BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu digunakan untuk mengidentifikasi perkembangan penelitian yang telah dilakukan sekaligus menemukan peluang pengembangan pada penelitian selanjutnya. Pada bidang analisis data transportasi, *Exploratory Data Analysis* (EDA) digunakan pada data transaksi Transjakarta untuk mengidentifikasi pola perjalanan penumpang dan waktu sibuk sebagai dasar optimalisasi operasional [10]. Selain itu, pengembangan sistem *Business Intelligence* yang mampu mengotomatisasi pengumpulan dan analisis umpan balik penumpang sehingga mempercepat proses evaluasi layanan transportasi [13].

Pada bidang *data warehouse*, pengimplementasian ETL dan star schema mampu meningkatkan kualitas analisis pada sektor properti [9]. *Data warehouse* sudah diterapkan pada domain *e-commerce* menggunakan Pentaho Data Integration dan OLAP untuk menghasilkan berbagai analisis bisnis [14], sedangkan penelitian lainnya mengintegrasikan *data warehouse* dengan Sistem Informasi Geografis (GIS) guna mendukung analisis spasial persebaran mahasiswa [15].

Dari sisi aksesibilitas data, *Natural Language Interface* berbasis *OLAP Hypercube* dikembangkan untuk mempercepat pemrosesan kueri pada *data warehouse* [16]. Pendekatan *semantic parsing* berbasis *rule-based* juga telah dilakukan pada *chatbot* NL2SQL dengan tingkat akurasi 96,72% [11], sedangkan pemanfaatan *Natural Language Interface* berbasis *Large Language Model* (LLM) mampu meningkatkan akurasi kueri sekaligus mempercepat proses penyusunan SQL [17]. Lebih lanjut, pengintegrasian LLM dan *intelligent agents* pada proses text-to-SQL sehingga menghasilkan *valid SQL* hingga 99% dan *execution accuracy* sebesar 95,5% [12]. Dari aspek metodologi pengembangan basis data, Anwar dan Purnama [17] membuktikan bahwa *Database Life Cycle* (DBLC) menyediakan tahapan yang sistematis mulai dari desain konseptual, logikal, hingga fisikal dalam membangun basis data. Rangkuman penelitian terdahulu yang menjadi acuan dalam penelitian ini disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis	Judul	Jurnal	Tahun	Metode	Domain	Hasil
1.	Nouval Trezandy Lapatta, Andi Hendra, Raden Muhamad Firzatullah [10]	*Transaction Analysis for Operational Optimization in Transjakarta Public Transportation System*	Inland Waterways Journal	2024	Exploratory Data Analysis (EDA)	Transportasi Bus Transjakarta	EDA menghasilkan wawasan prediktif mengenai waktu sibuk, profil demografi penumpang (54% wanita), serta preferensi pembayaran cashless (50% menggunakan Bank DKI) guna mengoptimalkan efisiensi operasional layanan Transjakarta.
2.	Willsen Wijaya, Jansen Wiratama, Santo Fernandi Wijaya [9]	*Implementation of Data Warehouse and Star Schema for Optimizing Property Business Decision Making*	G-Tech: Jurnal Teknologi Terapan	2024	ETL, *Star Schema*	*Data warehouse* pada Properti	*Data warehouse* berhasil meningkatkan analisis data untuk pengambilan keputusan, seperti menentukan lokasi strategis dan membandingkan harga properti dengan pasar
3.	Angelina Yang, Jansen Wiratama, Santo Fernandi Wijaya [14]	*Empowering Data Transformation: Transforming Raw Data into A Strategic Planning for E-Commerce Success*	Journal of Information Systems and Informatics	2024	ETL, *Star Schema,* OLAP	*Data Warehouse* pada e-*Commerce*	ETL menggunakan Pentaho dan *star schema* memungkinkan transformasi data mentah menjadi *insight* strategis melalui analisis seperti *sales trend*, *product performance*, dan *operational efficiency* untuk mendukung pengambilan keputusan bisnis
4.	Irmayani, Rahman, Syahbudin, Asrul Azhari Muin, A. Mustika Abidin [15]	Integrasi Data Warehouse dan Sistem Informasi Geografis untuk Analisis Spasial Persebaran Mahasiswa Baru	Jurnal Publikasi Sistem Informasi dan Manajemen Bisnis	2025	ETL, *Star Schema*	Analitik data spasial	Sistem berhasil mengolah 15.826 data mahasiswa periode 2022–2024 dan mengidentifikasi bahwa 53.2% berasal dari 5 wilayah utama
5.	Adi Kurniawan, Abdiansah Abdiansah, Alvi Syahrini Utami [11]	*NL2SQL for Chatbot with Semantic Parsing Using Rule-Based Methods*	Sriwijaya Journal of Informatic and Applications	2024	NL2SQL, *Semantic Parsing*, *Rule-Based*	*Chatbot* di sistem akademik	Chatbot mampu menghasilkan akurasi 96.72% dan waktu eksekusi rata-rata 0.05 detik pada dataset terbatas sebanyak 122 kueri

No.	Penulis	Judul	Jurnal	Tahun	Metode	Domain	Hasil
6.	Samuel Ojuria, The Anh Hana, Raymond Chiong, Alessandro Di Stefano [12]	*Optimizing text-to-SQL conversion techniques through the integration of intelligent agents and large language models*	Information Processing and Management	2025	NL2SQL, LLM	Analisis penjualan mobil	Pendekatan berbasis LLM seperti GPT-4 menghasilkan akurasi tertinggi mencapai 99% valid SQL dan 95% execution accuracy
7.	Fadi H. Hazboun, Majdi Owda, Amani Yousef Owda [16]	*A Natural Language Interface to Relational Databases Using an OLAP Hypercube*	AI	2021	NLP, NL2SQL, OLAP *Hypercube*	Informasi akademik	NL2SQL pada OLAP *hypercube* menghasilkan kecepatan respons sekitar 0.001 detik dan total proses sekitar 0.07–0.14 detik serta mendukung pemrosesan kueri kompleks pada data besar
8.	Michał Banka, Jakub Daniłowski, Mirosław Czerlinski, Jakub Murawski, Renata Zochowska, Aleksander Sobota [13]	*A Feedback Analysis Automation Using Business Intelligence Technology in Companies Organizing Urban Public Transport*	Sustainability	2022	Power BI, Power Automate	Sistem analitik transportasi umum	Sistem BI mampu mengotomatisasi pengumpulan dan analisis *feedback* penumpang sehingga mempercepat pengambilan keputusan dan peningkatan layanan transportasi
9.	Praneeth Thoutam [17]	*Unlocking Insights with Natural Language: The Role of Natural Language Interfaces in Modern Data Exploration*	International Journal for Multidisciplinary Research	2024	NLP, NLI, LLM, *Semantic Parsing*	Analitik Data	Natural Language Interfaces mampu menghasilkan akurasi hingga hingga sekitar 87.3% serta mempercepat proses formulasi kueri hingga 94% dibandingkan SQL tradisional
10.	Muhammad Rehan Anwar, Suryari Purnama [18]	*Boarding House Search Information System Database Design*	International Journal of Cyber and IT Service Management	2022	DBLC	DBLC pada Sistem Pencarian Kos	Desain *database* menghasilkan total 7 entitas yang terstruktur untuk mendukung sistem pencarian kos secara efisien

Berdasarkan penelitian-penelitian pada Tabel 2.1, masing-masing penelitian telah memberikan kontribusi pada aspek tertentu, namun masih memiliki beberapa keterbatasan. Penelitian pada bidang analisis transportasi berfokus pada pengolahan data untuk memperoleh insight operasional, tetapi belum membangun infrastruktur *data warehouse* yang mampu mengintegrasikan data dari berbagai sistem operasional [10], [13]. Sebaliknya, penelitian mengenai *data warehouse* telah berhasil membangun repositori analitik menggunakan ETL dan *star schema*, namun implementasinya masih terbatas pada domain tertentu serta belum menyediakan mekanisme akses data secara interaktif melalui bahasa alami [9], [14], [15].

