# **Regulus: A Cyclical Multi-Agent Architecture for Hybrid Graph and Vector-Based Enterprise Question Answering**

**Authors:** [Author Names], HSBC Global Research & Innovation

### **Abstract**

Large, regulated enterprises face a significant challenge in answering complex, multi-faceted questions that require synthesizing information from vast, heterogeneous data stores. Existing paradigms, including monolithic Large Language Model (LLM) applications and linear Retrieval-Augmented Generation (RAG) pipelines, often lack the modularity, adaptability, and deep reasoning capabilities required for such tasks. This paper introduces Regulus, a novel, supervisor-driven multi-agent architecture designed to address these limitations. Regulus leverages the LangGraph framework to orchestrate a team of specialized agents within a cyclical, stateful workflow, enabling iterative reasoning and dynamic task delegation. A central Supervisor agent decomposes natural language queries and routes sub-tasks to appropriate worker agents, including a Structured Query Agent, an Analytics Agent, and a novel Unstructured Query Agent. This latter agent implements a HybridRAG technique, synergizing semantic vector search with knowledge graph traversal to achieve superior contextual retrieval. The key contributions of this work are threefold: (1) the formal specification of the Regulus architecture, a new blueprint for scalable and auditable enterprise Q&A; (2) a demonstration of LangGraph's power in creating complex, cyclical agentic systems that surpass the capabilities of traditional Directed Acyclic Graph (DAG) frameworks; and (3) an empirical validation of the architecture through a complex use case in global data privacy compliance. By integrating these advanced components, Regulus provides a robust solution that significantly enhances the accuracy, explainability, and depth of knowledge retrieval in complex enterprise environments.

## **1. Introduction**

### **The Enterprise Knowledge Access Problem**

Global financial institutions like HSBC operate in an environment of immense data complexity. Business-critical knowledge is distributed across a vast and heterogeneous landscape, encompassing highly structured transactional databases, client relationship management systems, and an ever-growing corpus of unstructured and semi-structured documents, such as legal opinions, regulatory filings, market analysis reports, and internal policy documents. Consequently, answering a seemingly straightforward business question—"What is our aggregate credit risk exposure to renewable energy projects in Southeast Asia, considering current geopolitical risk assessments and internal compliance policies?"—is a formidable challenge. Such queries necessitate a multi-hop reasoning process that involves accessing, interpreting, and synthesizing information from these disparate sources. Currently, this process is predominantly manual, relying on teams of human experts to navigate various systems, a workflow that is inherently slow, costly, and susceptible to inconsistency and error.

### **Limitations of Current Paradigms**

The advent of Large Language Models (LLMs) has introduced powerful new capabilities for natural language understanding and generation. However, early approaches to building LLM-powered applications have revealed significant limitations when applied to the enterprise context. Monolithic LLM applications, which attempt to encode all logic into a single, massive system, lack the modularity required for maintainability, auditing, and scalability in a corporate setting.1

Retrieval-Augmented Generation (RAG) represents a significant advancement, grounding LLM responses in factual data retrieved from external knowledge bases.3 However, the predominant implementation of RAG follows a linear, one-shot retrieval pipeline.4 In this model, a query is used to retrieve a set of documents or text chunks, which are then passed to the LLM for synthesis. This approach is fundamentally brittle; if the initial retrieval step fails to gather all the necessary or most relevant context, the quality of the final answer is irrevocably compromised. These linear pipelines struggle with compositional questions that require iterative exploration, fact-checking, and the synthesis of information gathered over multiple steps.5

### **The Agentic Shift**

To overcome these limitations, the field is undergoing a paradigm shift towards multi-agent systems (MAS).1 This approach decomposes a complex problem into a set of discrete tasks, each assigned to a specialized, autonomous agent.1 This architecture mirrors the functioning of human expert teams, where a project manager coordinates the efforts of specialists like researchers, analysts, and writers. This division of labor brings substantial benefits in terms of modularity, allowing individual agents to be developed and updated independently; scalability, as agents can operate in parallel; and fault tolerance, as the failure of a single agent does not necessarily derail the entire system.7

