BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Sub-bab ini menyajikan tinjauan sistematis terhadap sepuluh penelitian utama yang menjadi landasan teoretis dalam pengembangan arsitektur basis data graf hirarkis untuk sistem pembelajaran adaptif. Fokus utama dari literatur yang dipilih mencakup analisis komparatif performa basis data relasional terhadap basis data graf dalam menangani data terhubung [29], evolusi teknik *Retrieval-Augmented Generation* (RAG) yang mengintegrasikan *Knowledge Graph* untuk meningkatkan akurasi ekstraksi informasi [18], hingga implementasi model pembelajaran personalisasi berbasis struktur semantik hirarkis [30]. Melalui pemetaan penelitian terdahulu ini, dapat diidentifikasi posisi kebaruan penelitian dalam mengoptimalkan *Native Vector Indexing* di dalam satu *engine* graf untuk mengatasi tantangan latensi dan konsistensi data dalam sistem pendidikan cerdas [31].

Tabel 2.1 Hasil Tinjauan Literatur

No. Ref	Metodologi Penelitian	Framework / Algoritma	Hasil Temuan	Tantangan atau Keterbatasan	Hubungan dengan Penelitian Ini
[18]	Integrasi *Knowledge Graph* (KG) dengan *Vector Retrieval* untuk ekstraksi informasi.	HybridRAG	Meningkatkan akurasi dan efisiensi ekstraksi informasi secara signifikan dibanding *Vector* RAG murni.	Beban komputasi dan latensi tinggi jika sistem *retrieval* dilakukan secara terpisah (*decoupled*).	Menjadi acuan arsitektur hibrida, namun penelitian ini menggunakan *Native Vector Indexing* di Neo4j untuk efisiensi latensi.

No. Ref	Metodologi Penelitian	Framework / Algoritma	Hasil Temuan	Tantangan atau Keterbatasan	Hubungan dengan Penelitian Ini
[29]	Analisis perbandingan performa kueri data terhubung pada data skala besar.	Neo4j vs MySQL	Neo4j jauh lebih cepat dalam kueri relasi kompleks dengan penggunaan sumber daya yang stabil.	*Database* relasional (SQL) memerlukan proses JOIN yang sangat berat untuk data hirarkis yang mendalam.	Dasar utama pemilihan Neo4j untuk menangani struktur kurikulum yang kaya akan relasi (*connected data*).
[32]	Evaluasi eksperimental menggunakan *benchmark* LDBC SNB pada berbagai basis data graf.	Neo4j, JanusGraph, TigerGraph	Neo4j terbukti memiliki performa terbaik di hampir semua metrik (kecepatan, RAM, dan CPU).	Pengelolaan memori yang sangat besar diperlukan jika jumlah data graf mencapai jutaan *node*.	Bukti untuk menjawab pertanyaan penguji mengenai alasan pemilihan Neo4j dibanding kompetitornya.
[7]	Desain kolaborasi *multi-agent* menggunakan LLM dan KG untuk dialog edukasi.	IntelliChain (*Chain-of-Thought* + KG)	Meningkatkan reliabilitas dan akurasi jawaban dalam simulasi metode pengajaran Socratic.	Koordinasi agen yang kompleks dapat meningkatkan latensi respon sistem secara keseluruhan.	Landasan teori untuk implementasi fitur Socratic Tutor yang menggunakan graf sebagai kontrol logika jawaban.
[30]	Analisis graf semantik hirarkis untuk menangkap status kognitif siswa secara multi-level.	IMEPAL (*Hierarchical Semantic Graph*)	Berhasil memetakan jalur belajar yang adaptif dan personal berdasarkan konteks unik user.	Ketergantungan yang sangat tinggi pada kualitas awal modul pemahaman semantik teks.	Referensi dalam merancang struktur graf hirarkis untuk mendukung visualisasi Skill Tree yang dinamis.

No. Ref	Metodologi Penelitian	Framework / Algoritma	Hasil Temuan	Tantangan atau Keterbatasan	Hubungan dengan Penelitian Ini
[33]	Konstruksi model pembelajaran adaptif cerdas berbasis graf di dalam Neo4j.	KRO Model (*Knowledge-Resource-Objective*)	Efektif dalam memandu proses belajar dan mengevaluasi hasil secara terstruktur di *database*.	Kesulitan dalam melakukan otomatisasi konstruksi graf dari konten pembelajaran yang berantakan.	Digunakan untuk mendefinisikan skema *node* dan *edge* (relasi materi ke tujuan belajar) yang akan diimplementasikan.
[34]	Pengembangan *probabilistic neural framework* untuk asesmen pengetahuan adaptif.	*Hierarchical Bayesian Neural Networks*	Mengoptimalkan pemilihan pertanyaan ujian dan mengurangi waktu pengujian secara drastis.	Kompleksitas matematis yang tinggi dalam melakukan proses *inference* Bayesian secara *real-time*.	Dasar logika untuk fitur *Knowledge* Tracing yang ditanam langsung pada properti *edge* di level *database* graf.
[14]	Investigasi properti aljabar pada alat ukur kemiripan semantik hirarkis teks.	*WordNet Similarity Measures*	Menemukan batasan teoritis di mana vektor murni gagal menangkap relasi hirarki yang bersifat absolut.	Ketiadaan pemetaan logika yang deterministik (pasti) pada model probabilitas vektor.	Argumen kuat kenapa penelitian ini butuh struktur Graf (deterministik) sebagai pendamping model vektor (probabilistik).
[31]	Survei peta jalan (*roadmap*) penyatuan LLM dengan *Knowledge Graph* (KG).	*Unified LLM-KG Framework*	KG membuat sistem lebih mudah dijelaskan, sementara LLM membuat sistem lebih fleksibel dalam memahami bahasa.	Sulitnya menjaga konsistensi data graf secara otomatis saat basis pengetahuan berkembang dinamis.	Visi jangka panjang penelitian ini untuk menciptakan sistem tutor yang transparan dan dapat dijelaskan (*explainable AI*).

No. Ref	Metodologi Penelitian	Framework / Algoritma	Hasil Temuan	Tantangan atau Keterbatasan	Hubungan dengan Penelitian Ini
[17]	Kontekstualisasi LLM yang efisien menggunakan *user embeddings* untuk personalisasi.	User-LLM	Berhasil meningkatkan personalisasi respon berdasarkan preferensi dan riwayat interaksi user.	Adanya isu privasi data dan kebutuhan akan dataset aktivitas user yang besar untuk pelatihan awal.	Landasan teknis untuk mengimplementasikan fitur Persona (misal: Chef, Kuli) dalam gaya penyampaian materi.

Dari sepuluh literatur yang dikaji, terdapat tiga kluster temuan yang secara langsung membentuk landasan metodologis penelitian ini. Kluster pertama berkaitan dengan kemampuan infrastruktur basis data graf dalam menangani data terhubung berskala besar. Studi komparasi performa antara Neo4j dan MySQL menunjukkan bahwa basis data graf mampu menjalankan kueri relasi kompleks dengan waktu eksekusi yang jauh lebih stabil, terutama pada data dengan kedalaman relasi bertingkat [29]. Temuan ini diperkuat oleh evaluasi menggunakan standar *benchmark* LDBC SNB yang membuktikan Neo4j unggul dalam hampir semua metrik dibanding kompetitor sesama basis data graf [32]. Namun, kedua studi tersebut dijalankan pada konteks data jaringan sosial dan belum menyentuh skenario kurikulum pendidikan yang memiliki logika urutan belajar. Kesenjangan inilah yang menjadi dasar RQ1 penelitian ini: apakah Neo4j layak dan mampu mendukung *pipeline* RAG adaptif dalam konteks kurikulum hierarkis pemrograman Python, dengan PostgreSQL sebagai baseline perbandingan struktural.