Pada sisi aksesibilitas data, penelitian NL2SQL berbasis *rule-based* masih bergantung pada aturan yang telah ditentukan sehingga kurang fleksibel dalam menangani variasi bahasa alami [11]. Sementara itu, penelitian berbasis *Large Language Model* lebih berfokus pada peningkatan akurasi konversi *natural language* ke SQL tanpa membahas integrasi dengan data warehouse maupun penerapannya pada sistem analitik transportasi [12], [16], [17]. Selain itu, penelitian mengenai DBLC hanya membahas perancangan basis data operasional dan belum menerapkannya pada pembangunan *data warehouse* [18].

Berdasarkan keterbatasan tersebut, belum ditemukan penelitian yang mengintegrasikan proses ETL dari beberapa basis data operasional ke dalam sebuah *data warehouse* berbasis *star schema*, kemudian menyediakan akses analitik melalui *chatbot* NL2SQL berbasis *Large Language Model* serta menyajikan hasil analisis dalam bentuk visualisasi data. Oleh karena itu, penelitian ini mengusulkan pembangunan sistem analitik terintegrasi yang mengombinasikan proses ETL, *data warehouse*, *chatbot* NL2SQL berbasis LLM, dan visualisasi *dashboard* untuk mendukung pengambilan keputusan operasional pada perusahaan transportasi bus.

2.2 Teori yang berkaitan

2.2.1 *Data Warehouse*

Data warehouse adalah repositori data terpusat yang menyimpan data terstruktur dari berbagai sistem operasional untuk mendukung analisis dan pengambilan keputusan manajemen [19]. Secara konseptual, *data warehouse* memiliki empat karakteristik utama, yaitu *subject-oriented*, *integrated*, *time-variant*, dan *non-volatile* [20]. *Subject-oriented* berarti data diorganisasikan

berdasarkan subjek bisnis tertentu seperti penjualan atau pelanggan, bukan berdasarkan proses aplikasi. *Integrated* berarti data yang berasal dari berbagai sumber yang bermacam-macam diseragamkan ke dalam format dan penamaan yang konsisten. *Time-variant* berarti setiap data menyimpan informasi waktu sehingga memungkinkan analisis tren historis. *Non-volatile* berarti data yang sudah masuk ke dalam warehouse tidak diubah atau dihapus, melainkan hanya dibaca untuk keperluan analisis. Dari sisi arsitektur, data warehouse umumnya terdiri dari basis data pusat berbasis relasional atau multidimensi, mekanisme integrasi data melalui proses ETL, serta lapisan pelaporan dan analitik untuk menghasilkan dashboard dan laporan bisnis [21]. Dengan karakteristik dan arsitektur tersebut, *data warehouse* menjadi *single source of truth* yang andal bagi organisasi dalam melakukan analisis tren historis, peramalan, dan evaluasi kinerja operasional. Perbedaan antara basis data transaksional dan *data warehouse* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perbedaan *Transactional Database* dan *Data Warehouse*
Sumber: [22]

Parameter	***Database* Transaksional**	***Data Warehouse***
Tujuan	Menangani banyak transaksi kecil, cepat, *real-time*	Analisis data besar, laporan, tren
Jenis Data	Data terkini, selalu *up to date*	Data historis, di *update* berkala
Struktur data	Normalisasi	Denormalisasi
Operasi	Dominan *insert/update/delete*	Banyak kueri *read* kompleks agregasi kolom
Beban Kerja	OLTP (Online Transaction Processing)	OLAP (*Online Analytical Processing*)
Contoh Penggunaan	Input penjualan, transfer bank, pemesanan online	Laporan bulanan, dashboard manajemen, analitik historis

Tabel 2.2 menyajikan perbandingan antara basis data transaksional dan *data warehouse* berdasarkan beberapa parameter utama yang mencerminkan perbedaan fungsi dan karakteristik keduanya [22]. Basis data transaksional dirancang untuk menangani banyak transaksi kecil secara cepat dan bersifat *real-time* dengan data yang selalu baru, menggunakan struktur data yang ter-normalisasi untuk menjaga konsistensi, serta mendukung operasi *insert*, *update*, dan *delete* dalam beban kerja OLTP. Sebaliknya, *data warehouse* berfokus pada analisis data dalam skala besar dengan memanfaatkan data historis yang diperbarui secara berkala, menggunakan struktur denormalisasi untuk mengoptimalkan performa kueri, serta mendukung

operasi pembacaan data kompleks seperti agregasi dalam beban kerja OLAP. Perbedaan ini juga tercermin dalam penggunaannya, di mana basis data transaksional digunakan untuk aktivitas operasional seperti input penjualan atau transaksi perbankan, sedangkan *data warehouse* dimanfaatkan untuk kebutuhan analitis seperti pembuatan laporan, dashboard manajemen, dan analisis tren historis.

2.2.2 *Star Schema*

Star schema merupakan salah satu model data multidimensi yang umum digunakan dalam *data warehouse* untuk mempermudah serta mempercepat proses analisis data bisnis [23]. Struktur skema ini menyerupai bentuk bintang, di mana terdapat satu tabel pusat yang dikelilingi oleh beberapa tabel pendukung. Secara konseptual, *star schema* terdiri dari dua komponen utama, yaitu tabel fakta dan tabel dimensi [9]. Tabel fakta berperan sebagai tabel pusat yang menyimpan data transaksi atau data kuantitatif yang dapat diukur, seperti penjualan, jumlah tiket, atau pendapatan. Sementara itu, tabel dimensi merupakan tabel yang mengelilingi tabel fakta dan berisi atribut deskriptif yang memberikan konteks terhadap data pada tabel fakta, seperti waktu, produk, pelanggan, lokasi, atau cabang. Setiap *record* pada tabel fakta terhubung dengan tabel dimensi melalui *foreign key* sehingga proses analisis data dapat dilakukan dari berbagai sudut pandang, misalnya berdasarkan waktu, jenis produk, maupun lokasi, sehingga mendukung kebutuhan analisis yang lebih fleksibel dan efisien. Perbedaan antara *star schema* dan *snowflake schema* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Perbedaan *Star Schema* dan *Snowflake Schema*
Sumber: [24]

Parameter	***Star Schema***	***Snowflake Schema***
Bentuk dimensi	1 tabel dimensi (denormalisasi)	Dimensi dipecah ke beberapa sub-dimensi (dinormalisasi)
Relasi antar dimensi	Tidak ada, semua langsung ke fact	Ada rantai relasi dimensi dengan subdimensi
Performa kueri	Lebih cepat karena *join* lebih sedikit	Lebih lambat karena melibatkan lebih banyak *join*
Kompleksitas	Relatif sederhana dan mudah dipahami	Lebih kompleks dan membutuhkan perancangan lebih detail

Pada tabel 2.3 terdapat perbedaan antara *star schema* dan *snowflake schema* terletak pada struktur dimensi, relasi antar tabel, performa kueri, serta tingkat

kompleksitasnya [24]. Pada *star schema*, tabel dimensi disusun dalam satu tabel tunggal yang bersifat denormalisasi, sehingga tidak terdapat relasi antar dimensi karena seluruh tabel dimensi terhubung langsung ke tabel fakta. *Star schema* juga cenderung lebih cepat dalam proses kueri karena melibatkan lebih sedikit operasi join dan memiliki struktur yang lebih sederhana dan mudah dipahami.

