangan Sistem Informasi, Mei Prabowo, M.Com.)

Tahapan-tahapan dalam pendekatan Waterfall:

1. Studi kelayakan (*Fasibility Study*)
   Studi kelayakan perangkat lunak atau system memiliki tujuan utama yaitu untuk menentukan apakah proyek tersebut layak secara finansial, teknis, dan operasional sebelum melakukan investasi yang signifikan dalam pengembangan.
2. Analisis dan spesifikasi kebutuhan (*Requirement Analysis and Specification*)
   Tahap ini memiliki tujuan untuk dapat memahami pengguna sistem, serta mendokumentasikannya dengan benar. Proses ini sangat penting karena keberhasilan proyek pengembangan perangkat lunak atau sistem seringkali

bergantung pada pemahaman yang jelas dan lengkap terhadap kebutuhan pengguna.

3. Desain (*Design*)
   Tujuan dari tahap desain adalah untuk menetapkan software requirements specification atau kebutuhan system spesifikasi persyaratan perangkat lunak dimana pada tahap ini dilakukan perencanaan arsitektur sistem yang akan dibangun serta rancangan detail dari komponen-komponen sistem tersebut.
4. *Coding and Unit Testing*
   Pada tahap ini dilakukan implementasi bahasa pemrograman untuk menterjemahkan desain perangkat lunak dengan menulis kode program yang akan menjadi inti dari sistem yang akan dibangun.
5. *Integration and System Testing*
   Integrasi berbagai modul dalam pengembangan perangkat lunak dilakukan setelah proses pengkodean, di mana setiap unit modul diuji secara individu. Integrasi modul adalah tahap di mana modul-modul yang telah selesai dikodekan dan diuji individu digabungkan menjadi satu kesatuan yang utuh, sehingga membentuk sistem yang lengkap.
6. Perawatan (*Maintenance*)
   Perawatan atau pemeliharaan merupakan tahap yang sangat penting untuk memastikan bahwa perangkat lunak tetap berfungsi dengan baik dan efisien. Perawatan perangkat lunak dapat dilakukan ketika terjadi kerusakan atau kesalahan pada perangkat lunak yang telah berjalan, serta ketika perubahan atau peningkatan perangkat lunak diminta oleh pelanggan atau pengguna.

Meskipun Waterfall memiliki keunggulan-keunggulan, penting untuk diingat bahwa model ini juga memiliki kelemahan, seperti kurangnya fleksibilitas terhadap perubahan kebutuhan, sulitnya kembali ke tahap sebelumnya jika ditemukan kesalahan, dan risiko penundaan dalam pengembangan. Oleh karena itu, pemilihan model pengembangan perangkat lunak harus dipertimbangkan dengan cermat sesuai dengan kebutuhan dan

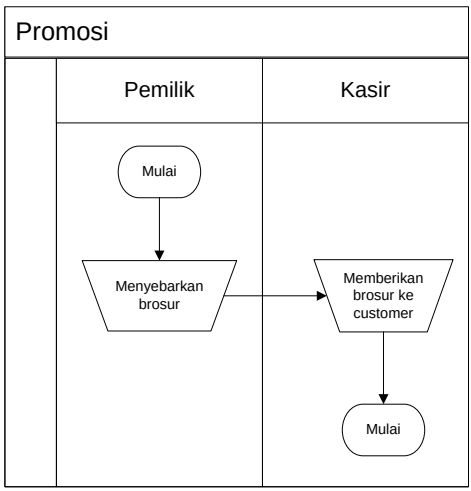
karakteristik proyek yang bersangkutan. Penting untuk diingat bahwa Waterfall tidak selalu cocok untuk setiap proyek, terutama proyek-proyek yang membutuhkan fleksibilitas tinggi terhadap perubahan kebutuhan atau iterasi desain. Dengan memahami kelemahan-kelemahan ini, organisasi pengembangan perangkat lunak dapat mempertimbangkan alternatif model pengembangan yang lebih cocok untuk kebutuhan proyek, yang lebih fleksibel dan responsif terhadap perubahan (Fitria Nur Hasanah, 2020).

6.3 Perancangan Sistem

6.3.1 Analisa Sistem yang Sedang Berjalan

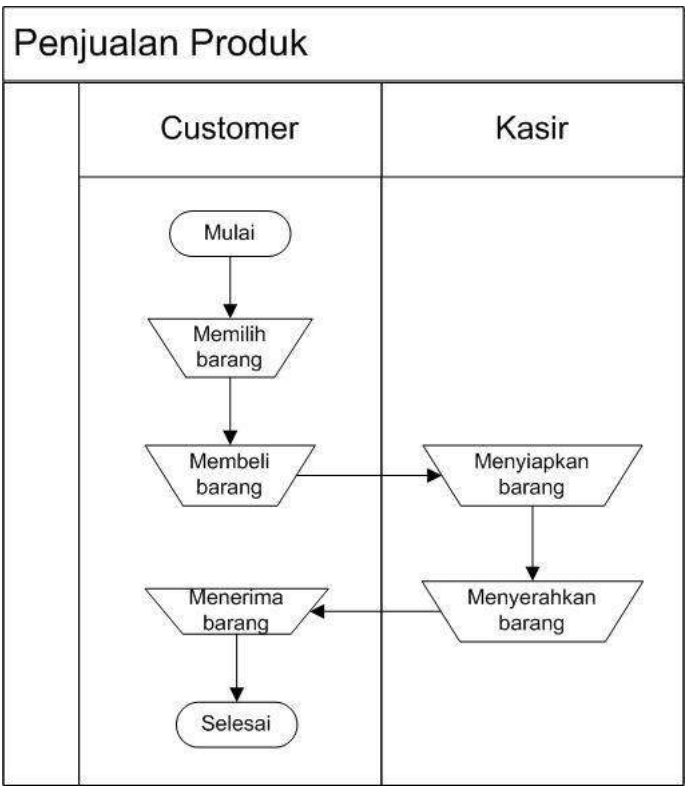
Analisis sistem yang sedang berjalan di Alvita Collection secara umum digambarkan dengan suatu flowchart proses penjualan produk yang terdiri dari 3 proses yaitu proses promosi produk, penjualan produk, pembayaran produk.