Kluster kedua berkaitan dengan kualitas sistem *Retrieval-Augmented Generation* dalam menghasilkan konteks yang akurat dan relevan. Pendekatan *HybridRAG* yang menggabungkan pencarian vektor dengan struktur *Knowledge Graph* terbukti meningkatkan akurasi ekstraksi informasi dibandingkan RAG berbasis vektor murni, tetapi masih menghadapi kendala latensi ketika komponen graf dan vektor dioperasikan sebagai sistem terpisah [18]. Di sisi lain, keterbatasan model berbasis kesamaan vektor dalam menangkap relasi hierarki yang bersifat absolut menunjukkan bahwa pencarian semantik murni rentan merekomendasikan

materi yang tidak sesuai urutan kurikulum [14]. Temuan ini menjadi basis RQ2: apakah *Graph-Augmented RAG* yang mengintegrasikan vektor dan prasyarat kurikulum dalam satu *engine* Neo4j menghasilkan kualitas *retrieval* yang lebih baik diukur melalui metrik RAGAS dibandingkan strategi *Vector-Only RAG*.

Kluster ketiga berkaitan dengan pelacakan penguasaan materi siswa secara adaptif. Pendekatan berbasis *Hierarchical Bayesian Neural Networks* berhasil mengoptimalkan asesmen pengetahuan, namun model tersebut beroperasi secara terisolasi dari *pipeline* RAG dan memerlukan infrastruktur komputasi yang kompleks [34]. Studi terpisah mengenai konstruksi model pembelajaran adaptif dalam Neo4j menunjukkan bahwa logika pemetaan materi ke tujuan belajar dapat direpresentasikan langsung dalam skema graf [33], namun pembaruan status penguasaan masih dikelola di lapisan aplikasi. Kesenjangan ini menjadi basis RQ3: apakah implementasi BKT langsung pada properti *edge* di level basis data graf menghasilkan akurasi pelacakan penguasaan yang valid, sekaligus membuktikan bahwa logika urutan belajar dapat berjalan deterministik tanpa koordinasi lapisan aplikasi.

Dengan memetakan ketiga kluster temuan tersebut ke masing-masing rumusan masalah, penelitian ini mengisi celah yang belum dijawab oleh literatur terdahulu: menyatukan kemampuan traversal graf hirarkis, *retrieval* semantik berbasis vektor, dan pelacakan penguasaan adaptif dalam satu *engine* basis data yang terintegrasi, alih-alih menggabungkan komponen-komponen tersebut secara terpisah.

Selain kesenjangan akademis tersebut, urgensi penelitian ini juga didorong oleh kondisi adopsi *AI* yang berkembang meluas ke banyak bidang dalam beberapa tahun terakhir. Laporan *OECD Digital Education Outlook 2026* mencatat bahwa *generative AI* sudah masuk ke sistem pendidikan secara luas dan cepat, namun laporan yang sama juga menegaskan bahwa penggunaan model *AI* generik tanpa rancangan yang ditujukan khusus untuk konteks belajar berisiko hanya meningkatkan kecepatan penyelesaian tugas tanpa disertai pembelajaran yang sesungguhnya [8]. Pola yang sebanding terlihat lebih spesifik pada konteks belajar pemrograman. Survei tahunan *Stack Overflow* terhadap lebih dari 49.000 *developer* di 177 negara melaporkan bahwa 44% responden menggunakan alat bantu *AI* sebagai salah satu cara mereka belajar *coding* pada tahun 2025, naik dari 37% pada

tahun sebelumnya, namun pada periode yang sama tingkat kepercayaan responden terhadap akurasi keluaran *AI* tersebut justru turun dari 40% menjadi 29% [9]. Temuan ini sejalan dengan kekhawatiran yang telah lama disampaikan dalam literatur pendidikan, yaitu bahwa keluaran *LLM* yang terdengar meyakinkan namun keliru dapat diterima begitu saja oleh peserta belajar dan berisiko membentuk kesalahpahaman konsep yang bertahan lama, terutama pada peserta yang belum memiliki dasar untuk memverifikasi kebenaran jawaban yang mereka terima [35]. Risiko ini menjadi lebih nyata pada materi yang tersusun berjenjang seperti kurikulum pemrograman Python. Pada materi semacam ini, kesalahan urutan penyampaian, misalnya memperkenalkan satu konsep sebelum prasyaratnya benar-benar dikuasai, tidak hanya berdampak pada konsep itu sendiri, tetapi berpotensi merambat ke materi-materi lanjutan yang bergantung padanya. Dengan demikian, kebutuhan akan infrastruktur yang mampu mengunci urutan penyampaian materi secara konsisten tidak lagi sekadar menjawab celah yang ditemukan di penelitian-penelitian sebelumnya, tetapi juga merespons skala adopsi *AI* dalam pembelajaran yang terus meningkat pada saat penelitian ini dilakukan.

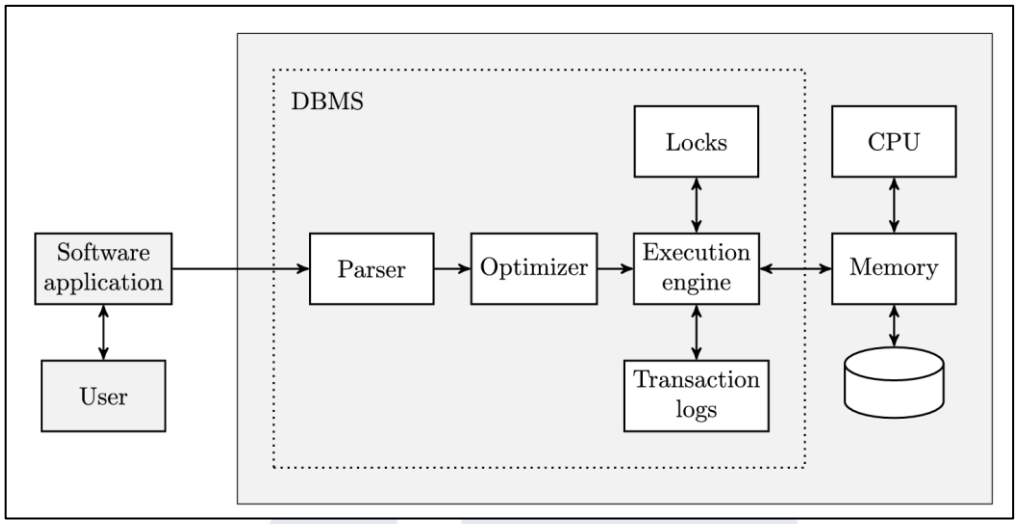
2.2 Teori Yang Berkaitan

Dalam subbab ini akan diuraikan landasan konseptual yang mendasari transformasi sistem penyimpanan data pendidikan, bergerak dari pendekatan konvensional menuju struktur yang lebih dinamis dan terhubung. Pembahasan diawali dengan pemetaan evolusi Sistem Manajemen Basis Data (DBMS) serta tantangan krusial dalam pengelolaan *unstructured data* yang kini mendominasi ekosistem pendidikan digital. Selanjutnya, analisis difokuskan pada kemampuan arsitektur *Graph database* dan *Knowledge Graph* dalam merepresentasikan kompleksitas hubungan materi, sebelum disintesiskan dengan prinsip-prinsip pembelajaran modern seperti *Adaptive Learning* dan metode *Socratic*. Kerangka teoritis ini disusun untuk memberikan alasan ilmiah bahwa efektivitas personalisasi pembelajaran sangat bergantung pada bagaimana data pengetahuan diorganisir secara fundamental.

2.2.1 Manajemen Basis Data dan Data Tidak Terstruktur

Sistem manajemen basis data atau *Database Management System* (DBMS) merupakan komponen penting dalam arsitektur perangkat lunak modern yang berfungsi sebagai fondasi utama dalam pengelolaan dan penyimpanan data secara terstruktur. Peran utama DBMS telah berkembang melampaui fungsi penyimpanan sederhana, di mana sistem ini kini menjadi alat dalam mengotomasi serta mendokumentasikan proses bisnis yang kompleks melalui integrasi sistem berbasis web. Dengan menerapkan metodologi pengumpulan informasi yang sistematis, DBMS memungkinkan transformasi alur kerja menjadi lebih terorganisir dan efisien. Jadi DBMS bukan lagi sekadar gudang data statis, melainkan sebuah mesin strategis yang menentukan efektivitas otomatisasi informasi dalam mendukung fungsionalitas sistem secara menyeluruh [36].