2.2.3 *Online Analytical Processing* (OLAP)

Online Analytical Processing (OLAP) merupakan pendekatan yang digunakan untuk menganalisis data secara multidimensi, sehingga pengguna dapat melihat informasi dari berbagai sudut pandang secara cepat dan interaktif [25]. OLAP umumnya terintegrasi dengan *data warehouse* atau basis data berukuran besar dan menyajikan data dalam bentuk struktur multidimensi seperti kubus yang merepresentasikan berbagai dimensi, misalnya waktu, lokasi, produk, dan pelanggan. Melalui OLAP, pengguna dapat melakukan berbagai operasi analisis seperti *drill-down* untuk melihat data secara lebih rinci, *roll-up* untuk memperoleh ringkasan data, serta *slice and dice*, *pivot*, dan filter untuk mengeksplorasi data dari perspektif yang berbeda [26]. Teknologi ini banyak digunakan dalam sistem *business intelligence*, *customer relationship management* (CRM), serta dashboard kinerja organisasi, karena kemampuannya dalam mengagregasi data dalam jumlah besar secara efisien serta menyajikan informasi yang mendukung pengambilan keputusan taktis maupun strategis.

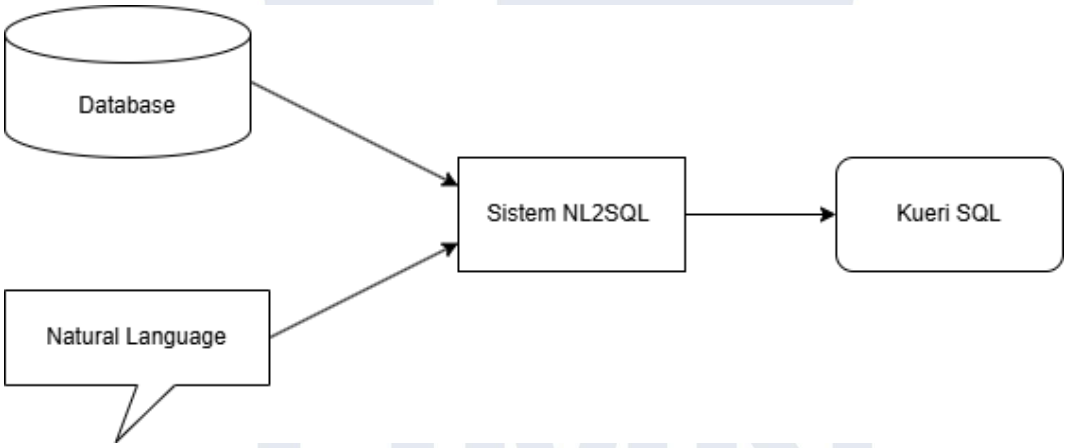
2.2.4 *Chatbot*

Chatbot adalah program komputer yang didesain untuk mensimulasikan percakapan dengan manusia yang biasanya terintegrasi dengan aplikasi *mobile* atau *website* secara online [27]. *Chatbot* juga dapat didefinisikan sebagai robot percakapan virtual yang bersifat non-manusia dan dapat menjawab pertanyaan dalam bentuk teks maupun suara [28]. Salah satu penggunaan *chatbot* adalah sebagai agen dialog dalam industri jasa [29]. Contohnya yaitu Erica, *chatbot* yang dikembangkan oleh Bank of America, yang dirancang untuk memberikan layanan perbankan melalui aplikasi mobile [30]. Bagi perusahaan, *chatbot* dapat melayani permintaan pelanggan kapan saja, meningkatkan kualitas pengalaman pengguna, serta membantu mengurangi biaya operasional layanan. Meskipun kemampuan

chatbot belum bisa menyamai manusia dalam menjawab pertanyaan yang rumit, chatbot memberikan kelebihan berupa aksesibilitas, responsivitas, dan ketersediaan sepanjang waktu.

2.2.5 NL2SQL

Natural Language to SQL (NL2SQL) adalah proses yang dikembangkan dari *Natural Language Processing* (NLP) untuk mentranslasi pertanyaan atau pernyataan dalam *natural language* atau bahasa alami menjadi kueri SQL yang terstruktur [31]. Proses ini dirancang untuk meningkatkan aksesibilitas, sehingga mereka dapat berinteraksi langsung dengan basis data tanpa perlu menulis kueri SQL [32]. Dengan pendekatan ini, proses pencarian informasi menjadi lebih intuitif karena pengguna cukup menyampaikan kebutuhan data dalam bentuk bahasa sehari-hari. Alur kerja NL2SQL secara umum dapat dilihat pada Gambar 2.1.



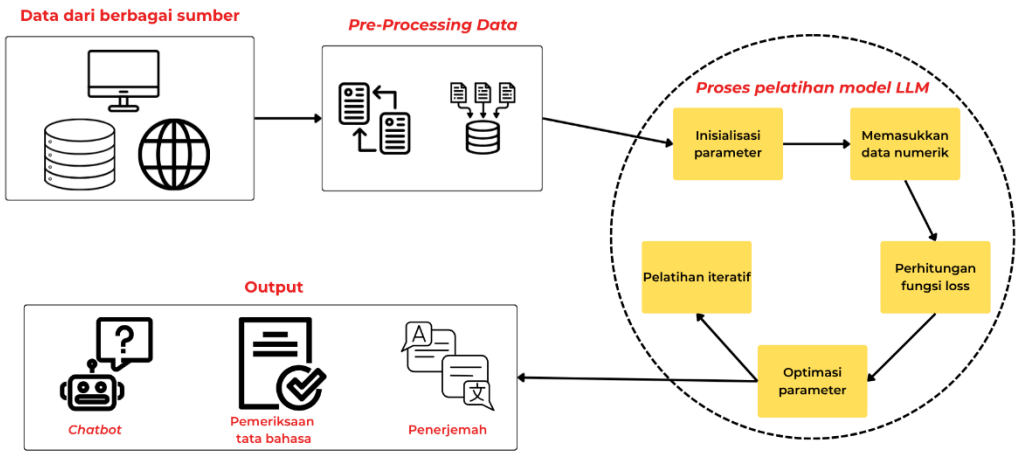
Gambar 2.1 Alur Proses NL2SQL
Sumber: [33]

Alur kerja NL2SQL secara umum ditunjukkan pada Gambar 2.1, yang menggambarkan tahapan mulai dari input berupa pertanyaan dalam bahasa alami hingga menghasilkan keluaran berupa hasil kueri dari basis data [33]. Pada tahap awal, sistem menerima input pengguna dalam bentuk teks, kemudian dilakukan proses pemahaman bahasa untuk mengidentifikasi maksud dan entitas yang relevan. Selanjutnya, sistem akan melakukan pemetaan terhadap skema basis data yang digunakan, seperti tabel, atribut, dan relasi antar tabel, guna memastikan bahwa kueri yang dihasilkan sesuai dengan struktur basis data. Setelah itu, sistem membangun kueri SQL yang valid berdasarkan hasil pemahaman konteks dan

pemetaan skema. Kueri yang dihasilkan kemudian dieksekusi pada sistem basis data, dan hasilnya dikembalikan kepada pengguna dalam bentuk data yang relevan dengan pertanyaan yang diajukan.

2.2.6 *Large Language Model* (LLM)

Large Language Model (LLM) merupakan model kecerdasan buatan berbasis *deep learning* yang dirancang untuk memahami, memproses, dan menghasilkan bahasa manusia secara alami [34]. Model ini dilatih menggunakan kumpulan data teks dalam jumlah sangat besar, sehingga mampu mempelajari pola bahasa, konteks, serta hubungan antar kata secara kompleks. Mayoritas LLM dibangun dengan arsitektur *transformer* dan memiliki jumlah parameter yang sangat besar, mulai dari miliaran hingga triliunan, yang memungkinkan model memprediksi kata atau urutan teks berikutnya dengan tingkat akurasi yang tinggi [35]. Berkat kemampuan tersebut, LLM dapat digunakan untuk berbagai tugas tanpa memerlukan pelatihan khusus untuk setiap kasus, seperti menjawab pertanyaan, menerjemahkan bahasa, merangkum teks, menghasilkan konten baru, hingga menulis kode program melalui pendekatan *few-shot* maupun *zero-shot learning*. Alur kerja LLM secara umum dapat dilihat pada Gambar 2.2.