### **Introducing Regulus**

This paper introduces Regulus, a novel architecture designed to realize the potential of MAS for complex enterprise question-answering. The core thesis of Regulus is that a **cyclical, stateful, supervisor-driven multi-agent architecture** is essential for navigating the complexity and dynamism of enterprise knowledge. Regulus is not merely a sequence of operations but a collaborative, iterative reasoning process. It is orchestrated using the LangGraph library, which enables the creation of cyclical workflows that are fundamental for agentic behavior.8 At its heart, a central Supervisor agent receives a user's query, decomposes it into a logical plan, and dynamically delegates sub-tasks to a collective of specialized worker agents. A key innovation within this collective is an Unstructured Query Agent that employs a HybridRAG mechanism, combining the speed of semantic vector search with the contextual depth of knowledge graph traversal to retrieve highly relevant, multi-faceted information.9

### **Contributions**

This paper makes the following primary contributions to the field of large-scale data systems and applied AI:

1. **The design and formal specification of the Regulus architecture**, a novel and robust framework for building modular, scalable, and auditable question-answering systems in a complex enterprise environment.
2. **A practical demonstration of how the LangGraph framework can be utilized to implement complex, cyclical, and stateful multi-agent workflows.** This moves beyond the theoretical limitations of traditional Directed Acyclic Graph (DAG)-based agentic frameworks, enabling true iterative reasoning, re-planning, and collaboration among agents.
3. **The integration and evaluation of a HybridRAG agent within a multi-agent system.** This agent uniquely uses vector search to identify salient entry points into a knowledge graph, which is then traversed to perform high-fidelity, multi-hop retrieval, a process critical for answering complex, interconnected questions.
4. **A detailed application and evaluation of the architecture on a high-value, real-world financial services use case:** creating and querying a dynamic metamodel of global data privacy regulations to provide compliance officers with accurate and synthesized answers.

### **Paper Structure**

The remainder of this paper is structured as follows. Section 2 reviews related work in agent orchestration, multi-agent systems, and advanced RAG techniques. Section 3 provides a detailed, formal description of the Regulus architecture and its core components. Section 4 presents the data privacy compliance use case, including the construction of the regulatory knowledge graph and a walkthrough of a complex query. Section 5 outlines the methodology for evaluating the Regulus architecture against relevant baselines. Finally, Section 6 concludes with a summary of our findings and a discussion of future research directions.

## **2. Background and Related Work**

This section situates the Regulus architecture within the broader context of existing research, establishing the foundations upon which it is built and highlighting its novel contributions. Our work exists at the confluence of three rapidly evolving fields: LLM orchestration frameworks, multi-agent systems theory, and advanced information retrieval techniques. The novelty of Regulus lies not in the invention of any single component, but in their synergistic integration into a cohesive and powerful system.

### **2.1 From Chains to Graphs: The Evolution of LLM Orchestration**

The development of applications powered by LLMs has progressed from simple, single-prompt interactions to complex, multi-step workflows. This evolution has been mirrored by the frameworks designed to manage them.

#### **Linear Chains and DAGs**

Initial frameworks, most notably LangChain, popularized the concept of "chains," which link LLM calls with other components like data retrievers and tools in a sequential manner.11 This model is fundamentally based on a Directed Acyclic Graph (DAG), where data flows in a single direction from one node to the next without loops.12 DAGs are highly effective for predictable, linear workflows, such as a process that retrieves a document, summarizes it, and then analyzes its sentiment.11 The one-way flow makes these systems relatively easy to debug and reason about. However, this linearity is also a significant constraint. It does not naturally support processes that require iteration, reflection, or dynamic re-planning based on intermediate results—hallmarks of sophisticated reasoning.12

#### **The Need for Cycles**

True agentic behavior, which mimics human problem-solving, is inherently cyclical. A human expert does not simply follow a fixed plan; they reason about the problem, take an action, observe the outcome, and then *re-think* their approach based on the new information.8 To enable this in AI systems, the underlying orchestration framework must support cycles.