Visualisasi pada gambar 2.1 merepresentasikan klasifikasi dan ragam arsitektur DBMS yang menjadi fokus dalam studi komparasi performa pada literatur terkini [36]. Gambar tersebut merinci berbagai jenis sistem, mulai dari basis data relasional hingga NoSQL dan NewSQL, yang masing-masing dirancang untuk menangani karakteristik beban kerja yang berbeda. Penjabaran klasifikasi ini sangat penting guna memahami spektrum teknologi basis data yang tersedia, sekaligus menjadi dasar untuk mengevaluasi efisiensi setiap sistem dalam skenario penggunaan tertentu [36]. Melalui pemetaan ini, dapat terlihat bagaimana perkembangan arsitektur DBMS berusaha menjawab tantangan kebutuhan data yang semakin spesifik dan masif di industri.



Gambar 2.1 Arsitektur DBMS. [36]

Efisiensi dalam manajemen data merupakan aspek yang sangat krusial karena kemampuan sistem untuk merespons dengan cepat memberikan manfaat langsung bagi pengguna sekaligus menjamin efisiensi biaya bagi penyedia layanan [36]. Namun, pencapaian efisiensi yang optimal sering kali terkendala oleh metodologi pengujian performa saat ini, di mana banyak studi komparasi dilakukan dengan cara yang tidak mencerminkan kasus penggunaan di dunia nyata [36]. Selain itu, minimnya detail teknis dalam pelaporan pengujian menyebabkan hasil tersebut sulit untuk direplikasi atau dijadikan dasar penarikan kesimpulan yang valid mengenai keunggulan suatu sistem basis data [36]. Pengukuran performa DBMS memerlukan pendekatan yang lebih transparan dan realistis, karena efisiensi yang sesungguhnya harus diukur dari kemampuan sistem dalam menangani beban kerja nyata secara konsisten.

Tantangan manajemen data dalam skala industri kini semakin kompleks seiring dengan tuntutan sistem terhadap volume, variasi termasuk masifnya data tidak terstruktur (*unstructured data*) kecepatan, serta kualitas data (*Volume, Variety, Velocity, Quality*) yang tidak dapat dikompromikan [37]. Hal tersebut menjadi sangat krusial terutama pada pengembangan *deep learning* yang membutuhkan pasokan data berkualitas tinggi agar sistem tetap dapat dikembangkan dan dipelihara secara optimal [37]. Guna mengatasi kompleksitas format data tersebut, arsitektur sistem modern mulai mengadopsi spesialisasi sistem basis data yang disesuaikan secara khusus dengan beban kerja (*workload*) dan format data dari

setiap layanan untuk mencapai skalabilitas serta isolasi data yang kuat [38]. Keberagaman karakteristik data modern yang didominasi oleh format tidak terstruktur mengharuskan pemilihan arsitektur basis data yang terspesialisasi, karena penggunaan sistem konvensional yang bersifat umum tidak lagi memadai dalam menjamin kualitas dan performa sistem pada beban kerja yang sangat spesifik.

Dari keempat dimensi tantangan data modern tersebut, tantangan yang paling relevan dalam konteks penelitian ini adalah keberagaman format atau *variety*, khususnya dalam bentuk data tidak terstruktur yang dihasilkan dari ekosistem pendidikan. Jenis data ini memerlukan pendekatan tersendiri karena karakteristiknya yang berbeda secara fundamental dari data terstruktur yang selama ini menjadi ranah utama DBMS konvensional.

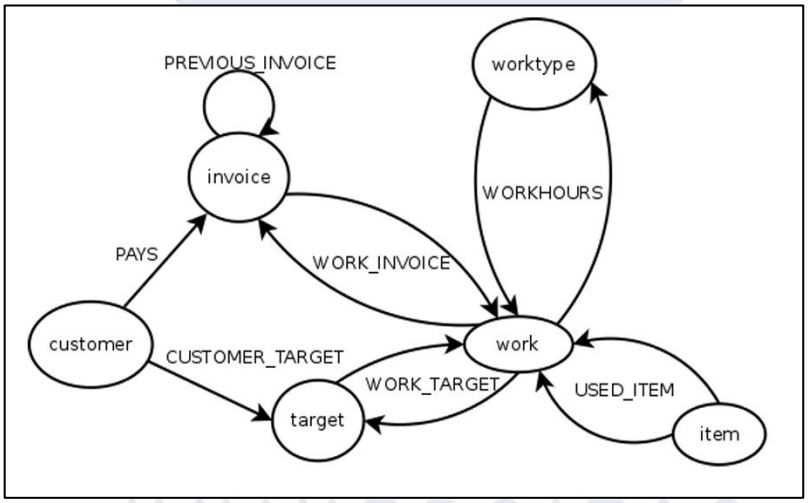
Data tidak terstruktur (*unstructured data*) dalam ekosistem pendidikan merupakan kumpulan data besar yang beragam dan tidak memiliki format tetap, seperti ulasan mahasiswa, opini terhadap materi pembelajaran, hingga teks dari media sosial [39], [40]. Karakteristik data ini menjadi komponen inti dalam *Educational Big Data Analytics* karena mampu merepresentasikan persepsi dan interaksi nyata dari para *stakeholder* di lingkungan belajar [41]. Namun, karena volumenya yang masif dan strukturnya yang kompleks, data ini sangat sulit untuk dikelola atau dianotasi secara manual tanpa bantuan teknologi yang mumpuni [39], [40]. Untuk mengelola dan memanfaatkan data ini secara optimal, diperlukan integrasi antara *Artificial Intelligence* (AI) dan *Natural Language Processing* (NLP) guna menangkap detail kontekstual serta logika ketergantungan informasi yang sering terlewat oleh metode konvensional [39]. Penerapan NLP dalam sistem interaktif berbasis kecerdasan buatan telah terbukti efektif, antara lain pada pembangunan chatbot layanan pelanggan yang memanfaatkan platform NLP untuk memproses dan memindai setiap pertanyaan pengguna secara otomatis berdasarkan konteks dialog [42].

Penggunaan *tools* modern seperti Python dan API dari model bahasa besar (LLM) memungkinkan pengolahan data teks mentah menjadi *insight* yang berharga untuk meningkatkan pengalaman belajar serta mengasah kemampuan berpikir kritis mahasiswa [41]. Melalui proses pemodelan yang tepat, data tidak terstruktur ini

tidak hanya berfungsi sebagai alat evaluasi kurikulum pembelajaran bagi institusi, tetapi juga membantu mahasiswa memahami bagaimana sebuah sistem cerdas bekerja serta cara mengantisipasi potensi bias dalam pengolahan informasi digital [39], [40].

2.2.2 Graph database dan *Knowledge Graph*

Penggunaan *graph database* dalam arsitektur data modern muncul sebagai solusi atas keterbatasan *database* relasional (RDBMS) dalam menangani data dengan relasi yang sangat kompleks. Masalah utama pada RDBMS terletak pada ketergantungan terhadap operasi "JOIN" antartabel yang membutuhkan sumber daya komputasi tinggi seiring bertambahnya volume dan kedalaman hubungan data [43]. Sebagai NoSQL yang berorientasi pada hubungan, *graph database* menggeser fokus dari penyimpanan tabel statis menjadi struktur yang dinamis, di mana data disimpan sebagai *nodes* dan *edges* untuk mempermudah representasi jaringan informasi yang saling terhubung secara masif [43], [44]

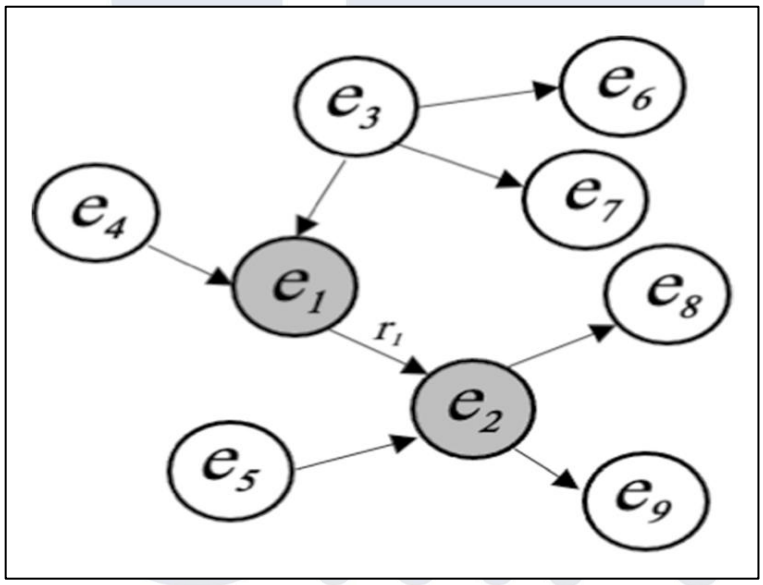


Gambar 2.2 Contoh Struktur *Graph database*. [43]

Berdasarkan struktur pada Gambar 2.2, perbedaan mendasar terletak pada cara sistem melakukan navigasi data, jika RDBMS melakukan pencarian melalui kunci asing (*foreign keys*) yang memicu proses penggabungan tabel yang lambat, maka *graph database* menggunakan mekanisme *index-free adjacency*. Dalam model ini, setiap hubungan atau *link* antar-data diimplementasikan sebagai *pointer* fisik, sehingga proses penelusuran (*traversal*) dapat dilakukan dalam waktu yang konstan

(*constant time*) tanpa terpengaruh oleh total ukuran dataset. Efisiensi ini menjadi alasan utama mengapa model *graph* jauh lebih unggul dalam menjalankan *complex queries* dibandingkan sistem *database* tradisional [43].