1. Flowchart Proses Promosi
   Pada Gambar 6.2 dijelaskan proses promosi yang dilakukan di Alvita Collection. Flowchart dimulai dari pemilik toko menyebarkan brosur promosi ke calon pembeli. Penyebaran brosur promosi berguna untuk menarik calon pembeli datang ke toko dan membeli barang. Kemudian karyawan/kasir akan memberikan promosi tersebut pada pelanggan.



Gambar 6.2. Flowchart Promosi

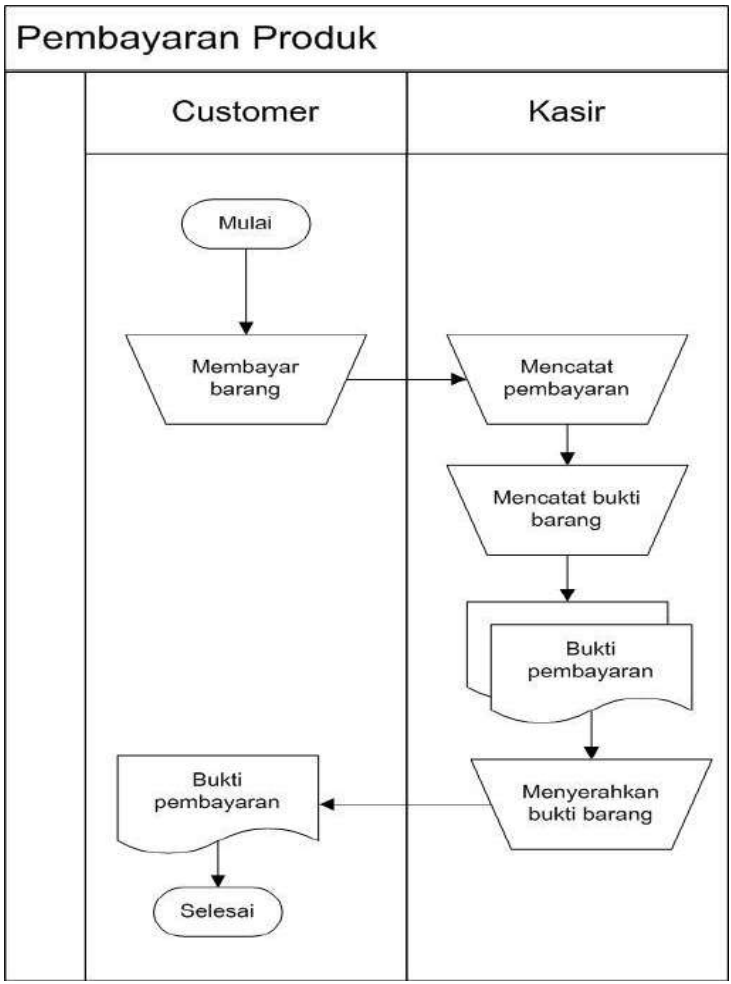
2. Flowchart Proses Penjualan Produk
   Pada Gambar 6.3 dijelaskan sebuah proses penjualan yang dilakukan di Alvita Collection. Dimulai dari pelanggan memilih barang mana yang akan dibeli. Setelah memilih barang, pelanggan baru memutuskan untuk melakukan pembelian dan menyerahkan barang yang akan dibeli pada kasir toko. Selanjutnya kasir akan menyiapkan barang tersebut untuk dikemas dan melakukan perhitungan total bayar pelanggan. Kemudian kasir menyerahkan barang tersebut kepada pelanggan dan proses penjualan selesai.



Gambar 6.3. Flowchart Proses Penjualan

3. Flowchart Pembayaran Produk
   Pada Gambar 6.4 dijelaskan sebuah proses pembayaran produk yang dilakukan di Alvita Collection. Dimulai dari