LangGraph was developed specifically to address this need.13 It extends the DAG paradigm by allowing developers to define graphs with loops, where the flow of control can return to a previous node.4 This is achieved through two core concepts: a persistent

**state object** that is passed between nodes and updated at each step, and **conditional edges** that can dynamically route the workflow based on the contents of that state.8 This cyclical, stateful model is the fundamental enabler for the Regulus architecture. While other agent frameworks like CrewAI (focused on role-based collaboration) and AutoGen (focused on asynchronous agent conversations) exist, LangGraph's lower-level, explicit control over state and transitions provides the granular, auditable orchestration required for enterprise-grade applications where explainability and reliability are paramount.17

This cyclical capability fundamentally changes the nature of agentic reasoning. The ReAct (Reason-Act) pattern, which structures an agent's operation into a "Thought-Action-Observation" loop, is a powerful tactic for tool use.18 In a DAG-based system, this pattern is often confined to a single, isolated step. LangGraph, however, provides the

*macro-loop* that elevates ReAct from a single-agent tactic to a multi-agent strategic capability. An agent can complete a ReAct cycle, report its findings to the central state, and allow a supervisor to decide the next step for the entire system, which might involve re-invoking the same agent with a refined task or delegating to another. This enables a more authentic and powerful form of collaborative intelligence.

### **2.2 Multi-Agent Systems (MAS) Architectures**

The concept of breaking down a complex system into a collective of collaborating agents is a well-established principle in computer science.1 The

**hierarchical** or **supervisor-worker** model is a particularly relevant architectural pattern.1 In this model, a central "master" or "supervisor" agent acts as an orchestrator, responsible for overall strategy and task decomposition.6 It delegates specific sub-tasks to a team of specialized "worker" agents, each equipped with its own distinct skills and tools.7

This architectural pattern offers several advantages analogous to those of microservices in software engineering.1 It promotes

**modularity**, as each agent is a self-contained unit that can be developed, tested, and maintained independently. It enhances **scalability**, as multiple agents can execute their tasks in parallel. Finally, it improves **resilience**, as the failure of one worker agent can be managed by the supervisor without causing a catastrophic failure of the entire system.7

The academic database community has recently begun to explore the application of MAS to complex data processing tasks. Recent work presented at top-tier conferences like SIGMOD and VLDB has demonstrated the use of multi-agent frameworks for challenges such as automated financial KPI extraction 7, large-scale SQL generation 23, and intelligent query rewriting.24 These studies validate the timeliness and academic relevance of applying MAS principles to data-intensive problems, providing a strong precedent for the architectural approach taken by Regulus.

### **2.3 Advanced Retrieval-Augmented Generation (RAG)**

RAG has become a cornerstone technique for mitigating LLM hallucinations and grounding responses in factual, external knowledge. However, the initial "naive" approach to RAG is being superseded by more sophisticated methods that better handle the complexity of real-world data.

#### **VectorRAG**

Standard RAG, which we term VectorRAG, operates by embedding a corpus of documents into a high-dimensional vector space.10 When a query is received, it is also embedded, and a vector similarity search is performed to find the most semantically similar text chunks.25 These chunks are then provided to the LLM as context. The primary strength of VectorRAG is its ability to retrieve information based on conceptual meaning, even if keywords do not match exactly.9 However, its main weakness lies in

**context fragmentation**. By breaking documents into independent chunks, it loses the explicit relationships and structural connections between pieces of information, making it difficult to answer multi-hop questions that require synthesizing facts from different parts of the knowledge base.26