Perkembangan lebih lanjut dari struktur *graph* ini mengarah pada pembentukan *Knowledge Graph* (KG), yang tidak hanya berfungsi sebagai wadah penyimpanan tetapi juga mengorganisir pengetahuan secara struktural mirip dengan pola kognitif manusia. KG merepresentasikan konsep dan entitas dunia nyata beserta hubungan mereka dalam format yang dapat dipahami secara mendalam oleh mesin (*machine-readable*). Dengan memanfaatkan *Knowledge Graph*, sistem kecerdasan buatan dapat mengelola informasi yang bersifat heterogen dengan cara yang lebih terintegrasi, yang sangat krusial untuk aplikasi tingkat lanjut seperti pencarian semantik dan sistem pakar [45].



Gambar 2.3 Representasi *Knowledge* Graph. [46]

Berdasarkan representasi pada Gambar 2.3, menggambarkan bagaimana *Knowledge Graph* mengintegrasikan berbagai lapisan data. Simbol e merepresentasikan entitas sebagai objek tunggal dalam sistem, sedangkan garis (r) antar lingkaran melambangkan relasi yang merupakan interaksi antar entitas tersebut [46]. Angka-angka yang menyertai simbol tersebut (seperti $e_1\ e_2\ r_i$) berfungsi sebagai indeks identitas unik untuk membedakan ribuan entitas dan jenis hubungan yang heterogen dalam dataset pendidikan [46]. Dalam konteks

knowledge reasoning, format *triple* ini memungkinkan sistem untuk memetakan informasi ke dalam ruang vektor berdimensi rendah (*embeddings*). Hal ini sangat krusial agar mesin tidak hanya mengenali teks, tetapi mampu melakukan operasi matematika untuk memprediksi hubungan baru yang belum terpetakan secara eksplisit, atau yang dikenal dengan istilah *knowledge graph completion* [45], [46].

Secara keseluruhan, integrasi antara *graph database* dan *Knowledge Graph* menjadi fondasi krusial bagi pengembangan sistem *Adaptive Learning* yang dioptimasi dengan *Retrieval-Augmented Generation* (RAG). Melalui kemampuan *knowledge reasoning*, sistem mampu menarik kesimpulan dari data pendidikan yang kompleks untuk menentukan jalur pembelajaran yang paling personal bagi pengguna [45]. Dengan mengatasi berbagai tantangan teknis dalam representasi pengetahuan, kombinasi teknologi ini memastikan bahwa sistem RAG tidak hanya sekadar mengambil data, tetapi mampu memberikan konteks yang akurat dan relevan sesuai dengan kebutuhan logika urutan belajar [46].

2.2.3 Adaptive Learning dan Metode *Socratic*

Adaptive Learning atau pembelajaran adaptif didefinisikan sebagai metode pembelajaran cerdas yang secara dinamis menyesuaikan konten, kecepatan, dan penyampaian materi berdasarkan kebutuhan unik setiap individu [27]. Berbeda dengan metode tradisional yang menerapkan pendekatan "satu ukuran untuk semua" (*one-size-fits-all*), sistem ini menggunakan analisis data dan algoritma kecerdasan buatan (AI) untuk mengenali kekuatan, kelemahan, dan preferensi gaya belajar mahasiswa secara *real-time* [1], [27]. Tujuan utamanya adalah menempatkan mahasiswa sebagai pusat dari proses pendidikan (*learner-centered education*), di mana sistem secara otomatis memberikan dukungan yang dipersonalisasi untuk meningkatkan performa belajar mereka [4].

Pemilihan metode Socratic sebagai pendekatan tutoring dalam penelitian ini didasarkan pada bukti empiris bahwa metode ini secara signifikan lebih efektif dalam membangun pemahaman konseptual jangka panjang dibandingkan dengan pendekatan direct instruction [7]. Penelitian menunjukkan bahwa mahasiswa yang dibimbing melalui pertanyaan pemantik cenderung mengembangkan kemampuan problem-solving dan penalaran logis yang lebih kuat karena mereka dipaksa untuk

mengonstruksi pemahaman secara mandiri, bukan sekadar menerima jawaban [16]. Hal ini sangat relevan dalam konteks pembelajaran pemrograman, di mana pemahaman mendalam terhadap logika dan struktur kode jauh lebih krusial daripada kemampuan menghafal sintaksis semata. Selain itu, pendekatan ini secara langsung menghilangkan risiko perilaku "salin-tempel" yang umum terjadi ketika sistem AI memberikan solusi akhir secara langsung.

2.3 Framework & Algoritma yang digunakan

Sub-bab ini menguraikan spesifikasi kerangka kerja (*framework*) serta algoritma utama yang menjadi fondasi teknis dalam pembangunan arsitektur sistem. Pemilihan seluruh komponen teknologi ini dilakukan melalui proses evaluasi yang menyeluruh dengan merujuk pada temuan dan rekomendasi dari tinjauan literatur yang telah dilakukan sebelumnya. Setiap alat, mulai dari model graf hingga metodologi pengembangan sistem, dipilih secara selektif berdasarkan kompatibilitasnya dalam mendukung integrasi *Hierarchical Graph database* dengan *Retrieval-Augmented Generation* (RAG). Sinergi antar-komponen ini dirancang secara spesifik untuk menjawab kebutuhan sistem pembelajaran adaptif yang menuntut efisiensi performa tinggi, validitas struktur data yang ketat, serta skalabilitas dalam menangani logika urutan belajar yang kompleks.

2.3.1 Model Data dan Indeks Vektor Neo4j

Inti dari arsitektur penyimpanan Neo4j adalah model *Labeled Property Graph* (LPG) yang terdiri dari komponen utama berupa *node*, *relationship*, dan *property* [44]. Dalam model ini, *node* merepresentasikan entitas, sementara relasi atau *edge* berfungsi menghubungkan antar entitas tersebut secara berarah. Keunggulan LPG terletak pada fleksibilitasnya dalam menyematkan informasi tambahan berupa pasangan kunci-nilai (*key-value pairs*) pada setiap elemen graf, serta penggunaan label untuk kategorisasi data yang lebih efisien [47]. Sintesis dari model ini memberikan kemampuan bagi pengembang untuk memetakan domain masalah dunia nyata ke dalam struktur data secara lebih intuitif dan fleksibel tanpa perlu melakukan proses normalisasi data yang rumit.

Implementasi model LPG dalam ekosistem Neo4j terbukti memberikan keunggulan performa yang signifikan, terutama pada operasi navigasi data yang

memiliki kedalaman relasi yang tinggi [43]. Berbeda dengan *database* SQL yang memerlukan proses *join* tabel yang sangat berat secara komputasi, model graf ini memungkinkan pencarian pola data dilakukan dengan waktu eksekusi yang lebih konstan dan cepat [36]. Hal ini menunjukkan bahwa integrasi antara model LPG dan mesin *database* Neo4j merupakan langkah krusial untuk membangun sistem modern yang menuntut skalabilitas serta responsivitas tinggi dalam mengolah informasi yang saling berkaitan secara dinamis.