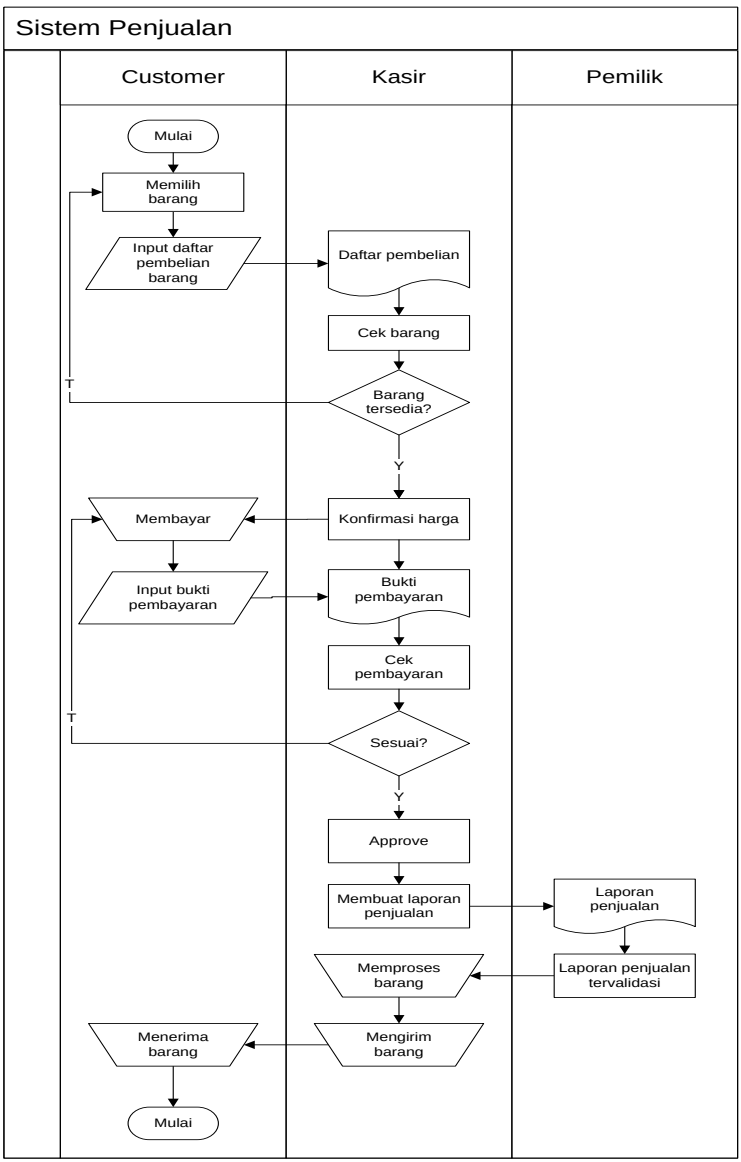
pelanggan yang sudah selesai memilih barang akan melakukan pembayarannya. Barang yang telah dipilih diserahkan pada kasir untuk dicatat pada buku transaksi dan pada nota transaksi rangkap dua, yang mana rangkap nota akan dimasukan pada arsip. Bukti pembayaran atau nota kemudian diserahkan pada pelanggan dan pelanggan melakukan pembayaran. Setelah proses pembayaran selesai pelanggan menerima barang yang dibelinya, dan proses pun telah selesai.



Gambar 6.4. Flowchart Pembayaran Produk

6.3.2 Analisa Sistem yang Akan Dikembangkan

Berikut gambaran sistem yang akan dikembangkan di Alvita Collection beserta penjelasan terkait proses penjualan produk yang akan dikembangkan.



Gambar 6.5. Flowchart Penjualan Produk

Proses penjualan produk pada Alvita Collection yang akan dikembangkan dimulai dari pelanggan memilih barang yang akan dibeli, kemudian melakukan input barang pada keranjang belanjanya. Langkah selanjutnya ialah pengecekan stok barang, apabila barang tersedia maka akan muncul konfirmasi biaya pembelian. Jika barang tidak tersedia maka proses akan kembali ke pemilihan barang. Setelah konfirmasi biaya, pelanggan harus membayar barangnya sesuai dengan total yang muncul, dan pelanggan harus mengirimkan bukti pembayaran yang telah dilakukannya. Kemudian bukti pembayaran akan dicek, bila bukti pembayaran belum valid maka pelanggan diminta melakukan pembayarannya ulang, karena bisa jadi pelanggan belum melakukan pembayarannya atau ada kesalahan lainnya. Apabila sudah sesuai, maka prosesnya berlanjut ke proses pengiriman barang. Proses akan berakhir apabila barangnya sudah sampai dengan selamat kepada pelanggan.

6.3.3 Pengguna Sistem

Aktor – aktor yang terlibat dalam penggunaan sistem, maka aktor memiliki beberapa fungsi utama sesuai dengan kebutuhan masing – masing pengguna dijelaskan dalam tabel.