#### **GraphRAG**

GraphRAG addresses this limitation by leveraging knowledge graphs as the retrieval source.10 In this paradigm, information is represented as a network of entities (nodes) and their relationships (edges). Retrieval is performed by traversing this graph structure, allowing the system to follow connections and gather a rich, interconnected subgraph of context.26 This approach excels at answering questions that depend on understanding relationships and has been shown to produce answers with significantly higher

**faithfulness** (i.e., accuracy with respect to the source material).27 It is particularly well-suited for domains like legal or medical analysis, where relationships are paramount.25

#### **HybridRAG**

HybridRAG has emerged as a state-of-the-art approach that seeks to combine the strengths of both VectorRAG and GraphRAG.9 The most effective HybridRAG systems do not treat the two methods as parallel, competing retrievers. Instead, they create a synergistic pipeline. A common and powerful pattern involves using vector search as a first-pass mechanism to efficiently identify candidate entities or concepts within the vast knowledge graph that are semantically relevant to the query.10 These identified nodes then serve as starting points for a more focused and deep graph traversal, which uncovers the precise relationships and contextual details needed to form a comprehensive answer.10 This synthesis of semantic search for discovery and graph traversal for deep contextualization is a core component of the Regulus architecture. Recent academic and industry research on "VectorGraphRAG" and other hybrid systems confirms that this is a cutting-edge area of active development.29

### **2.4 Knowledge Graphs for Regulatory Compliance**

The application of knowledge graphs to model complex legal and regulatory frameworks is a growing field of research and practice. The dense, cross-referential, and hierarchical nature of legal texts makes them an ideal candidate for representation in a graph structure. A knowledge graph can explicitly model entities such as Regulations, Articles, Obligations, and Penalties, along with the relationships that connect them, such as APPLIES\_TO, MANDATES, and CONFLICTS\_WITH.32

Several projects have demonstrated the feasibility of this approach. The "Regulatory Knowledge Graph" project, for instance, used language models to automate the construction of a compliance KG from financial regulations.34 More recently, the "PrivComp-KG" project proposed a framework for verifying privacy policy compliance against GDPR by modeling both regulations and policies within a unified knowledge graph, using RAG and Semantic Web technologies to populate and query it.35 These works provide a solid foundation for our chosen use case, showing that transforming complex regulatory documents into a queryable graph is a viable and valuable endeavor.

The convergence of these distinct research areas—cyclical orchestration, multi-agent systems, and hybrid retrieval—creates the foundation for the Regulus architecture. Regulus does not simply adopt these components in isolation; it integrates them into a cohesive system where each component enables and enhances the others. The MAS paradigm provides the high-level organizational structure; LangGraph provides the underlying "operating system" for stateful, cyclical communication; and HybridRAG serves as a highly specialized "knowledge access module" for the agents. This synthesis allows Regulus to tackle a class of complex, compositional enterprise queries that are beyond the reach of systems based on any single one of these paradigms alone.

## **3. The Regulus Architecture**

The Regulus architecture is designed from the ground up to address the requirements of complex question-answering in a large-scale, heterogeneous enterprise environment. It is founded on the principles of modularity, dynamic orchestration, and iterative reasoning. This section provides a formal description of the system's components, their interactions, and the underlying technologies that enable its functionality. The architecture is an embodiment of modern software design principles, such as microservices and composability, applied to the domain of intelligent systems.2 Each worker agent functions as a "cognitive microservice," a specialized and independent service with a clearly defined API. The Supervisor, in turn, acts as the service orchestrator, intelligently routing requests and managing the overall workflow. This design ensures that the system is not only powerful but also maintainable, scalable, and auditable—critical requirements for any system deployed within a global financial institution.

### **3.1 System Overview**

At a high level, the Regulus architecture is a supervisor-driven multi-agent system. A user's natural language query initiates a workflow that is managed by a central Supervisor Agent. This Supervisor does not answer the query directly; instead, it analyzes the query, decomposes it into a series of logical sub-tasks, and delegates these tasks to a collective of specialized Worker Agents. The entire process is orchestrated as a stateful, cyclical graph using the LangGraph framework, allowing for intermediate results to be reviewed and the plan to be dynamically adapted as new information is gathered.