Model LPG menjawab kebutuhan penelusuran relasi yang efisien, tetapi sistem pembelajaran adaptif juga memerlukan pencarian berbasis makna terhadap materi. Untuk kebutuhan ini, Neo4j menghadirkan fitur *native vector indexing* yang ditenagai oleh pustaka pencarian Apache Lucene. Fitur ini memungkinkan pengguna untuk menyimpan *embedding* vektor secara langsung sebagai properti pada *node* ataupun *relationship* di dalam graf. Untuk menyeimbangkan antara kecepatan pencarian dan akurasi hasil pada dataset yang besar, Neo4j menggunakan struktur indeks berbasis *Hierarchical Navigable Small Worlds* (HNSW), di mana pengguna dapat mengatur dimensi vektor dan fungsi kemiripan sesuai kebutuhan data yang diproses [48].

Hubungan antara konsep *Vector Database* dengan Neo4j terletak pada kemampuan Neo4j untuk menjalankan fungsi VDBMS secara terintegrasi, yang mendukung apa yang disebut sebagai *hybrid queries* [49]. Dengan adanya indeks vektor bawaan (*native*), sistem tidak hanya mampu melakukan pencarian berdasarkan kemiripan vektor semata, tetapi juga dapat menggabungkannya dengan filter berbasis metadata atau struktur graf dalam satu kali eksekusi kueri [48]. Hal ini memungkinkan sistem untuk memproses data yang kompleks dengan lebih efisien karena tidak memerlukan perpindahan data antar sistem yang berbeda [49].

Kaitannya dengan penelitian ini, penggunaan *native vector indexing* di Neo4j difungsikan sebagai solusi arsitektur *single engine* untuk mengatasi masalah latensi yang sering terjadi pada sistem pembelajaran adaptif yang menggunakan *database* terpisah. Dengan menyatukan penyimpanan vektor dan logika graf hierarkis dalam satu tempat, sistem dapat melakukan *Retrieval-Augmented Generation* (RAG) yang tidak hanya akurat secara semantik, tetapi juga patuh pada aturan prasyarat (*prerequisite*) materi yang tersimpan dalam graf. Pendekatan ini memastikan bahwa

rekomendasi materi yang diberikan oleh AI tetap berada dalam jalur kurikulum yang benar dan meminimalisir risiko halusinasi urutan pembelajaran. Kemampuan *single-engine* inilah yang menjadi objek pengujian pada RQ1, di mana latensi dan konsistensi traversal prasyarat Neo4j diukur secara komparatif terhadap PostgreSQL untuk membuktikan kelayakannya sebagai fondasi *pipeline* RAG adaptif.

2.3.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) merupakan sebuah teknik pengembangan kecerdasan buatan yang dirancang untuk mengatasi keterbatasan mendasar pada *Large Language Models* (LLM), seperti kecenderungan memberikan informasi yang tidak akurat atau halusinasi serta terbatasnya pengetahuan internal model yang sering kali sudah kedaluwarsa. Secara konseptual, sistem ini bekerja dengan mengintegrasikan kemampuan model bahasa dalam membuat teks dengan mekanisme pencarian informasi, di mana sistem akan terlebih dahulu menelusuri dan mengambil data relevan dari sumber eksternal yang terpercaya sebelum memproses pertanyaan pengguna. Informasi yang diperoleh dari pencarian tersebut kemudian digabungkan dengan pertanyaan awal sebagai konteks tambahan, yang berfungsi sebagai referensi bagi model untuk menghasilkan jawaban yang lebih berkualitas dan faktual. Pendekatan ini dinilai sangat efisien karena memungkinkan model untuk menyajikan informasi terkini tanpa harus melalui proses pelatihan ulang yang memakan waktu dan biaya besar, sehingga sangat efektif untuk meningkatkan akurasi dalam berbagai tugas pemrosesan bahasa [50].

Penerapan RAG pada penelitian ini dikembangkan lebih lanjut melalui integrasi dengan arsitektur *Hierarchical Graph database* menggunakan fitur *native vector indexing* pada Neo4j. Berbeda dengan implementasi standar yang hanya berfokus pada kemiripan semantik teks, mekanisme ini mengikat proses *retrieval* pada aturan prasyarat (*prerequisite*) kurikulum yang tertanam dalam struktur graf. Pendekatan ini bertujuan untuk memastikan bahwa setiap materi yang diekstraksi dan disajikan oleh sistem tidak hanya faktual, tetapi juga sesuai urutan kurikulum dan sesuai dengan status kompetensi pengguna, sehingga secara efektif mengurangi risiko

halusinasi urutan pembelajaran (*sequence hallucination*) yang sering terjadi pada model probabilistik murni. Efektivitas pendekatan ini diuji secara kuantitatif pada RQ2 menggunakan framework RAGAS, dengan membandingkan skor *Faithfulness*, *Context Precision*, dan *Answer Relevance* antara strategi *Graph-Augmented RAG* dan *Vector-Only RAG* sebagai baseline.

Retrieval-Augmented Generation Assessment (RAGAS) merupakan *framework* evaluasi tanpa referensi (*reference-free*) yang dirancang khusus untuk mengukur kualitas sistem RAG secara otomatis menggunakan LLM sebagai evaluator. Framework ini mengatasi keterbatasan evaluasi manual yang tidak skalabel dengan menyediakan metrik kuantitatif yang dapat direproduksi [51].

Terdapat tiga metrik utama yang digunakan dalam penelitian ini. Pertama, *Faithfulness* mengukur sejauh mana setiap klaim dalam jawaban yang dihasilkan sistem dapat diverifikasi kebenarannya berdasarkan konteks yang diambil (*retrieved context*), dengan nilai berkisar antara 0 hingga 1 di mana nilai lebih tinggi menandakan minimnya halusinasi faktual. Kedua, *Context Precision* mengukur proporsi konteks yang diambil yang benar-benar relevan terhadap pertanyaan yang diajukan, sehingga mengevaluasi kemampuan sistem dalam memfilter informasi yang tidak diperlukan. Ketiga, *Answer Relevance* mengukur seberapa relevan jawaban yang dihasilkan terhadap pertanyaan asli pengguna, dihitung melalui rata-rata *cosine similarity* antara vektor pertanyaan asli dengan pertanyaan-pertanyaan yang di-*generate* ulang oleh LLM berdasarkan jawaban sistem [51].

Dalam penelitian ini, RAGAS digunakan untuk mengevaluasi sejauh mana sistem *Graph-Augmented RAG* mampu menghasilkan konteks yang relevan dan akurat dari sisi pembelajaran pada materi pemrograman Python yang bersifat hierarkis. Pengukuran dilakukan dengan *Vector-Only RAG* sebagai baseline referensi struktural untuk memberikan konteks validasi terhadap kontribusi struktur graf hirarkis, bukan sebagai objek perbandingan utama penelitian.

2.3.3 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) adalah salah satu metode pemodelan siswa yang paling populer digunakan dalam pengembangan sistem pembelajaran adaptif atau *Intelligent Tutoring Systems*. Pada dasarnya, tujuan utama dari BKT adalah

untuk memperkirakan keadaan pengetahuan (*knowledge state*) siswa secara dinamis dari waktu ke waktu berdasarkan interaksi mereka, seperti jawaban benar atau salah pada serangkaian soal latihan [23]. Konsep ini pertama kali diperkenalkan oleh Corbett dan Anderson pada tahun 1994, di mana pengetahuan dimodelkan sebagai variabel biner; artinya, seorang siswa dianggap berada dalam salah satu dari dua kondisi: sudah menguasai (*learned*) atau belum menguasai (*unlearned*) suatu keterampilan atau konsep tertentu [24]. Dengan menerapkan metode ini, sistem pendidikan dapat memprediksi apakah siswa akan mampu menjawab soal berikutnya dengan benar, sehingga materi pembelajaran dapat disesuaikan secara personal dengan kebutuhan individu siswa tersebut [23].