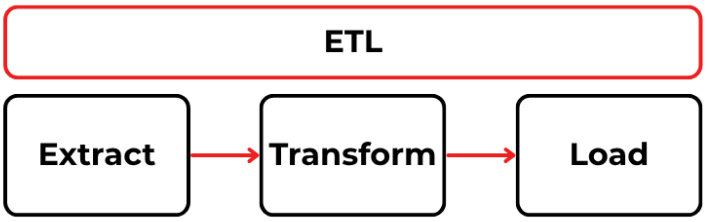
Gambar 2.2 Proses Pelatihan LLM
Sumber: [36]

Proses pelatihan LLM pada Gambar 2.2 dimulai dari pengumpulan data teks dalam jumlah besar dari berbagai sumber, yang kemudian melalui tahap *pre-*

processing untuk membersihkan dan menyiapkan data sebelum digunakan. Selanjutnya, pada tahap pelatihan, data teks diubah menjadi representasi numerik dan diproses melalui serangkaian langkah, yaitu inisialisasi parameter secara acak, perhitungan *loss function* untuk mengukur kesalahan prediksi, serta optimasi parameter yang dilakukan secara iteratif hingga model mencapai tingkat akurasi yang diinginkan. Setelah proses pelatihan selesai, model yang dihasilkan mampu digunakan untuk berbagai tugas pemrosesan bahasa alami, seperti *chatbot*, penerjemahan, dan pemeriksaan tata bahasa sehingga mendukung pengolahan informasi secara otomatis dan efisien.

2.2.7 *Extract, Transform, Load* (ETL)

Extract, Transform, Load (ETL) adalah proses dalam integrasi data yang digunakan untuk mengumpulkan data dari berbagai sumber, mentransformasikannya ke dalam format yang sesuai, serta menyimpannya ke dalam sistem penyimpanan terpusat seperti *data warehouse* untuk keperluan analisis [37]. Proses ETL berperan sebagai penghubung antara data mentah dan informasi yang siap digunakan dalam pengambilan keputusan. Dalam lingkungan data modern, ETL tidak hanya berfungsi sebagai proses teknis, tetapi juga sebagai mekanisme yang mendukung kualitas data dan keandalan sistem *business intelligence* [38]. Tahapan ETL dapat dilihat pada Gambar 2.3.



Gambar 2.3 Tahapan ETL
Sumber: [39]

Berdasarkan Gambar 2.7, proses ETL terdiri dari tiga tahapan utama sebagai berikut [39]:

1. *Extract*

Tahap *extract* merupakan proses pengambilan data dari berbagai sumber yang beragam, seperti basis data operasional, *file*, API, maupun sistem eksternal lainnya. Data yang diekstraksi dapat berupa data terstruktur maupun tidak terstruktur. Pada tahap ini, data biasanya ditempatkan pada penyimpanan sementara sebelum diproses lebih lanjut, dengan tujuan untuk menghindari gangguan terhadap sistem sumber serta memungkinkan validasi awal terhadap data yang diambil.

2. *Transform*

Tahap *transform* merupakan proses pengolahan data untuk meningkatkan kualitas dan konsistensi data sebelum dimasukkan ke dalam *data warehouse*. Proses transformasi meliputi pembersihan data, standarisasi format, penghapusan duplikasi, penggabungan data dari berbagai sumber dan penerapan aturan bisnis. Transformasi merupakan tahap paling krusial karena pada tahap ini data mentah diubah menjadi data yang memiliki makna dan siap digunakan untuk analisis. Proses ini juga berperan dalam memastikan integritas, konsistensi, serta kesesuaian data dengan kebutuhan sistem analitik.

3. *Load*

Tahap *load* merupakan proses pemindahan data yang telah ditransformasi ke dalam sistem target seperti *data warehouse* atau *data lake*. Proses ini dapat dilakukan dalam beberapa metode, seperti *initial load* atau pengisian data awal secara keseluruhan, *incremental load* atau penambahan data secara berkala berdasarkan perubahan, dan *full refresh* atau penggantian seluruh data lama dengan data baru. Tahap ini harus dilakukan secara efisien untuk memastikan performa sistem tetap optimal serta menjaga integritas data selama proses pemuatan.

2.2.8 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa standar yang divisualisasikan dalam bentuk diagram untuk mendefinisikan, mendokumentasikan, dan membangun sistem perangkat lunak, khususnya yang berorientasi objek [40]. UML digunakan untuk memberikan gambaran spesifikasi

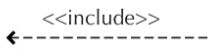
dalam pengembangan sistem, termasuk proses bisnis, pembuatan kelas, desain basis data, dan komponen lainnya yang dibutuhkan dalam pengembangan sistem [41]. Selain itu, UML juga membantu menggambarkan kebutuhan sistem melalui skenario penggunaan oleh pengguna serta menunjukkan batasan-batasan sistem yang relevan. Dengan menyediakan perspektif struktural, perilaku, dan fungsional, UML menjadi alat penting dalam menghasilkan model yang baik untuk pengembangan perangkat lunak [42]. Penggunaan UML dapat mempercepat proses pengembangan dengan memanfaatkan kembali diagram UML yang sudah ada, hanya perlu sedikit penyesuaian sesuai kebutuhan. Beberapa macam UML yaitu:

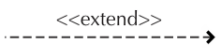
1. *Use Case Diagram*

   Use case diagram digunakan dalam fase analisis proyek untuk mengidentifikasi fungsionalitas sistem [43]. Diagram ini memisahkan sistem menjadi aktor (peran pengguna seperti manusia, komputer, atau perangkat lunak lain) dan *case* (fungsi-fungsi sistem). Komponen utama *use case* diagram meliputi aktor untuk merepresentasikan peran pengguna, *case* untuk menggambarkan semua fungsi sistem, *system boundary* untuk menentukan batasan sistem, dan *dependency* untuk menunjukkan hubungan antar elemen. Elemen-elemen *use case diagram* disajikan pada Tabel 2.4.

Tabel 2.4 Sintaks *Use Case Diagram*

Sumber: [44]

Elemen	Simbol	Deskripsi
Actor		Orang atau sistem luar yang berinteraksi dengan sistem.
Use Case		Fungsi atau fitur utama di dalam sistem.
Subject Boundary		Batas yang menunjukkan ruang lingkup sistem yang dimodelkan.
Association Relationship		Hubungan antara aktor dan use case yang digunakan.
Include Relationship		Menunjukkan use case yang selalu memanggil use case lain sebagai bagian prosesnya.

Elemen	Simbol	Deskripsi
Extend Relationship		Menunjukkan fungsi tambahan yang dijalankan pada kondisi tertentu.
Generalization Relationship		Menunjukkan hubungan pewarisan antara aktor atau use case yang memiliki karakteristik serupa.

Berdasarkan Tabel 2.4, *use case diagram* tersusun atas elemen-elemen yang digunakan untuk menggambarkan interaksi antara aktor dan sistem. Aktor berperan sebagai pihak yang menggunakan atau berinteraksi dengan sistem, sedangkan *use case* merepresentasikan layanan atau fungsi yang disediakan sistem. Selain itu, use case diagram dilengkapi dengan *subject boundary* untuk menunjukkan batas sistem serta berbagai jenis *relationship* yang digunakan untuk menggambarkan hubungan antar elemen, seperti *association*, *include*, *extend*, dan *generalization*. Melalui elemen-elemen tersebut, *use case diagram* dapat digunakan untuk memodelkan kebutuhan fungsional sistem secara visual dan mudah dipahami.