Figure 1 illustrates the high-level data flow within the Regulus architecture.

Figure 1: High-Level Architecture of the Regulus System

(A diagram would be here, showing the following flow)

1. A **User** submits a **Natural Language Query** via an interface.
2. The query is received by the **Supervisor Agent**.
3. The Supervisor Agent, using its internal LLM, decomposes the query and consults the central **State Object**. Based on the plan, it uses a **Conditional Edge** to route a sub-task to one of the Worker Agents.
4. The **Worker Agent Collective** receives the sub-task. This collective includes:
   * **Structured Query Agent:** Interacts with **Enterprise SQL Databases**.
   * **Unstructured Query Agent:** Interacts with a **HybridRAG System** (comprising a Vector Store and a Knowledge Graph Database).
   * **Analytics Agent:** Utilizes tools like a **Python REPL/Calculator**.
5. The selected Worker Agent executes its task, potentially using the **ReAct pattern** to interact with its tools.
6. The Worker Agent returns its result, which updates the central **State Object**.
7. Control returns to the **Supervisor Agent**, which re-evaluates the state and decides the next step, either delegating another task (continuing the cycle) or determining the process is complete.
8. Once complete, the Supervisor synthesizes the final answer from the information in the State Object and returns it to the User.

### **3.2 The Supervisor Agent: State-Driven Dynamic Orchestration**

The Supervisor Agent is the cognitive core and central orchestrator of the Regulus system. Its primary responsibility is to manage the end-to-end workflow, transforming a high-level user request into a series of executable steps performed by the specialized worker agents.6

#### **Role and Responsibility**

The Supervisor functions as a master agent or a "manager" in a hierarchical multi-agent system.1 Upon receiving a query, it leverages a powerful LLM to perform several key functions:

* **Query Decomposition:** It breaks down a complex, multi-faceted query into a logical sequence of smaller, more manageable sub-tasks.
* **Agent Selection:** For each sub-task, it determines which worker agent possesses the necessary skills and tools to complete it.
* **Dynamic Planning and Re-planning:** It maintains and updates a plan of execution. Crucially, this plan is not static. After each worker agent completes a task and returns its findings, the Supervisor re-evaluates the overall state and can dynamically adjust the plan—adding, removing, or modifying subsequent steps as needed.
* **Result Synthesis:** Once all sub-tasks are complete, the Supervisor is responsible for synthesizing the intermediate results gathered from the workers into a final, coherent, and human-readable answer.

#### **Stateful Orchestration with LangGraph**

The dynamic and cyclical nature of the Supervisor's workflow is implemented using the LangGraph library. LangGraph's ability to create stateful graphs with loops is the key technical enabler for this orchestration model.14

* **The State Object:** The entire workflow is built around a central, persistent state object, typically defined using a Python TypedDict or Pydantic BaseModel.16 This state object acts as the system's shared memory or "workbench," persisting across all nodes in the graph.36 A typical state definition for Regulus would include:
  + original\_query: str: The initial user request.
  + plan: List[str]: A list of steps the supervisor has planned.
  + intermediate\_results: Annotated], operator.add]: A list that accumulates the outputs from each worker agent. The Annotated type with operator.add ensures that new results are appended rather than overwriting the list.4
  + conversation\_history: MessagesState: The full history of interactions, managed by LangGraph's built-in message handling.
  + final\_answer: Optional[str]: The final synthesized response.
* **Nodes and Edges:** The Supervisor and each Worker Agent are implemented as **nodes** within the StateGraph.13 A node is a function or a LangChain Runnable that receives the current state object as input and returns a dictionary of updates to that state.4 The power of the orchestration lies in the  
  **conditional edges**. After the Supervisor node runs, a conditional edge—a Python function that inspects the state—is executed. This function examines the plan and intermediate\_results to decide which node to route to next. It might return "structured\_query\_agent," "unstructured\_query\_agent," or "FINISH," thereby dynamically directing the flow of control.8
* **The Cyclical Flow:** The core workflow is a loop, as depicted in Figure 2. A normal edge connects each worker agent node back to the Supervisor node. This ensures that after a worker completes its task and updates the state, control always returns to the Supervisor. This central loop is what enables iterative refinement. The Supervisor can send a task, get a result, and then, based on that result, formulate a follow-up task for the same or a different agent. This cycle continues until the Supervisor's routing logic determines that the original query has been fully resolved and transitions to the END state.8