Secara teknis, BKT bekerja sebagai kasus khusus dari *Hidden Markov Model* yang memisahkan antara kondisi pengetahuan siswa yang tidak terlihat (*latent*) dengan kinerja mereka yang terlihat (*observable*) saat menjawab soal. Untuk melakukan estimasi ini, BKT menggunakan empat parameter probabilitas utama yang terus diperbarui. Parameter pertama adalah *Initial Knowledge*, yaitu kemungkinan siswa sudah mengetahui materi tersebut sebelum mulai mengerjakan latihan. Parameter kedua adalah *Learning Rate*, yang menggambarkan peluang siswa untuk beralih dari kondisi tidak tahu menjadi tahu setelah mendapatkan kesempatan belajar [23], [24]. Selain itu, model ini juga mempertimbangkan kemungkinan terjadinya kesalahan melalui parameter *Slip*, yaitu ketika siswa yang sebenarnya sudah paham melakukan kesalahan dalam menjawab, dan parameter *Guess*, yaitu ketika siswa yang belum paham berhasil menjawab benar karena menebak [24].

Proses pembaruan probabilitas dalam BKT dilakukan menggunakan prinsip inferensi Bayesian. Setiap kali siswa memberikan respon terhadap sebuah soal, sistem akan menghitung ulang estimasi penguasaan keterampilan mereka saat itu. Lebih lanjut, untuk memodelkan evolusi status penguasaan (*mastery state*) pengguna secara dinamis, aturan pembaruan (*update rule*) BKT diaplikasikan dengan persamaan matematika berikut:

$$P(L_t) = P(L_{t-1}|Action) + (1 - P(L_{t-1}|Action)) \cdot P(T)$$

Berdasarkan persamaan di atas, $P(L_t)$ merupakan probabilitas penguasaan materi pada waktu t. Sementara itu, $P(T)$ adalah Probabilitas Transisi (*Learning Rate*), yang merepresentasikan besaran kemungkinan seorang pengguna mengalami transisi dari kondisi belum menguasai menjadi menguasai (*mastery state*) setelah melalui satu kali interaksi pembelajaran.

Dalam model BKT standar, terdapat asumsi dasar bahwa proses belajar berjalan satu arah, yang berarti sekali siswa menguasai suatu keterampilan, mereka dianggap tidak akan melupakannya; oleh karena itu, probabilitas *forgetting* dalam model standar ini diasumsikan bernilai nol [23], [24]. Pendekatan ini memungkinkan sistem untuk melacak kemajuan belajar siswa secara berkesinambungan dan menentukan kapan seorang siswa dianggap telah mencapai ketuntasan (*mastery*) pada suatu topik tertentu [23].

Dalam konteks penelitian *Educational Data Mining*, BKT memiliki keunggulan tersendiri dibandingkan metode pengukuran lain seperti *Item Response Theory* (IRT). Jika IRT cenderung melihat kemampuan siswa sebagai sesuatu yang statis atau hanya mengukur level kemampuan pada satu titik waktu, BKT justru berfokus pada pelacakan proses belajar dan perubahan pengetahuan siswa seiring berjalannya waktu [23]. Meskipun BKT memiliki penyederhanaan model seperti anggapan bahwa pengetahuan bersifat biner metode ini tetap menjadi standar yang kuat karena parameternya yang mudah diinterpretasikan. Hal ini memungkinkan pengembang sistem untuk mendiagnosis perilaku belajar siswa, seperti seberapa cepat mereka belajar atau seberapa sering mereka menebak jawaban, serta memberikan intervensi pendidikan yang lebih tepat sasaran [24].

Dalam penelitian ini, keempat parameter BKT tersebut tidak dikelola di lapisan aplikasi, melainkan ditanamkan langsung sebagai properti pada *edge* MASTERY_STATE di dalam graf Neo4j. Validitas pendekatan ini diuji pada RQ3, yang mengukur apakah logika BKT yang berjalan di level basis data menghasilkan akurasi pelacakan penguasaan yang setara dengan implementasi konvensional, sekaligus membuktikan sifat deterministiknya.

2.3.4 Database System Development Life Cycle

Database System Development Life Cycle (DBSDLC) merupakan metodologi pengembangan basis data yang mencakup 11 tahapan sistematis, mulai dari perencanaan hingga pemeliharaan, untuk memastikan basis data yang dibangun memenuhi kebutuhan fungsional dan non-fungsional sistem secara menyeluruh [52]. Berbeda dengan *Database Life Cycle* (DBLC) konvensional yang berfokus pada perancangan skema relasional, DBSDLC dirancang untuk memenuhi kebutuhan berbagai jenis arsitektur basis data modern, termasuk basis data graf (*graph database*) [52].

Dalam penelitian ini, DBSDLC diadaptasi dengan menggantikan konstruk relasional dengan primitif *Labeled Property Graph* (LPG) pada setiap tahapan utamanya. Tahap *database planning* mendefinisikan tujuan arsitektur graf hirarkis sebagai fondasi sistem RAG adaptif. Tahap *system definition* menentukan batasan dan ruang lingkup tiga lapisan graf: lapisan Domain, Concept, dan *Topic* beserta relasi PREREQUISITE-nya. Tahap *conceptual design* menghasilkan model abstrak node dan edge tanpa mempertimbangkan implementasi fisik. Tahap *logical design* memetakan model konseptual ke dalam skema *Labeled Property Graph* dengan atribut dan tipe relasi yang spesifik. Tahap *physical design* dan *implementation* mentransformasikan desain logis ke dalam struktur Neo4j yang aktual, mencakup pembuatan indeks HNSW untuk *vector retrieval* dan konfigurasi properti BKT pada edge MASTERY_STATE [52].

2.3.5 Web Framework

Web framework adalah kerangka kerja perangkat lunak yang menyediakan struktur dan komponen siap pakai untuk membangun aplikasi berbasis web secara efisien. Dalam pengembangan sistem berbasis kecerdasan buatan modern, arsitektur aplikasi umumnya dibagi menjadi dua lapisan yang bekerja secara terpisah: lapisan *backend* yang menangani logika pemrosesan data dan komunikasi dengan basis data, serta lapisan *frontend* yang menyajikan antarmuka interaktif kepada pengguna. Pemisahan ini memungkinkan masing-masing lapisan dikembangkan dan diuji secara mandiri tanpa saling bergantung satu sama lain [53].

Lapisan *backend* dalam penelitian ini dibangun menggunakan FastAPI, sebuah *web framework* Python yang dirancang untuk membuat antarmuka pemrograman aplikasi atau *Application Programming Interface* (API) yang cepat dan ringan. FastAPI mendukung pemrosesan secara asinkronus, artinya sistem dapat menangani banyak permintaan secara bersamaan tanpa harus menunggu satu permintaan selesai sebelum memproses yang lain. Dibandingkan dengan *framework* Python lain seperti Flask dan Django, FastAPI terbukti mencatatkan kecepatan tertinggi dalam pengujian komparatif pada berbagai jenis operasi [54]. Dalam penelitian ini, FastAPI berperan sebagai pusat koordinasi yang menerima permintaan dari antarmuka pengguna, menjalankan kueri ke Neo4j untuk memeriksa jalur prasyarat dan memperbarui status penguasaan BKT, memanggil model bahasa Qwen2.5:7b melalui Ollama, lalu mengirimkan respons tutor adaptif kembali ke pengguna secara bertahap menggunakan *Server-Sent Events* (SSE).

Lapisan *frontend* dibangun menggunakan React 19 sebagai pustaka pembangunan antarmuka berbasis komponen, Vite 6 sebagai alat bantu pengembangan dan pembangunan aplikasi, serta Nginx sebagai *web server* untuk mengelola pengiriman berkas aplikasi di dalam lingkungan Docker. React bekerja dengan cara memisahkan tampilan antarmuka menjadi komponen-komponen kecil yang dapat digunakan ulang. Setiap kali ada perubahan data, seperti saat respons tutor diterima secara bertahap, React hanya memperbarui bagian tampilan yang berubah tanpa memuat ulang seluruh halaman [53]. Vite dipilih karena mampu menjalankan server pengembangan hampir seketika dengan memanfaatkan kemampuan *browser* modern untuk memuat modul JavaScript secara langsung, tanpa perlu memproses seluruh kode aplikasi terlebih dahulu [55]. Nginx bertugas menyajikan berkas hasil *build* dari Vite kepada pengguna sekaligus meneruskan permintaan API ke layanan *backend* [56]. Ketiga komponen ini dikemas dalam satu wadah Docker tersendiri dan berkomunikasi dengan layanan *backend* melalui REST API dan SSE, sesuai dengan pola arsitektur *microservices* yang diterapkan pada keseluruhan sistem.