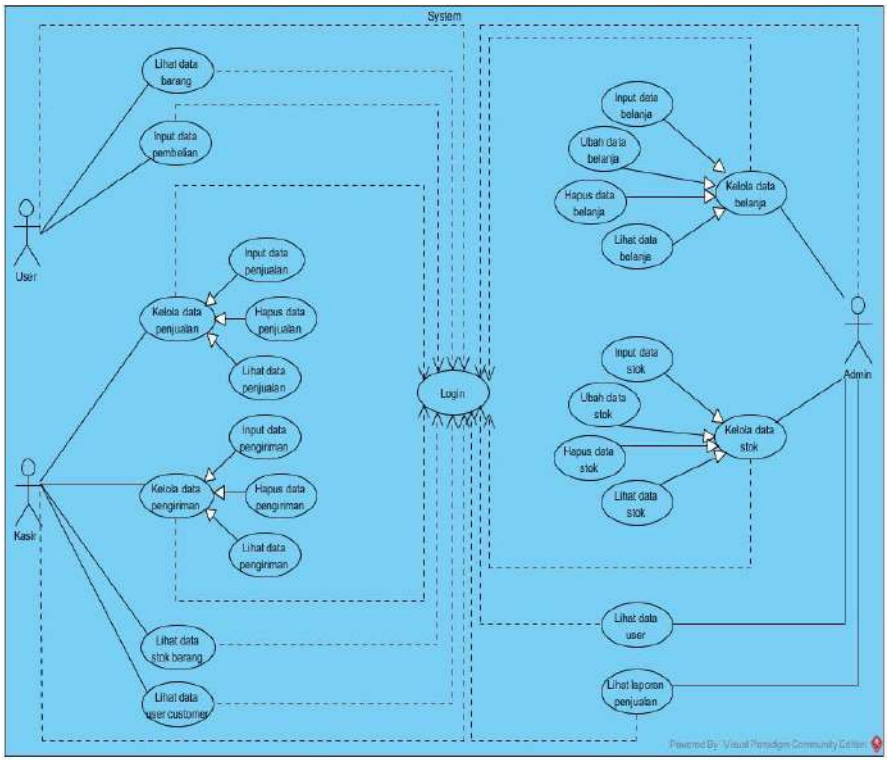
Tabel 6.1. Analisis Kebutuhan Pengguna

No	Aktor	Kebutuhan
1.	Admin	1. Login pada tampilan admin 2. Input data pembelanjaan 3. Edit data pembelanjaan 4. Hapus data pembelanjaan 5. Melihat data pembelanjaan 6. Input data stok barang 7. Edit data stok barang 8. Hapus data stok barang 9. Melihat data stok barang 10. Melihat data pelanggan

No	Aktor	Kebutuhan
		11. Melihat data penjualan
2.	Customer	1. Login pada tampilan pelanggan 2. Input data pembelian 3. Melihat data barang
3.	Kasir	1. Login pada tampilan kasir 2. Input data penjualan 3. Hapus data penjualan 4. Melihat data penjualan 5. Melihat data stok barang 6. Input data pengiriman 7. Hapus data pengiriman 8. Melihat data pengiriman 9. Melihat data pelanggan

Pemilik (Admin) memiliki tanggung jawab penuh terhadap toko, yaitu tidak hanya melakukan pengawasan akan tetapi juga melakukan pendataan stok barang. Adapun Kasir/Cashier (Pegawai) memiliki tanggung jawab hanya sebatas melakukan perhitungan transaksi. Sedangkan pelanggan dapat mengakses website untuk melihat, memilih, dan membeli produk.

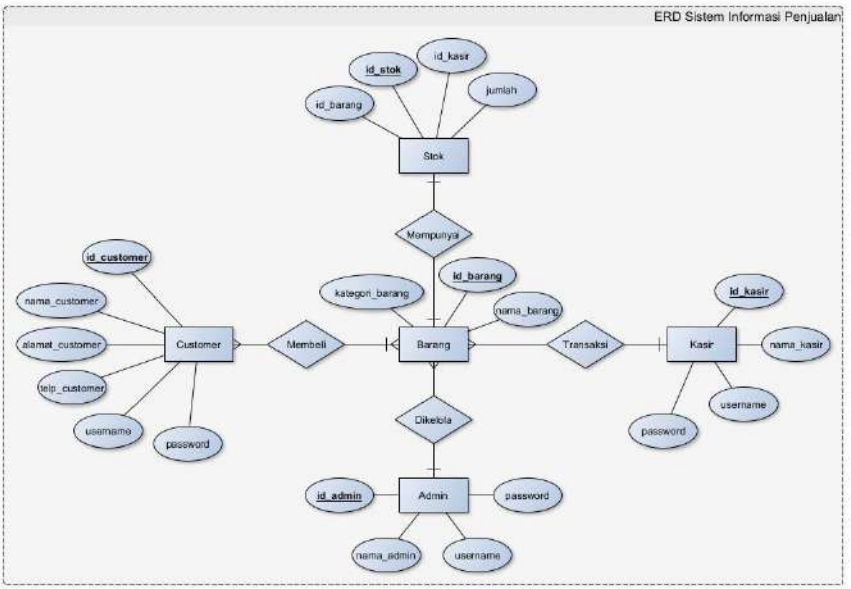
6.3.4 Use Case Diagram



Gambar 6.6. Usecase Sistem Informasi Penjualan Berbasis Website pada Alvita Collection

Pada Gambar 6.6 Usecase Sistem Informasi Penjualan Berbasis Website pada Alvita Collection, dijelaskan bahwa terdapat tiga aktor pada sistem yaitu admin/pemilik, kasir, dan pelanggan. Ketiga level tersebut diwajibkan login sebelum masuk ke sistem, kecuali pelanggan yang hanya login apabila melakukan transaksi. Pelanggan hanya login saat transaksi agar data yang sudah pernah diisi tersimpan sehingga saat repeat order tidak perlu lagi mengisikan data – datanya.

6.3.5 ERD (*Entity Relationship Diagram*)



Gambar 6.7. ERD Sistem Informasi Penjualan Berbasis Website pada Alvita Collection

Sistem memiliki beberapa entitas yang saling berhubungan, yang mana admin adalah pengguna yang melakukan pengelolaan barang dan stok. Sedangkan kasir melakukan penjualan dan pengiriman barang. Pelanggan pada ERD dijelaskan sebagai aktor yang membeli barang melalui proses transaksi. Berikut struktur dari masing-masing tabel.