Figure 2: The Central Supervisor-Worker Cycle in LangGraph

(A diagram would be here, showing a simplified state machine)

1. **START** node points to the **Supervisor Node**.
2. The **Supervisor Node** has a **Conditional Edge** branching out.
3. The branches point to multiple **Worker Agent Nodes** (e.g., Worker\_A, Worker\_B). Another branch points to the **END** node.
4. Each **Worker Agent Node** has a **Normal Edge** that loops back to the **Supervisor Node**.

The explicit state management provided by LangGraph is the linchpin that enables this complex reasoning. It serves as the shared context that allows for true collaboration. One agent can perform an action, record its findings in the state, and a subsequent agent can then access and build upon those findings. Without this persistent, shared "workbench," the synthesis of information from multiple specialized agents would be computationally expensive and practically infeasible.

### **3.3 The Worker Agent Collective**

The worker agents form a team of specialists, each designed to perform a specific function with high proficiency.2 Each agent is a self-contained unit, equipped with a specific set of tools and prompted to excel in its designated role.

#### **3.3.1 The Structured Query Agent**

This agent is the system's interface to the enterprise's structured data repositories.

* **Functionality:** Its primary tool is a robust Text-to-SQL capability. When the Supervisor delegates a task like, "What was the total trade volume for HSBC in the European market for Q4 2023?", this agent is responsible for translating that natural language question into a syntactically correct and semantically accurate SQL query, executing it against the appropriate data warehouse, and returning the structured result.
* **Implementation:** This can be built using state-of-the-art LLMs fine-tuned for SQL generation, combined with tools that provide the LLM with the relevant database schema information (table names, columns, types, and relationships). Recent academic work on multi-agent systems for Text-to-SQL, which decompose the task into schema understanding, SQL generation, and validation, provides a strong blueprint for this agent's internal logic.21

#### **3.3.2 The Unstructured Query Agent & HybridRAG**

This agent is arguably the most complex and innovative worker, responsible for extracting insights from the vast corpus of unstructured documents. It implements a sophisticated HybridRAG retrieval process.

* **Hybrid Retrieval Process:** The agent's core function is a two-stage retrieval mechanism designed to maximize both relevance and contextual depth.
  1. **Stage 1: Vector Search for Entry-Point Discovery.** Given a sub-task from the Supervisor (e.g., "What are GDPR's rules on data breach notification timelines?"), the agent first creates a vector embedding of the query. It then performs a semantic similarity search against a vector index of all nodes in the knowledge graph.10 This search does not aim to find the final answer, but rather to identify the most relevant entities or concepts within the graph that should be investigated further. For the example query, this stage would return a list of candidate nodes like  
     GDPR, Data Breach, Notification Obligation, and Article 33.10
  2. **Stage 2: Graph Traversal for Contextual Retrieval.** The nodes identified in Stage 1 serve as entry points for a deep-dive graph traversal.10 The agent constructs a graph query (e.g., using Cypher for a Labeled Property Graph database) to explore the neighborhood of these entry points.10 It follows relationships such as  
     MANDATES, SPECIFIES, and APPLIES\_TO to gather a rich, interconnected subgraph of information. This process excels at answering multi-hop questions, as it reconstructs the context that is inherently present in the graph's structure but lost in a flat document index.26 For the example, it would retrieve not just the text of Article 33 but also its connection to the 72-hour timeline, the definition of a breach, and the specific penalties involved.
* **Synthesis:** The retrieved subgraph, including the text content of its nodes and relationships, is then linearized and passed to an LLM. This rich, structured context enables the LLM to generate a far more comprehensive and accurate answer to the sub-task than would be possible with fragmented text chunks from a simple vector search. This approach is validated by recent research on systems like HybridRAG and VectorGraphRAG.29

#### **3.3.3 The Analytics Agent**

This agent serves as the system's computational and logical reasoning engine.