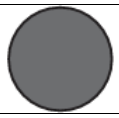
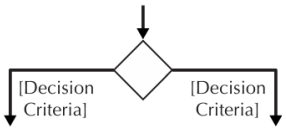
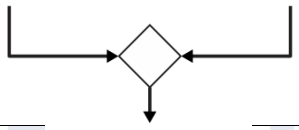
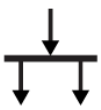
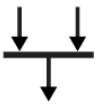
2. *Activity Diagram*

Activity diagram digunakan untuk menggambarkan aspek dinamis sistem melalui representasi alur aktivitas [43]. Diagram ini berbentuk diagram alur yang menunjukkan perpindahan dari satu aktivitas ke aktivitas lain, baik secara sekuensial, bercabang, maupun paralel. Komponen utama diagram ini meliputi *action* sebagai unit dasar perilaku, *swimlanes* untuk membagi aktivitas ke dalam alur paralel atau sekuensial, *start state* sebagai titik awal, *final state* untuk titik akhir, dan *flow final* untuk mengakhiri jalur tertentu. Diagram ini juga menggunakan elemen seperti *fork*, *join*, *decision*, dan *merge* untuk menangani kontrol alur dalam sistem. Elemen-elemen *activity diagram* disajikan pada Tabel 2.5.

Tabel 2.5 Sintaks *Activity Diagram*

Sumber: [44]

Elemen	Simbol	Deskripsi
Action		Satuan perilaku terkecil/tunggal yang tidak bisa dibagi lagi.

Elemen	Simbol	Deskripsi
Control Flow		Menunjukkan urutan jalannya eksekusi dari satu langkah ke langkah berikutnya.
Initial Node		Titik awal dimulainya suatu alur aktivitas.
Final Activity Node		Menghentikan seluruh alur kerja dalam diagram tersebut.
Decision Node		Titik percabangan kondisi (IF/THEN). Alur akan memilih salah satu jalur berdasarkan kriteria.
Merge Node		Menyatukan kembali jalur-jalur yang sebelumnya terpisah dari *decision node*.
Fork Node		Memecah satu aliran menjadi beberapa aliran aktivitas yang berjalan bersamaan (paralel).
Join Node		Menggabungkan kembali beberapa aliran paralel menjadi satu aliran tunggal.

Berdasarkan Tabel 2.5, *activity diagram* terdiri atas beberapa elemen yang digunakan untuk menggambarkan alur proses dan aktivitas dalam sistem. *Action* digunakan untuk merepresentasikan aktivitas yang dilakukan, sedangkan *control flow* menunjukkan urutan perpindahan dari satu aktivitas ke aktivitas lainnya. Alur aktivitas dimulai dari *initial node* dan berakhir pada *final activity node*. Selain itu, *activity diagram* menyediakan elemen *decision node* dan *merge node* untuk menggambarkan percabangan dan penggabungan alur berdasarkan kondisi tertentu, serta *fork node* dan *join node* untuk memodelkan aktivitas yang berjalan secara paralel. Dengan elemen-elemen tersebut, activity diagram dapat digunakan untuk memvisualisasikan proses bisnis maupun alur kerja sistem secara terstruktur dan mudah dipahami.

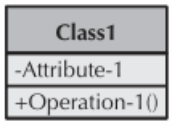
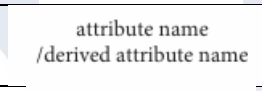
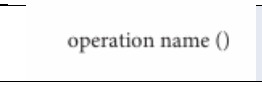
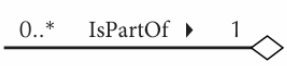
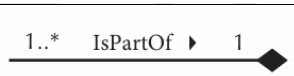
3. *Class Diagram*

Class diagram mendeskripsikan atribut dan operasi dari sebuah kelas serta berbagai batasan yang diterapkan pada sistem [43]. Diagram ini sangat

penting dalam pemodelan sistem berorientasi objek karena dapat langsung dipetakan ke bahasa pemrograman berorientasi objek. Komponen utama *class diagram* adalah kelas, yang terdiri dari tiga bagian: nama kelas, atribut, dan operasi. Hubungan antar kelas dijelaskan melalui relasi. Diagram ini juga mencakup *abstract class*, yang berisi operasi umum yang dapat digunakan kembali oleh *subclass* dan *interface*, yang mendefinisikan layanan untuk kelas atau komponen lain. Selain itu, ada *template class* yang merepresentasikan keluarga kelas dengan struktur dan perilaku independen terhadap parameter formalnya.

Tabel 2.6 Sintaks *Class Diagram*

Sumber: [44]

Elemen	Simbol	Deskripsi
Class		Representasi objek dunia nyata yang menyimpan informasi. Baris 1: Nama, Baris 2: Atribut, Baris 3: Operasi.
Attribute		Properti atau data milik kelas yang menggambarkan karakteristiknya.
Operation		Fungsi atau *method* tindakan yang bisa dilakukan oleh kelas tersebut.
Assosiation		Hubungan relasi antar kelas, dilengkapi angka kardinalitas (seperti 1, 0..*) untuk menghitung hubungan jumlah objek.
Generalization		Hubungan pewarisan (*inheritance* atau hubungan induk-anak)
Aggregation		Hubungan bagian-dari (*part-of*) yang bersifat lemah.
Composition		Hubungan bagian-dari (*part-of*) yang sangat kuat secara fisik.

Berdasarkan Tabel 2.6, *class diagram* digunakan untuk menggambarkan struktur statis sistem melalui representasi kelas, atribut, operasi, dan hubungan antar kelas. Kelas menjadi komponen utama yang menyimpan data serta perilaku sistem melalui atribut dan operasi yang dimilikinya. Hubungan antar kelas dapat direpresentasikan menggunakan *association, generalization, aggregation*, dan *composition* sesuai dengan

kebutuhan pemodelan. Melalui elemen-elemen tersebut, class diagram dapat digunakan untuk menunjukkan struktur objek, hubungan antar objek, serta organisasi data dan fungsi dalam sistem secara terstruktur sehingga memudahkan proses perancangan dan implementasi sistem berorientasi objek.

2.2.9 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah tahap awal dan salah satu teknik yang paling umum digunakan dalam perancangan basis data relasional [45]. ERD digunakan sebagai diagram grafis untuk memodelkan struktur data pada tahap konseptual sebelum basis data fisik dibuat. ERD menggambarkan entitas, atribut, dan relasi dalam suatu sistem sehingga kebutuhan data pengguna dapat diterjemahkan menjadi desain basis data yang konsisten dan mudah diimplementasikan [46]. ERD berfungsi sebagai alat bantu komunikasi antara analis, pengembang, dan pengguna non-teknis, serta sebagai *blueprint* konseptual yang kemudian diterjemahkan ke dalam bentuk tabel, *primary key*, *foreign key*, dan *constraint* pada model relasional [47]. ERD yang dirancang dengan baik juga dapat membantu menjaga integritas data dan mengurangi redundansi.