Tabel 6.2. Tabel User

No	Nama Field	Tipe Data	Panjang	Keterangan
1	id_user	int	5	Primary Key
2	nama	varchar	25	
3	email	varchar	25	
4	password	varchar	10	
5	foto	varchar	25	
6	telp	varchar	13	

No	Nama Field	Tipe Data	Panjang	Keterangan
7	alamat	varchar	100	
8	level	int	10	
9	date_create	date	-	
10	active	int	1	

Tabel 6.3. Tabel Barang

No	Nama Field	Tipe Data	Panjang	Keterangan
1	id_barang	varchar	5	Primary Key
2	nama_barang	varchar	25	
3	stok_barang	int	10	
4	harga_barang	int	25	
5	kategori_barang	varchar	25	
6	gambar_barang	varchar	25	
7	detail_barang	varchar	100	

Tabel 6.4. Tabel Stok

No	Nama Field	Tipe Data	Panjang	Keterangan
1	id_stok	int	10	Primary Key
2	id_transaksi	int	10	Foreign Key
3	id_barang	int	10	Foreign Key
4	jumlah	int	-	

Tabel 6.5. Tabel Penjualan

No	Nama Field	Tipe Data	Panjang	Keterangan
1	id_penjualan	varchar	5	Primary Key
2	id_transaksi	int	5	Foreign Key
3	nama_user	varchar	25	
4	tgl_penjualan	date	-	
5	tgl_penjualan	int	-	

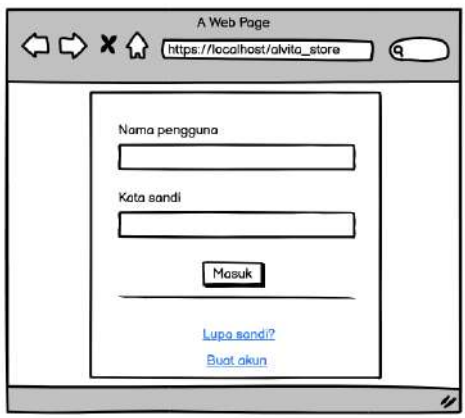
Tabel 6.6. Tabel transaksi

No	Nama Field	Tipe Data	Panjang	Keterangan
1	id_transaksi	int	5	Primary Key
2	id_kasir	int	5	Foreign Key
3	id_barang	int	5	Foreign Key
4	jml_penjualan	int	10	
5	harga_penjualan	int	-	

6.3.6 Rancangan Antarmuka

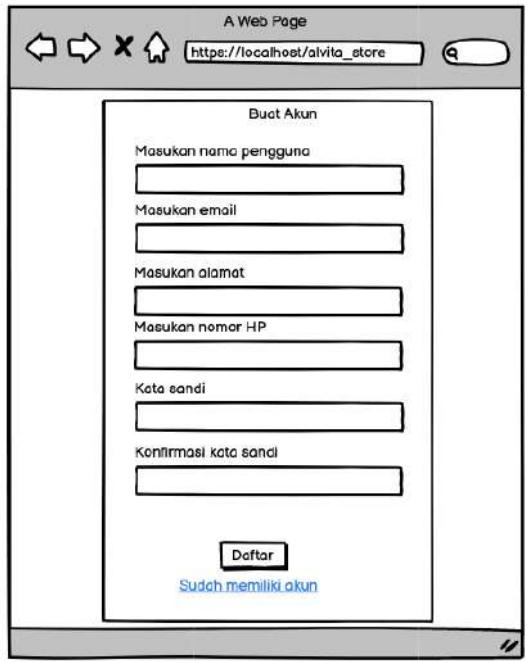
Perancangan antarmuka (*interface*) ini akan mempermudah dalam mengimplementasikan aplikasi. Berikut perancangan Sistem Informasi Penjualan Berbasis Website pada Alvita Collection.

1. Rancangan Antarmuka Halaman Login
   Setelah mengakses web sistem pengguna diminta untuk login sebelum masuk ke halaman utama dengan memasukkan nama pengguna dan kata sandi. Pengguna dapat memasukkan data sesuai dengan masing – masing levelnya pada sistem. Apabila pengguna lupa kata sandinya, maka pengguna dapat memulihkan atau membuat kata sandi baru menggunakan link 'lupa kata sandi'. Dan pada bagian paling bawah tampilan ada link 'buat akun', link ini khusus untuk pelanggan yang belum memiliki akun.