* **Functionality:** It is equipped with tools such as a sandboxed Python REPL (Read-Eval-Print Loop) and a calculator. The Supervisor delegates tasks to this agent when a query requires numerical calculation, data aggregation, or the logical comparison and synthesis of results provided by other agents. For example, after the Structured Query Agent retrieves trade volumes and the Unstructured Query Agent retrieves market sentiment scores, the Analytics Agent could be tasked with "Calculate the correlation between trade volume and market sentiment and provide a summary."

### **3.4 The ReAct Pattern within Agents**

To ensure that the actions of each worker agent are transparent, auditable, and robust, their internal operation is structured around the **ReAct (Reason-Act)** pattern.16 ReAct forces the agent to externalize its reasoning process, creating a clear chain of thought that can be logged and reviewed—a critical feature for enterprise governance.2 A typical ReAct cycle within an agent proceeds as follows:

**Example Trace of the Unstructured Query Agent:**

* **Input (from Supervisor):** "Find the maximum financial penalty for a GDPR violation."
* **Thought 1:** The user is asking for a specific financial penalty under GDPR. I need to find the relevant article that specifies penalties. My primary tool for this is searching the regulatory knowledge graph. I will formulate a search query for the graph.
* **Action 1:** tool: graph\_search(query="GDPR maximum financial penalty")
* **Observation 1:** The tool returns a subgraph centered on the Penalty node connected to GDPR. The text from the connected Article 83 node is retrieved: "Administrative fines shall... be subject to fines up to 20,000,000 EUR, or in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year, whichever is higher."
* **Thought 2:** The observation contains the complete answer. It specifies two conditions (€20 million and 4% of global turnover) and the rule for selecting the higher value. I have sufficient information to formulate the final answer for my sub-task.
* **Final Answer (to Supervisor):** The maximum financial penalty for a GDPR violation is the greater of €20 million or 4% of the company's total worldwide annual turnover from the preceding financial year.

This explicit, step-by-step process of reasoning, acting, and observing allows the agent to handle complex tasks methodically and provides a clear audit trail of its decision-making process.

## **4. Use Case: A Dynamic Metamodel for Global Data Privacy Compliance**

To demonstrate the practical utility and power of the Regulus architecture, we apply it to a complex, high-value problem domain within HSBC: navigating the intricate landscape of global data privacy regulations. A global financial institution must adhere to a complex and evolving patchwork of data protection laws across numerous jurisdictions, such as the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA) in the United States.42 Understanding the specific, often overlapping or conflicting, obligations imposed by these regulations is a significant operational challenge for compliance, legal, and risk departments.34

### **4.1 Constructing the Regulatory Knowledge Graph**

The foundation of our use case is a knowledge graph that serves as a dynamic, queryable metamodel of regulatory information. This graph is the primary knowledge source for the Unstructured Query Agent.

#### **Modeling Approach**

We chose to model the regulatory landscape using the **Labeled Property Graph (LPG)** model.32 The LPG model, with its intuitive structure of nodes, relationships, and properties, is well-suited for representing the entities and connections within legal texts.33 While the Resource Description Framework (RDF) offers more formal semantics and is aligned with W3C standards, the LPG model provides greater schema flexibility and is supported by query languages like Cypher, which are highly optimized for the type of traversal and pattern-matching queries required for our application-specific analysis.32 The process of creating this graph follows established best practices for applied ontology development, including engaging with legal and compliance stakeholders to define the core concepts and iteratively refining the model based on real-world regulatory texts.46