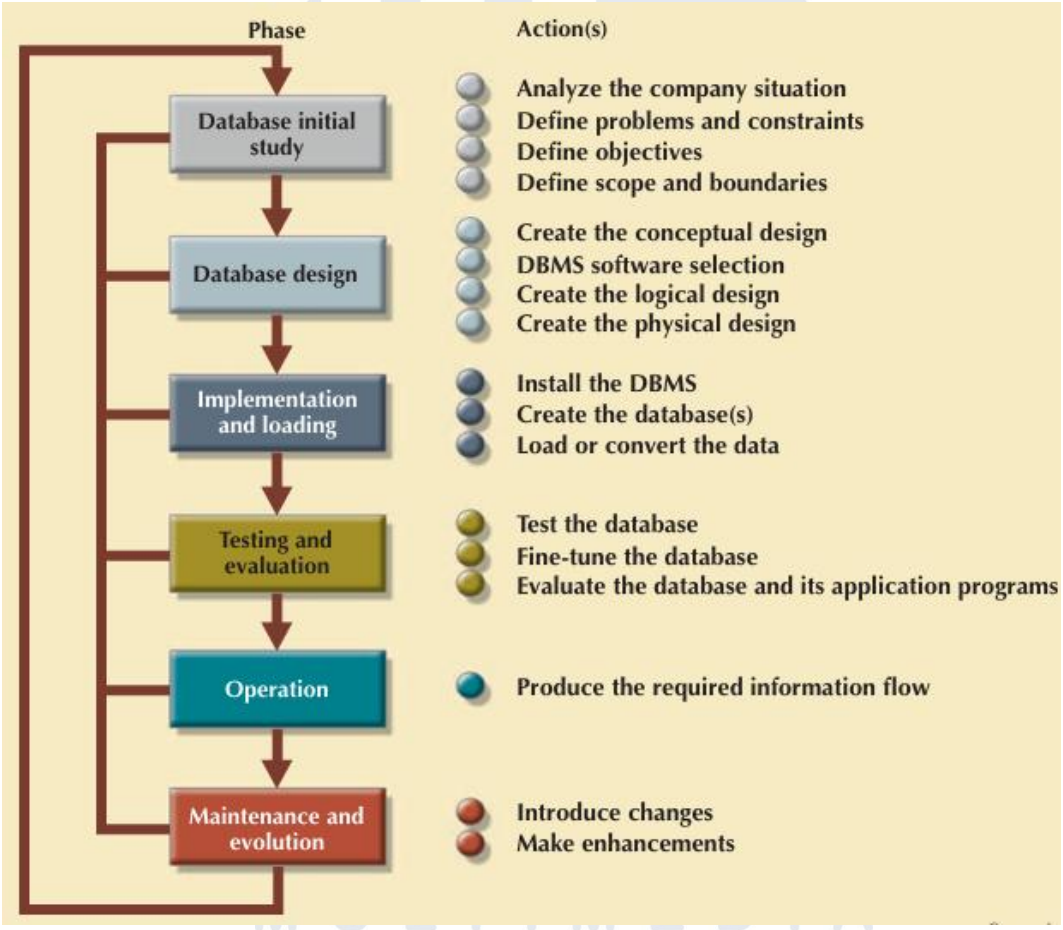
2.2.10 *User Acceptance Testing* (UAT)

User Acceptance Testing adalah pengujian yang dilakukan untuk memastikan bahwa sistem atau aplikasi yang dikembangkan memenuhi kebutuhan dan harapan pengguna akhir sebelum dirilis secara luas [48]. UAT bertujuan untuk menilai kelayakan dan fungsi perangkat lunak dari perspektif pengguna dan dapat mengidentifikasi kesalahan yang tidak terdeteksi dalam pengujian sebelumnya seperti *white box* atau *black box* testing [48]. Pada pelaksanaannya, UAT sering dibagi menjadi beberapa jenis, seperti Alpha *testing* yang dilakukan secara internal oleh pengembang untuk mengidentifikasi bug, dan *Beta testing* yang melibatkan pengguna eksternal untuk memberikan umpan balik langsung mengenai perangkat lunak [49]. Selain itu, UAT juga merupakan langkah akhir dalam siklus pengembangan aplikasi yang memungkinkan perbaikan lebih lanjut dan penyempurnaan aplikasi berdasarkan umpan balik pengguna [50].

2.3 Framework/Algoritma yang digunakan

2.3.1 *Database Development Lifecycle* (DBLC)

Database Development Lifecycle (DBLC) merupakan metode atau pendekatan yang digunakan untuk tahapan lengkap dalam pengembangan basis data, mulai dari tahap perencanaan hingga implementasi dan pemeliharaan [51]. DBLC bertujuan untuk memastikan bahwa basis data yang dibangun mampu memenuhi kebutuhan pengguna, mendukung proses bisnis, serta memiliki kinerja, integritas, dan keamanan yang optimal [52]. Tahapan-tahapan utama DBLC disajikan pada Gambar 2.4.



Gambar 2.4 Tahapan Utama DBLC
Sumber: [53]

Berdasarkan Gambar 2.4, ada 6 tahapan utama DBLC yang wajib untuk diterapkan [53], yaitu:

1. *Database Initial Study*

Tahap ini merupakan tahap awal yang berfokus pada identifikasi kebutuhan organisasi terhadap sistem basis data. Kegiatan yang dilakukan meliputi analisis permasalahan yang ada, penentuan tujuan sistem, serta identifikasi kebutuhan data dan informasi. Selain itu, dilakukan pula studi kelayakan untuk menilai apakah sistem yang akan dikembangkan layak dari segi teknis, operasional, dan ekonomis.

2. *Database Design*

Tahap perancangan basis data bertujuan untuk menghasilkan struktur basis data yang sesuai dengan kebutuhan pengguna. Proses ini mencakup perancangan konseptual, logis, dan fisik. Perancangan konseptual dilakukan dengan memodelkan data menggunakan diagram seperti *Entity Relationship Diagram* (ERD), sedangkan perancangan logis mengubah model tersebut ke dalam bentuk model relasional. Perancangan fisik berfokus pada implementasi teknis seperti pemilihan struktur penyimpanan dan indeks.

3. *Implementation and Loading*

Pada tahap ini, hasil perancangan basis data diimplementasikan ke dalam *Database Management System* (DBMS). Aktivitas yang dilakukan meliputi pembuatan tabel, penentuan relasi antar tabel, penerapan constraint, serta proses pengisian data awal (data loading).

4. *Testing and Evaluation*

Tahap pengujian dilakukan untuk memastikan bahwa sistem basis data telah berjalan sesuai dengan kebutuhan yang telah ditentukan. Pengujian mencakup validasi struktur data, pengujian kueri, serta evaluasi terhadap performa dan integritas data. Apabila ditemukan kesalahan atau ketidaksesuaian, maka dilakukan perbaikan sebelum sistem digunakan secara operasional.

5. *Operation*

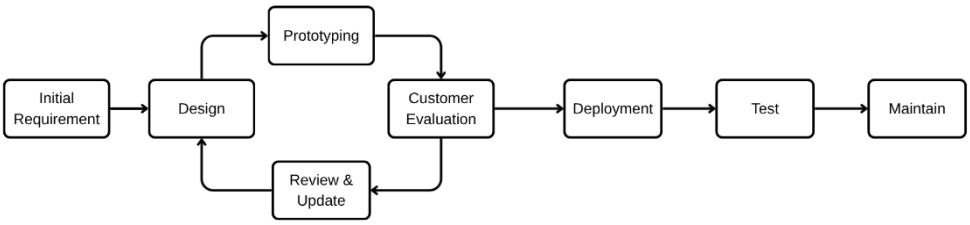
Tahap operasi merupakan fase di mana sistem basis data mulai digunakan dalam lingkungan nyata. Pada tahap ini, basis data mendukung aktivitas operasional organisasi, seperti pengolahan transaksi dan penyediaan informasi bagi pengguna.

6. *Maintenance and Evolution*

Tahap terakhir adalah pemeliharaan dan pengembangan sistem secara berkelanjutan. Kegiatan yang dilakukan meliputi perbaikan kesalahan, peningkatan performa, serta penyesuaian sistem terhadap perubahan kebutuhan organisasi. Tahap ini bersifat dinamis karena sistem basis data harus mampu beradaptasi dengan perkembangan kebutuhan pengguna.