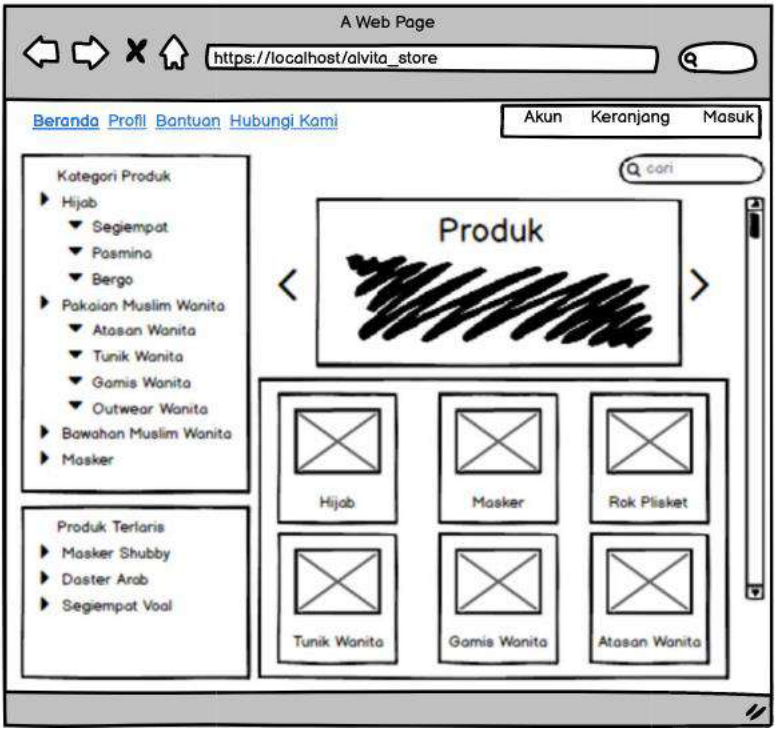
Gambar 6.8. Halaman Login

2. Rancangan Antarmuka Halaman Registrasi
   Bagi pelanggan yang belum punya akun dapat membuat akun baru pada form ini. Terdapat untuk membuat akun serta sebuah link yang mengarah ke halaman login apabila sudah memiliki akun.



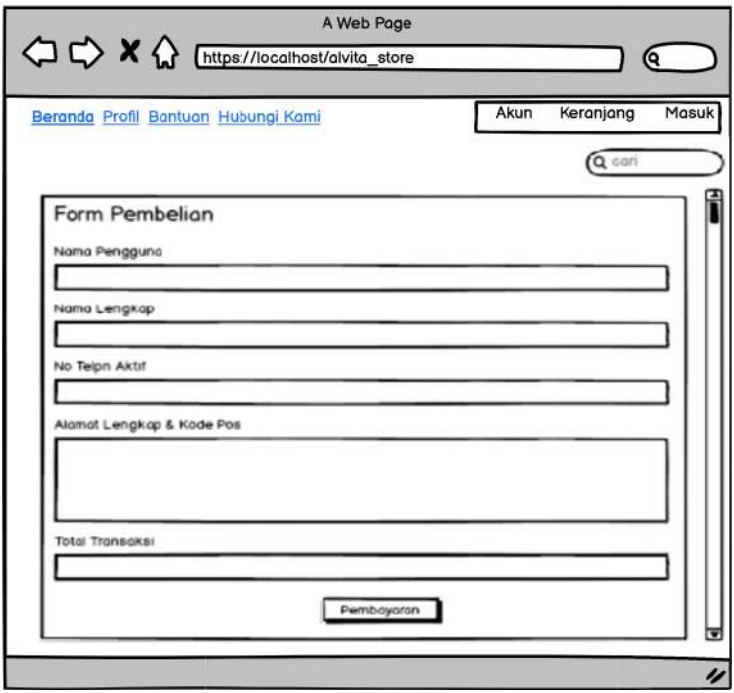
Gambar 6.9. Halaman Registrasi

3. Rancangan Antarmuka Halaman Utama Pelanggan
   Setelah berhasil login, pelanggan dapat memasuki halaman utama dimana terdapat beberapa link button dipojok kiri atas, dan button lainnya di kanan atas. Pada sidebar web akan ada tampilan kategori produk untuk mempermudah pelanggan mencari produknya. Kemudian di halaman utamanya, tampilan ini akan menampilkan produk – produk yang ada di Noviand Collcetion. Halaman pelanggan ini dapat di scroll sampai ke batas halaman web atau footer web.



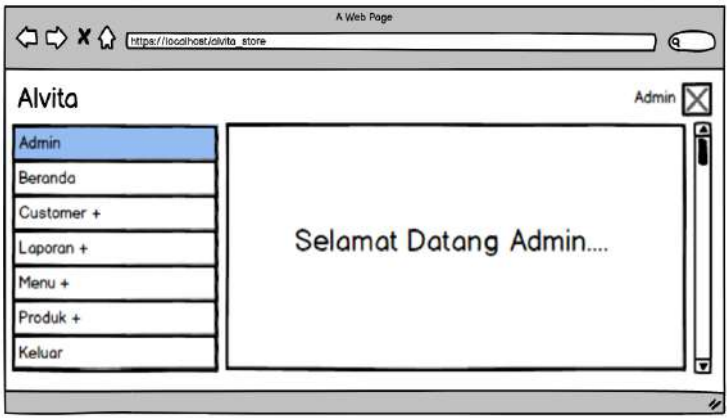
Gambar 6.10. Halaman Utama Pelanggan

4. Rancangan Antarmuka Pembelian Pelanggan
   Sebelum melakukan pembelian pelanggan baru akan dimintai sebuah data melalui form pembelian barang. Form tersebut terdiri dari nama pengguna, nama lengkap, no telepon aktif, alamat lengkap dan kode pos, dan total transaksi. Bagian paling bawah akan ada tombol pembayaran untuk melanjutkan proses pembelian ke pembayaran produk.