2.3.6 LaBSE

Language-Agnostic BERT Sentence Embedding (LaBSE) adalah model pembuat vektor kalimat yang dikembangkan oleh Google dan dirancang untuk bekerja lintas bahasa tanpa terikat pada satu bahasa tertentu. Model ini dibangun di atas arsitektur BERT menggunakan kerangka *dual encoder*, yaitu dua buah pengkode (*encoder*) berbagi bobot yang masing-masing menerima kalimat sumber dan kalimat target secara terpisah. Representasi akhir setiap kalimat diambil dari lapisan terakhir token khusus, menghasilkan vektor berdimensi 768 yang merepresentasikan makna keseluruhan kalimat tersebut. Model ini dilatih dengan menggabungkan beberapa teknik sekaligus, yaitu *masked language modeling* untuk pemahaman bahasa, *translation language modeling* untuk keselarasan lintas bahasa, dan *additive margin softmax* sebagai fungsi kerugian yang mempertegas jarak antara pasangan kalimat yang saling menerjemahkan dengan yang bukan pasangannya. Hasilnya, LaBSE mampu mendukung lebih dari 109 bahasa dan mencatatkan akurasi pencarian teks sejajar sebesar 83,7% pada pengujian lintas 112 bahasa, jauh di atas model sebelumnya yang hanya mencapai 65,5% [57].

Dalam penelitian ini, LaBSE digunakan pada tahap kelima *pipeline* ETL untuk mengubah setiap potongan materi atau *chunk* menjadi vektor berdimensi 768 yang kemudian disimpan langsung sebagai properti node *Chunk* di Neo4j. Pemilihan LaBSE didasarkan pada dua pertimbangan utama. Pertama, seluruh materi kurikulum dalam sistem ini disajikan dalam Bahasa Indonesia dengan tetap mempertahankan istilah teknis pemrograman dalam Bahasa Inggris, sehingga dibutuhkan model *embedding* yang mampu menangani teks dua bahasa sekaligus dalam satu ruang vektor yang sama. LaBSE memenuhi kebutuhan ini karena sifatnya yang *language-agnostic*, artinya teks dari bahasa yang berbeda namun bermakna serupa akan menghasilkan vektor yang berdekatan tanpa memerlukan proses penerjemahan terlebih dahulu [57]. Pendekatan serupa berbasis model pre-trained BERT untuk Bahasa Indonesia telah dibuktikan mampu menghasilkan kinerja yang kompetitif pada tugas question answering dan text mining, dengan model indobert-lite-squad mencatatkan akurasi tertinggi di antara kandidat model yang diuji pada dataset Bahasa Indonesia [58]. Kedua, dimensi vektor 768 yang dihasilkan LaBSE kompatibel langsung dengan konfigurasi indeks HNSW pada

Neo4j maupun pg*vector* pada PostgreSQL yang digunakan dalam penelitian ini, sehingga tidak diperlukan penyesuaian tambahan pada lapisan penyimpanan. Vektor yang dihasilkan menjadi fondasi teknis dari kemampuan pencarian berbasis kemiripan makna dalam *pipeline* RAG, di mana setiap pertanyaan pengguna diubah menjadi vektor menggunakan model yang sama sebelum dibandingkan dengan vektor *chunk* yang tersimpan di basis data.

2.4 Tools/software yang digunakan

Sub-bab ini menjabarkan spesifikasi perangkat lunak dan lingkungan pengembangan yang dipilih untuk merealisasikan arsitektur sistem yang diusulkan. Pemilihan alat teknis ini didasarkan pada kebutuhan performa tinggi dalam pengolahan data graf berskala besar serta dukungan ekosistem yang matang untuk integrasi kecerdasan buatan. Setiap *tools*, mulai dari mesin basis data hingga *Integrated Development Environment* (IDE), ditetapkan untuk memastikan efisiensi alur kerja pengembangan (*developer experience*) sekaligus menjamin stabilitas sistem saat menangani beban komputasi dari algoritma *Retrieval-Augmented Generation* (RAG) dan *Bayesian Knowledge Tracing* (BKT) secara *real-time*.

2.4.1 Neo4j

Neo4j merupakan sistem manajemen basis data berbasis graf (Graph DBMS) yang dipilih sebagai fondasi penyimpanan utama dalam penelitian ini karena arsitekturnya yang dirancang khusus untuk menangani data dengan keterhubungan tinggi. Berbeda dengan basis data relasional (RDBMS) seperti MySQL yang mengandalkan operasi *JOIN* yang mahal secara komputasi, Neo4j menerapkan mekanisme *index-free adjacency*. Mekanisme ini memungkinkan setiap *node* memiliki pointer langsung ke *node* tetangganya, sehingga proses penelusuran (*traversal*) data kurikulum yang hirarkis dapat dilakukan secara *real-time* tanpa degradasi performa yang signifikan seiring bertambahnya volume data. Studi komparasi menunjukkan bahwa pada kueri yang melibatkan kedalaman relasi bertingkat, Neo4j menawarkan waktu eksekusi yang jauh lebih stabil dan efisien dibandingkan RDBMS yang mengalami penurunan performa drastis akibat kompleksitas tabel relasional [29], [32].

Perlu dicatat bahwa studi *benchmark* pada [32] dilakukan pada skala data sosial (*social network*) dengan jutaan node menggunakan LDBC SNB, sedangkan konteks penelitian ini beroperasi pada skala kurikulum yang jauh lebih terbatas. Meskipun demikian, temuan [29] dan [32] tetap relevan karena membuktikan bahwa keunggulan performa Neo4j bersifat scale-dependent dan semakin nyata seiring bertambahnya kedalaman relasi, sehingga relevan sebagai proyeksi skalabilitas arsitektur kurikulum yang dikembangkan [29], [32].

Selain keunggulan arsitekturalnya dibandingkan model relasional, pemilihan Neo4j dalam penelitian ini juga mempertimbangkan hasil evaluasi komparatif antar solusi basis data graf. Berdasarkan pengujian menggunakan standar *benchmark* LDBC SNB (*Linked Data Benchmark Council Social Network Benchmark*), Neo4j menunjukkan karakteristik performa yang kompetitif dibandingkan solusi graf lain seperti JanusGraph dan TigerGraph, khususnya pada operasi pemuatan data dan eksekusi kueri dengan kedalaman relasi tinggi (*K-hop queries*) [32]. Perlu dicatat bahwa hasil pengujian tersebut dilakukan pada skala data jaringan sosial dengan jutaan node, sehingga tidak dapat digeneralisasi langsung ke semua konteks. Dalam konteks penelitian ini yang beroperasi pada skala kurikulum yang jauh lebih terbatas, Neo4j dipilih karena ekosistemnya yang matang, dukungan *native vector indexing* bawaan, serta ketersediaan pustaka APOC yang memungkinkan logika pembaruan BKT berjalan langsung di level basis data tanpa memerlukan komponen eksternal tambahan.

Selain kemampuan inti tersebut, Neo4j juga dilengkapi dengan pustaka ekstensi bernama APOC (*Awesome Procedures on Cypher*), yaitu kumpulan lebih dari 450 prosedur dan fungsi tambahan yang dapat dipanggil langsung dari dalam kueri Cypher. Salah satu fitur APOC yang paling krusial dalam penelitian ini adalah mekanisme *trigger*, yaitu prosedur yang berjalan secara otomatis setiap kali terjadi perubahan data tertentu di dalam graf tanpa perlu dipanggil secara eksplisit oleh lapisan aplikasi. Dalam sistem ini, APOC trigger digunakan untuk merambatkan pembaruan nilai penguasaan materi (*mastery*) secara otomatis dari level *Chunk* ke level *Topic*, lalu naik ke Concept, dan akhirnya ke Domain, setiap kali algoritma BKT selesai memperbarui nilai pada sebuah node. Mekanisme inilah yang

memungkinkan seluruh logika urutan belajar berjalan sepenuhnya di dalam basis data tanpa keterlibatan kode di lapisan aplikasi [59].