2.3.2 *System Development Lifecycle* (SDLC) *Prototyping*

Model *prototyping* dalam SDLC merupakan pendekatan pengembangan perangkat lunak yang memfokuskan pada pembuatan versi awal sistem secara cepat sebagai representasi awal dari kebutuhan pengguna [54]. Dalam model ini, pengembang terlebih dahulu membangun *prototype*, kemudian menunjukkannya kepada pengguna untuk memperoleh umpan balik. Proses ini dilakukan secara iteratif, di mana *prototype* terus disempurnakan berdasarkan masukan pengguna hingga kebutuhan sistem menjadi lebih jelas dan sesuai dengan harapan. Pendekatan ini sangat efektif digunakan pada kondisi di mana kebutuhan sistem belum terdefinisi secara rinci atau berpotensi mengalami perubahan, serta ketika keterlibatan pengguna sejak tahap awal pengembangan menjadi faktor yang krusial [55].



Gambar 2.5 Tahapan SDLC *Prototyping*
Sumber: [56]

Gambar 2.5 menampilkan tahapan-tahapan pada model SDLC *prototyping* yang dijelaskan sebagai berikut:

1. *Initial Requirement*

   Tahap ini merupakan proses identifikasi kebutuhan awal sistem yang diperoleh melalui komunikasi dengan pengguna. Kebutuhan yang dikumpulkan masih bersifat umum dan belum terperinci secara menyeluruh.

2. *Design*

Berdasarkan kebutuhan awal, dilakukan perancangan sistem secara sederhana yang mencakup struktur dasar, alur proses, serta tampilan awal sistem sebagai dasar pembuatan *prototype*.

3. *Prototyping*

Pada tahap ini dikembangkan model awal sistem yang dapat digunakan secara terbatas. *Prototype* berfungsi sebagai representasi awal sistem untuk membantu pengguna memahami gambaran sistem yang akan dibangun.

4. *Customer Evaluation*

Prototype yang telah dibuat kemudian diuji oleh pengguna untuk memperoleh umpan balik. Evaluasi ini bertujuan untuk mengidentifikasi kesesuaian sistem dengan kebutuhan pengguna.

5. *Review & Update*

Berdasarkan hasil evaluasi, dilakukan perbaikan dan penyempurnaan terhadap *prototype* maupun desain sistem agar lebih sesuai dengan kebutuhan yang diharapkan.

6. *Deployment*

Setelah *prototype* dinyatakan sesuai, sistem dikembangkan secara penuh dan diimplementasikan pada lingkungan operasional.

7. *Testing*

Sistem yang telah diimplementasikan kemudian diuji untuk memastikan fungsionalitas berjalan dengan baik serta bebas dari kesalahan yang signifikan.

8. *Maintenance*

Tahap terakhir adalah pemeliharaan sistem yang dilakukan secara berkala untuk memperbaiki kesalahan, meningkatkan kinerja, dan menyesuaikan sistem dengan kebutuhan yang berkembang.

2.3.3 GPT-4

GPT-4 (*Generative Pre-trained Transformer* 4) merupakan model kecerdasan buatan berbasis NLP yang dikembangkan oleh OpenAI [57]. Model ini termasuk dalam kategori LLM yang dirancang untuk memahami dan menghasilkan

bahasa alami secara kontekstual berdasarkan pola yang dipelajari dari data dalam skala besar. GPT-4 memiliki karakteristik multimodal, yaitu mampu menerima berbagai jenis input seperti teks dan, pada versi tertentu, gambar, serta menghasilkan output berupa teks [58]. Model ini dibangun menggunakan arsitektur *transformer* dan dilatih melalui proses pre-training untuk memprediksi token berikutnya (kata atau simbol) dalam suatu urutan teks. Selanjutnya, model disempurnakan melalui proses *fine-tuning* dengan pendekatan *reinforcement learning from human feedback* (RLHF) untuk meningkatkan kualitas, keamanan, dan relevansi respons yang dihasilkan [59].

Dalam bidang aplikasi, GPT-4 memiliki potensi yang luas, termasuk dalam sektor pendidikan, bisnis, dan kesehatan [60]. Model ini dapat dimanfaatkan untuk mendukung pengambilan keputusan, menghasilkan laporan secara otomatis, serta meningkatkan interaksi berbasis bahasa alami melalui *chatbot*. Selain itu, model GPT-4 juga dapat diakses melalui aplikasi seperti ChatGPT maupun melalui *Application Programming Interface* (API) untuk diintegrasikan ke dalam berbagai sistem aplikasi lain. Di sisi lain, GPT-4 juga memiliki sejumlah keterbatasan. Model ini masih berpotensi menghasilkan informasi yang tidak akurat, memiliki keterbatasan dalam memahami konteks dunia nyata secara mendalam, serta bergantung pada data pelatihan yang digunakan [61]. Selain itu, terdapat berbagai tantangan terkait implementasinya, seperti potensi bias, isu privasi, serta kurangnya transparansi dalam proses pelatihan. Perbandingan GPT-4 dengan LLM lain disajikan pada Tabel 2.4.

Tabel 2.7 Perbandingan GPT-4 dengan LLM lain

Sumber: [62]

Model	**Pengembang**	**Kemampuan Penalaran dan Instruksi**	**Kemampuan Belajar dari Feedback**	**Kemampuan Pemrograman**	**Kesesuaian untuk NL2SQL**
GPT-4	OpenAI	Sangat baik	Sangat baik	Sangat baik	Sangat sesuai
GPT-3.5	OpenAI	Baik	Baik	Baik	Sesuai
Gemini Ultra	Google	Baik	Cukup baik	Baik	Sesuai
Gemini Pro	Google	Baik	Cukup baik	Cukup baik	Cukup sesuai

Claude 2	Anthropic	Baik	Baik	Baik	Sesuai
Llama 2	Meta	Cukup baik	Terbatas	Cukup baik	Cukup sesuai
Code Llama	Meta	Fokus pada kode	Terbatas	Baik	Kurang sesuai

Berdasarkan perbandingan karakteristik berbagai *Large Language Model* pada Tabel 2.4, GPT-4 dipilih dalam penelitian ini karena memiliki kemampuan yang lebih unggul dalam memahami instruksi kompleks, memanfaatkan umpan balik untuk memperbaiki hasil, serta menghasilkan keluaran yang konsisten pada berbagai tugas pemrograman dan pengolahan bahasa alami. GPT-4 memiliki performa terbaik dibandingkan model lain yang diuji, termasuk Gemini Ultra, Claude 2, GPT-3.5, Llama 2, dan Code Llama [62]. Selain itu, GPT-4 menunjukkan kemampuan yang baik dalam menangani tugas pemrograman, operasi basis data, serta memahami konteks permasalahan yang kompleks. Karakteristik tersebut menjadikan GPT-4 sesuai untuk diterapkan pada penelitian ini, khususnya dalam proses penerjemahan pertanyaan bahasa alami menjadi kueri SQL pada *chatbot* analitis yang dikembangkan.

2.3.4 *Execution Accuracy*

Execution Accuracy (EX) adalah metrik yang digunakan untuk mengevaluasi kinerja sistem NL2SQL dengan membandingkan hasil eksekusi kueri SQL yang dihasilkan oleh model dengan kueri *ground-truth* (acuan) pada basis data yang sama [8]. Jika hasil eksekusi kedua query tersebut identik, prediksi dianggap benar [63]. Namun, EX memiliki kelemahan karena kueri dengan logika yang berbeda dapat menghasilkan output yang sama, sehingga dapat menyebabkan overestimasi akurasi prediksi model [64]. *Execution accuracy* (ACC_{EX}) dapat dihitung dengan Rumus 2.1.

$$ACC_{EX} = \frac{Jumlah\ kueri\ yang\ hasil\ eksekusinya\ benar}{Jumlah\ total\ kueri\ yang\ di\ uji} \quad (1)$$

Rumus 2.1 *Execution Accuracy*

Sumber: [65]

Rumus tersebut menunjukkan bahwa EX dihitung sebagai rasio antara jumlah kueri yang menghasilkan keluaran eksekusi yang sesuai dengan *ground-truth* terhadap total keseluruhan kueri yang diuji dalam dataset. Nilai yang diperoleh merepresentasikan tingkat keberhasilan model NL2SQL dalam menghasilkan kueri SQL yang secara hasil eksekusinya setara dengan kueri referensi pada basis data yang sama. Semakin tinggi nilai EX, semakin baik kemampuan model dalam menghasilkan kueri yang mampu memberikan hasil yang benar meskipun struktur kueri yang dihasilkan tidak selalu identik secara sintaksis dengan *ground-truth*.