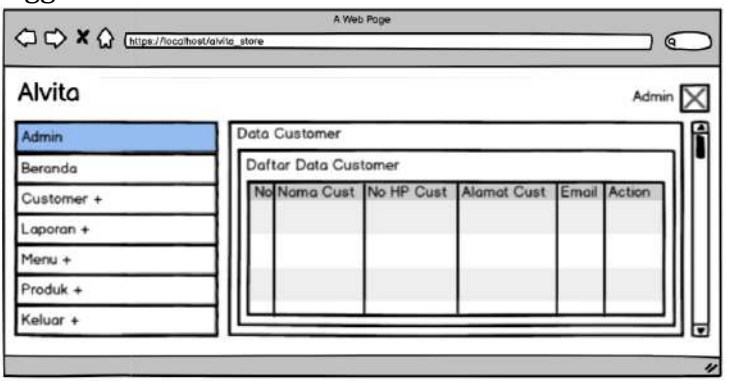
Gambar 6.11. Halaman Pembelian Pelanggan

5. Rancangan Antarmuka Halaman Utama Admin
   Pada bagian pojok kanan atas akan menampilkan foto admin dan nama admin. Sidebar admin akan menunjukan menu yang dapat diakses oleh admin, dan pada bagian utamanya akan menampilkan menu yang dipilih oleh admin.



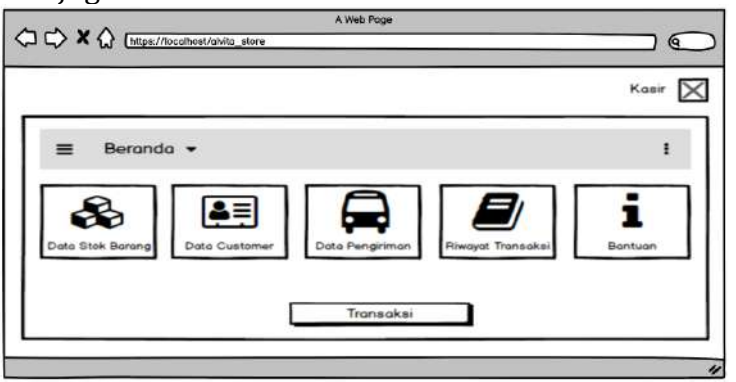
Gambar 6.12. Halaman Utama Admin

6. Rancangan Antarmuka Data Pelanggan
   Sidebar admin menampilkan menu yang dapat diakses oleh admin, dan pada bagian utamanya menampilkan data pelanggan.



Gambar 6.12. Halaman Data Pelanggan

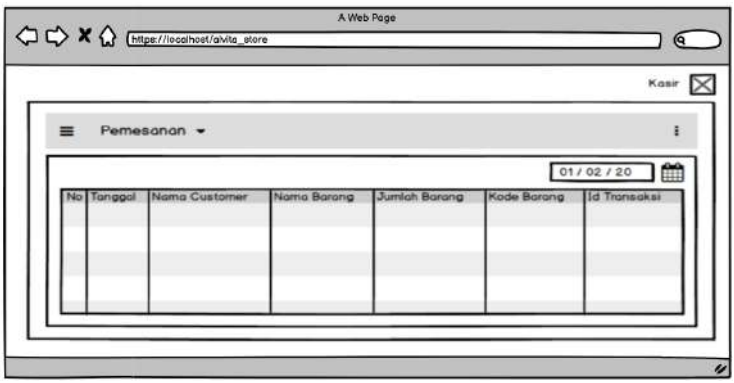
7. Rancangan Antarmuka Halaman Utama Kasir
   Top bar pada halaman utama kasir menampilkan foto dan nama kasir, sedangkan pada bagian bawahnya menampilkan transaksi untuk melakukan transaksi pada saat itu juga.



Gambar 6.13. Halaman Data Pelanggan

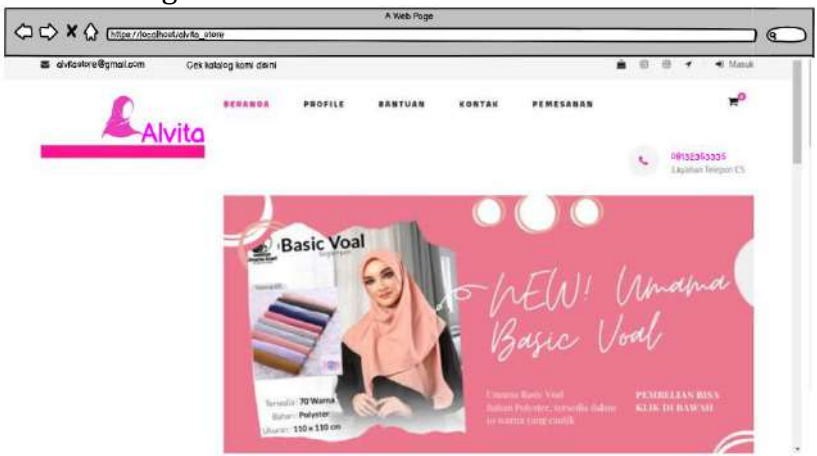
8. Rancangan Antarmuka Pemesanan Barang
   Pada tampilan kasir ini, kasir dapat melihat, dan melakukan pemrosesan pemesanan pelanggan. Pada tabel pemesanan

akan ditampilkan no, tanggal, nama pelanggan, nama barang, jumlah barang, kode barang, id transaksi.