2.4.2 Python

Python adalah bahasa pemrograman *high-level* yang bersifat *interpreted*, *dynamically typed*, dan mendukung paradigma pemrograman prosedural, berorientasi objek, maupun fungsional. Pertama kali dikembangkan oleh Guido van Rossum pada tahun 1991, Python dirancang dengan filosofi keterbacaan kode (*readability*) sebagai prioritas utama, yang tercermin dari sintaksisnya yang bersih dan ekspresif. Sebagai bahasa yang *general-purpose*, Python dapat digunakan untuk berbagai domain mulai dari pengembangan aplikasi web, otomasi sistem, analisis data, hingga pengembangan kecerdasan buatan [54].

Relevansi Python dalam penelitian ini terletak pada ekosistem library-nya yang menyeluruh untuk seluruh tahapan *pipeline* penelitian. Library sentence-transformers digunakan untuk generasi embedding LaBSE, neo4j driver untuk interaksi dengan *graph database*, beautifulsoup4 untuk web *scraping* ETL *pipeline*, dan fastapi untuk pengembangan backend API kesemuanya tersedia dalam satu ekosistem yang konsisten [54]. Python juga merupakan bahasa dominan dalam pengembangan sistem berbasis *Large Language Model* (LLM) dan *Retrieval-Augmented Generation* (RAG), sehingga kompatibilitasnya dengan Ollama API untuk inferensi Qwen2.5:7b secara lokal maupun framework evaluasi RAGAS dapat dijamin secara penuh.

2.4.3 Ollama

Ollama merupakan platform *open-source* yang dirancang untuk menjalankan dan mengelola *Large Language Model* (LLM) secara lokal pada perangkat keras konsumer tanpa ketergantungan terhadap layanan *cloud* eksternal [60]. Platform ini menyediakan *command-line interface*, REST API lokal, dan sistem manajemen model yang memungkinkan pengguna mengunduh serta menjalankan berbagai *open-weight model* seperti Qwen, Llama, dan Mistral langsung dari mesin lokal [60]. Secara teknis, Ollama menggunakan llama.cpp sebagai *inference engine* di baliknya, yaitu sebuah *library* C++ berperforma tinggi yang mendukung akselerasi GPU melalui CUDA serta optimasi CPU melalui instruksi SIMD. Model

didistribusikan dalam format GGUF (*GGML Unified Format*), sebuah format biner standar yang mengemas bobot model, tokenizer, dan metadata ke dalam satu berkas yang portabel dan mandiri [61].

Pemilihan Ollama dalam penelitian ini didasarkan pada tiga pertimbangan utama. Pertama, *reproducibility* model yang dijalankan secara lokal menghasilkan output yang konsisten tanpa dipengaruhi oleh pembaruan versi model dari pihak penyedia *cloud* [61]. Kedua, eliminasi latensi jaringan, mengingat RQ1 penelitian ini mengukur latensi kueri secara komparatif antara Neo4j dan PostgreSQL, penggunaan LLM lokal memastikan bahwa latensi yang terukur benar-benar mencerminkan performa arsitektur *graph* dan *pipeline* RAG, bukan *overhead* jaringan [60]. Ketiga, kemudahan integrasi Ollama menyediakan REST API yang kompatibel dengan standar OpenAI, sehingga integrasi dengan *pipeline* Python pada penelitian ini dapat dilakukan secara langsung tanpa konfigurasi tambahan yang kompleks.

2.4.4 Visual Studio Code

Visual Studio Code (VS Code) merupakan penyunting kode sumber (*source code editor*) lintas platform yang dikembangkan oleh Microsoft, yang menggabungkan kesederhanaan editor teks ringan dengan kemampuan fungsional *Integrated Development Environment* (IDE) yang kompleks. Berdasarkan tinjauan literatur terkini, VS Code diakui karena arsitekturnya yang sangat ekstensibel, didukung oleh ribuan ekstensi pihak ketiga yang memungkinkan kustomisasi lingkungan pengembangan sesuai kebutuhan spesifik proyek. Fitur unggulan utama yang ditawarkan meliputi *IntelliSense* untuk penyelesaian kode cerdas berdasarkan tipe variabel dan definisi fungsi, serta integrasi bawaan dengan sistem kontrol versi Git yang memfasilitasi manajemen siklus hidup perangkat lunak secara efisien [62].

VS Code dipilih sebagai lingkungan pengembangan terpadu karena kemampuannya dalam menangani ekosistem Python secara menyeluruh, mulai dari pengembangan backend FastAPI, *pipeline* RAG, hingga antarmuka prototipe berbasis React 19, dalam satu lingkungan yang menyambung. Kemampuan *debugging* bawaan pada VS Code memungkinkan inspeksi *call stack* dan eksekusi kode interaktif, yang sangat krusial untuk melacak aliran logika algoritma *Bayesian*

Knowledge Tracing (BKT) dan respons *asynchronous* dari sistem. Selain itu, efisiensi penggunaan memori VS Code menjadikannya pilihan strategis untuk menjaga performa sistem tetap optimal saat dijalankan bersamaan dengan kontainer Docker dan layanan basis data Neo4j di lingkungan pengembangan lokal.

2.4.5 Docker

Docker merupakan platform *containerization* yang memungkinkan pengemasan seluruh dependensi aplikasi beserta lingkungan eksekusinya ke dalam sebuah unit terisolasi yang disebut *container*. Berbeda dengan pendekatan instalasi konvensional yang sangat bergantung pada konfigurasi sistem operasi *host*, setiap *container* Docker berjalan secara independen dengan dependensi yang telah terdefinisi secara eksplisit, sehingga menjamin konsistensi lingkungan eksekusi di berbagai mesin yang berbeda [63], [64].

Dalam konteks penelitian ini, Docker difungsikan sebagai fondasi *reproducible research environment* yang memastikan seluruh komponen sistem dapat dijalankan secara konsisten dan terisolasi. Arsitektur sistem diimplementasikan menggunakan pendekatan *microservices* berbasis Docker, di mana setiap layanan, mulai dari *pipeline* data (materi-scrapper), backend FastAPI, antarmuka pengguna berbasis React 19, hingga basis data Neo4j dikemas dalam container tersendiri dan dikoordinasikan melalui Docker Compose. Pendekatan ini dipilih karena memungkinkan isolasi dependensi antar-*service* secara ketat, sehingga konflik versi *library* dapat dieliminasi sepenuhnya.

Selain itu, penggunaan Docker Compose sebagai alat koordinasi memungkinkan seluruh infrastruktur sistem dijalankan hanya dengan satu perintah, yang secara langsung meningkatkan aspek *reproducibility* penelitian. Hal ini menjadi pertimbangan krusial dalam konteks penelitian ilmiah, karena memungkinkan replikasi eksperimen secara identik tanpa memerlukan konfigurasi manual yang kompleks. Satu pengecualian dalam arsitektur ini adalah layanan LLM (Ollama + Qwen2.5:7b) yang dijalankan secara *native* di *host* tanpa *container*. Keputusan ini didasarkan pada keterbatasan teknis *GPU passthrough* pada Docker Desktop di lingkungan Windows 11 *native*, di mana akselerasi GPU untuk inferensi

LLM hanya dapat dilakukan secara optimal melalui Ollama yang berjalan langsung di *host*.

2.4.6 PostgreSQL dan pg*vector*

PostgreSQL adalah sistem manajemen basis data relasional (*Relational Database Management System* atau RDBMS) yang bersifat *open-source* dan mendukung penuh standar SQL. Sistem ini dikenal karena keandalannya dalam menjaga konsistensi data melalui prinsip ACID (*Atomicity, Consistency, Isolation, Durability*), kemampuannya menangani berbagai tipe data, serta ekosistem ekstensinya yang luas dan matang [12].

Dalam penelitian ini, PostgreSQL tidak digunakan sebagai sistem utama melainkan sebagai sistem pembanding struktural untuk evaluasi RQ1 dan RQ2. Sistem ini dilengkapi ekstensi pg*vector* yang menambahkan kemampuan penyimpanan dan pencarian vektor berbasis indeks HNSW langsung di dalam tabel relasional [65]. Seluruh parameter dikonfigurasi identik dengan Neo4j, yaitu dimensi vektor 768, fungsi kemiripan *cosine*, nilai m sebesar 16, dan ef_construction sebesar 64, agar perbedaan hasil yang terukur murni mencerminkan perbedaan arsitektur basis data.