2.4 Tools/software yang digunakan

2.4.1 Visual Studio Code

Visual Studio Code (VS Code) adalah editor kode yang dirancang ulang dan dioptimalkan untuk membangun dan melakukan *debugging* pada aplikasi *web* dan *cloud* modern [66]. Ini adalah editor kode dan alat pengembangan lintas platform pertama yang dibuat untuk berbagai sistem operasi populer, yaitu Microsoft, Windows, Linux, dan macOS [67]. Fitur-fitur yang dimilki *VS Code* antara lain *debugging*, penyorotan sintaks, pelengkapan kode otomatis, potongan kode, *refactoring*, *auto-indentations*, pencocokan tanda kurung dan mendukung berbagai bahasa pemograman. Selain itu, VS Code memiliki integrasi Git bawaan dan editor kode ini terus berkembangan semakin canggih. Pengguna juga dapat mengubah tema tampilan pengkodean dan *shortcut keyboard* serta memasang ekstensi, preferensi, dan hal fungsionalitas lainnya [67].

2.4.2 XAMPP

XAMPP singkatan dari X untuk *CrossPlatform*, A untuk Apache, M untuk MySQL, dan PP untuk PHP dan Perl [68]. XAMPP adalah salah satu server web lintas platform yang banyak digunakan oleh pengembang untuk membuat dan menguji program mereka pada server web lokal [69]. Ini dikembangkan oleh Apache Friends, dan kode sumber aslinya dapat direvisi atau dimodifikasi sesuai kebutuhan pengembang. XAMPP memiliki kelebihan yang menjadikannya pilihan populer bagi pengembang. Perangkat lunak ini gratis dan open source, memungkinkan siapa saja untuk menggunakannya tanpa biaya. Dengan instalasi

yang mudah, XAMPP menyediakan semua komponen utama seperti Apache, MySQL/MariaDB, PHP, dan phpMyAdmin dalam satu paket, sehingga pengembang dapat langsung menjalankan server lokal tanpa konfigurasi yang rumit.

2.4.3 Laravel

Laravel adalah sebuah *framework* web berbasis PHP yang dirancang untuk memudahkan dan mempercepat pembuatan aplikasi web. Laravel menggunakan pola arsitektur *Model View Controller* (MVC) sehingga struktur kode menjadi lebih rapi, terorganisir, dan mudah dikembangkan [70]. *Framework* ini menawarkan berbagai fitur modern seperti *routing* yang sederhana dan cepat, akses ke berbagai jenis basis data melalui *database abstraction layer* dan *Eloquent ORM*, autentikasi dan otorisasi, *middleware*, sampai *templating engine* Blade untuk membangun tampilan yang *reusable* [71]. Dibandingkan PHP *native*, Laravel sangat unggul meningkatkan produktivitas pengembang, keteraturan arsitektur, dan kemudahan pengelolaan aplikasi berskala besar [72]. Laravel sudah sering diterapkan dalam berbagai studi kasus seperti sistem perpustakaan, manajemen sekolah, rumah sakit hingga *e-commerce*.

2.4.4 Pentaho Data Integration

Pentaho Data Integration adalah sebuah platform *Business Intelligence* (BI) dan analitik data yang menyediakan rangkaian alat untuk mengumpulkan, mengolah, menganalisis, dan menyajikan data dalam bentuk laporan, dashboard, dan visualisasi interaktif [73]. Pentaho mendukung proses *data integration* ETL (*Extract, Transform, Load*) untuk menarik data dari berbagai sumber (*database*, file Excel, CSV, dll.), membersihkannya, lalu menyimpannya ke dalam *data warehouse* atau sistem analitik lain. Selain itu, Pentaho memiliki fitur *reporting*, *OLAP analysis*, *data mining*, dan *dashboard* interaktif yang membantu pengguna mengambil keputusan berbasis data [74]. Pentaho tersedia dalam versi komunitas (*open source*) dan versi enterprise berbayar, sehingga cukup populer di kalangan pengembang dan perusahaan yang ingin membangun solusi BI dengan biaya lebih terjangkau namun tetap fleksibel dan dapat dikembangkan.

2.4.5 MySQL

MySQL merupakan sistem manajemen basis data relasional (*Relational Database Management System* / RDBMS) milik Oracle Corporation yang bersifat *open source* dan banyak digunakan dalam berbagai aplikasi, mulai dari sistem skala kecil hingga sistem berskala besar [75]. MySQL berfungsi untuk mengelola data dalam bentuk tabel yang saling berelasi, serta mendukung penggunaan bahasa *Structured Query Language* (SQL) untuk melakukan operasi seperti penyimpanan, pengambilan, dan pengolahan data. Selain itu, MySQL dikembangkan untuk memberikan kinerja yang tinggi, kemudahan penggunaan, serta kemampuan dalam menangani data dalam jumlah besar secara efisien.

Sebagai salah satu RDBMS yang populer, MySQL memiliki keunggulan dalam hal performa, keandalan, dan skalabilitas [76]. Sistem ini mampu menangani ribuan kueri per detik serta mendukung pengelolaan data dalam skala besar, sehingga banyak digunakan dalam berbagai sektor seperti bisnis, pendidikan, dan teknologi informasi. Selain itu, MySQL juga mendukung berbagai bahasa pemrograman seperti PHP, Java, dan Python, serta dapat berjalan pada berbagai sistem operasi, sehingga memberikan fleksibilitas tinggi dalam pengembangan aplikasi. Kemudahan penggunaan, dukungan komunitas yang luas, serta kemampuan dalam mengelola data secara efektif menjadikan MySQL sebagai pilihan yang tepat dalam pembangunan sistem yang membutuhkan pengelolaan data yang andal dan efisien

2.4.6 HeidiSQL

HeidiSQL adalah sebuah perangkat lunak open-source yang digunakan untuk mengelola dan memanipulasi basis data relasional seperti MySQL, MariaDB, PostgreSQL, dan SQL Server melalui tampilan yang sederhana dan intuitif [77]. Dengan HeidiSQL, pengguna bisa membuat, mengedit, dan menghapus tabel serta data di dalam basis data, menjalankan perintah SQL, melakukan ekspor-impor data, serta mengelola hak akses pengguna tanpa harus menulis kode secara manual di *command line*. Dalam berbagai penelitian dan pengembangan sistem informasi misalnya pada aplikasi posyandu. HeidiSQL sering dipilih karena penggunaannya yang mudah dan kemampuannya mempercepat proses pengelolaan basis data secara

efisien. Alat ini sangat membantu baik bagi pengembang maupun administrator basis data dalam membangun aplikasi berbasis data yang terstruktur dan terintegrasi.

2.4.7 Figma

Figma merupakan perangkat lunak desain antarmuka berbasis web yang digunakan untuk merancang tampilan aplikasi secara kolaboratif dan interaktif [78]. Figma memungkinkan pengembang dan desainer untuk membuat *prototype*, *wireframe*, serta desain *User Interface* (UI) dan *User Experience* (UX) dalam satu *platform* tanpa memerlukan instalasi perangkat lunak tambahan. Keunggulan utama Figma terletak pada kemampuannya dalam mendukung kolaborasi secara *real-time*, sehingga beberapa pengguna dapat bekerja pada desain yang sama secara bersamaan. Selain itu, Figma menyediakan fitur komponen, *auto layout*, serta sistem desain yang mempermudah pembuatan tampilan yang konsisten dan *scalable*.