Gambar 6.14. Halaman Data Pelanggan

Hasil Rancangan Antarmuka



Gambar 6.15. Halaman Rancangan Antarmuka

DAFTAR PUSTAKA

Agus Rusmana, E.K. (2014) *Analisa Sistem Informasi*. 2nd edn. Tangerang: Universitas Terbuka.

Fitria Nur Hasanah, R.S.U. (2020) *Buku Ajar Rekayasa Perangkat Lunak*. Edited by M.K. Mohammad Suryawinata, S.Pd. Sidoarjo: UMSIDA PRESS.

Hendra Jatnika (2013) *Pengantar Sistem Basis Data Memahami Konsep Dasar & Tuntunan Praktis Perancangan Database*. Edited by Putri Christian. Yogyakarta: CV ANDI OFFSET.

Mei Prabowo (2019) *Metodologi Pengembangan Sistem informasi*. Edited by M.K. Avin Wimar Budyastomo. Salatiga: Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LP2M) Intitut Agama Islam Negeri (IAIN) Salatiga.

Ridwan Setiawan (2018) *Modul Praktikum Basis Data*. Pertama. Edited by F.F. Roji. Garut: CV. INSAN AKADEMIKA.

Tata Sutabri (2012) *Konsep Sistem informasi*. Edited by Inunk Nastiti. Yogyakarta: CV ANDI OFFSET.

BIODATA PENULIS

Meri Azmi, ST., M. Cs
Dosen Program Studi Manajemen Informatika
Jurusan Teknologi Informasi Politeknik Negeri Padang

Penulis lahir di Solok Sumatera Barat pada tahun 1981. Penulis adalah dosen tetap pada Program Studi Manajemen Informatika Jurusan Teknologi Informasi Politeknik Negeri Padang. Menyelesaikan pendidikan S1 pada Jurusan Teknik Elektro dan melanjutkan S2 pada Jurusan Ilmu Komputer Universitas Gadjah Mada.

BIODATA PENULIS

Dr. Irmawati, S.Kom., MMSI.
Dosen Program Studi Sistem Informasi
Fakultas Teknik dan Informatika
Universitas Multimedia Nusantara

Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik dan Informatika Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 dan S2 pada Program Studi Sistem Informasi Universitas Gunadarma dan S3 pada departemen Teknik Elektro Universitas Indonesia. Penulis menekuni bidang keilmuan meliputi *Artificial Intelligence*, *Image Processing*, *Machine Learning*, dan *Deep Learning*.

BIODATA PENULIS

Yulita Salim
Dosen Program Studi Teknik Informatika
Fakultas Ilmu Komputer Universitas Muslim Indonesia Makassar

Penulis lahir di Batusitanduk, Palopo pada tanggal 22 Juli 1981. Penulis adalah Dosen Tetap Yayasan UMI pada Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Muslim Indonesia Makassar. Menyelesaikan pendidikan S1 pada Program Studi Teknik Informatika FIKOM UMI, S2 pada Jurusan Teknik Elektro Konsentrasi Informatics and Computer UNHAS dan melanjutkan studi S3 pada Program Teknologi Informasi di MIIT Universiti Kuala Lumpur, Malaysia.

Penulis dapat dihubungi melalui email: yulita.salim@umi.ac.id

BIODATA PENULIS

Budi Wijaya Rauf, S.Kom., M.Kom.

Penulis lahir di Kendari, Sulawesi tenggara pada tanggal 28 November 1996. Penulis merupakan dosen tetap di Universitas Muhammadiyah Kendari program studi Sistem dan Teknologi Informasi pada jenjang S-1. Penulis menekuni pada bidang kecerdasan buatan dengan fokus kepada pengembangan *machine learning*, *deep learning* dan sekarang sedang mendalami tentang *Generative AI*.

BIODATA PENULIS

Mutia Rahmi Dewi, S.Kom., M.Kom.
Dosen Program Studi Sarjana Terapan Teknologi Rekayasa Perangkat Lunak Jurusan Teknologi Informasi Politeknik Negeri Padang

Penulis lahir di Padang tanggal 4 September 1996. Penulis adalah dosen tetap pada Program Studi Sarjana Terapan Teknologi Rekayasa Perangkat Lunak Jurusan Teknologi Informasi di Politeknik Negeri Padang. Menyelesaikan pendidikan S1 dan S2 pada Jurusan Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya. Rumpun Mata Kuliah (RMK) yang diambil oleh penulis adalah Rekayasa Perangkat Lunak (RPL). Penulis dapat dihubungi melalui alamat surel mutiarahmidewi@gmail.com.

BIODATA PENULIS

Lutfi Syafirullah, S.T., M.Kom.
Dosen Program Studi Teknik Informatika
Jurusan Komputer dan Bisnis Politeknik Negeri Cilacap

Penulis lahir di Cilacap tanggal 21 November 1984. Penulis adalah Dosen Tetap pada Program Studi Teknik Informatika, Jurusan Komputer dan Bisnis, Politeknik Negeri Cilacap. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Ilmu Komputer. Saat ini penulis aktif melaksanakan Tri Dharma (pengajaran, penelitian dan pengabdian kepada Masyarakat) sejak tahun 2016 di Politeknik Negeri Cilacap.