#### **Ontology and Schema**

The schema for our regulatory knowledge graph is designed to capture the key entities and relationships within data privacy laws. It provides a structured representation of the unstructured text, enabling precise, multi-hop queries. The core components of the schema are formally defined in Table 1. This explicit schema is essential for the reproducibility of our work and serves as a blueprint for modeling similar complex domains. The creation of such a graph, transforming dense legal text into a structured format, is a critical step towards "compliance as code".34

**Table 1: Labeled Property Graph Schema for Regulatory Compliance**

| Component Type | Label/Type | Properties (with Data Types) | Description |
| --- | --- | --- | --- |
| **Node** | Regulation | name: string, version: string, effective\_date: date, territorial\_scope: string | Represents a specific law or regulation (e.g., "GDPR", "CCPA"). |
| **Node** | Article | article\_id: string, title: string, full\_text: string | A specific article or section within a regulation. |
| **Node** | Obligation | obligation\_type: string (e.g., 'Notification', 'Consent'), description: string | A specific duty or requirement mandated by an article. |
| **Node** | DataSubject | type: string (e.g., 'EU Resident', 'California Consumer') | The category of individual protected by a regulation. |
| **Node** | Penalty | max\_amount\_eur: float, max\_percent\_turnover: float, description: string | The financial or other penalties for non-compliance. |
| **Node** | ThirdParty | role: string (e.g., 'Data Processor', 'Data Controller') | An entity involved in data processing. |
| **Relationship** | CONTAINS | {} | Connects a Regulation to its Articles. |
| **Relationship** | MANDATES | {} | Connects an Article to an Obligation. |
| **Relationship** | APPLIES\_TO | {} | Connects a Regulation to a DataSubject type. |
| **Relationship** | SPECIFIES | {} | Connects an Obligation to a Penalty. |
| **Relationship** | DEFINES\_ROLE | {} | Connects a Regulation to a ThirdParty role. |

### **4.2 Answering a Multi-Hop Compliance Query**

The true power of the Regulus architecture and the underlying knowledge graph is revealed when tackling complex, multi-hop queries that a human compliance officer might pose. These queries cannot be answered by a simple keyword search or by retrieving a single document chunk.

#### **Example Query**

Consider the following realistic scenario and query:

"A data breach at an HSBC third-party data processor located in India has exposed the personal data of clients in both California and Germany. Compare the breach notification obligations, including timelines and responsible parties, under CCPA and GDPR."

This query is complex because it requires:

* Identifying two separate regulatory regimes (GDPR for Germany, CCPA for California).
* Understanding the specific rules within each regime for data breaches.
* Filtering those rules for the context of a *third-party data processor*.
* Extracting specific details like timelines and responsible parties.
* Synthesizing the findings into a comparative analysis.

#### **Execution Trace**

The following trace details how the Regulus system would process this query, showcasing the collaboration between the Supervisor and the worker agents.

1. **User -> Supervisor:** The query is submitted to the system.
2. **Supervisor -> State:** The Supervisor's LLM analyzes the query and decomposes it into a plan, updating the state object: plan =.
3. **Supervisor -> Unstructured Query Agent:** The Supervisor reads the first task from the plan and delegates it to the Unstructured Query Agent.
4. **Unstructured Query Agent (GDPR Task):**
   * **Vector Search:** The agent embeds the task and finds relevant entry-point nodes in the knowledge graph, such as GDPR, Data Breach, and ThirdParty (with role: 'Data Processor').
   * **Graph Traversal:** Using these entry points, the agent executes a Cypher query to traverse the graph. It follows paths from the GDPR node through CONTAINS to Article nodes (specifically finding Article 33), then through MANDATES to Obligation nodes (of obligation\_type: 'Notification'). The query also confirms the relationship DEFINES\_ROLE for 'Data Processor'. This traversal reveals that under GDPR, a data processor must notify the data controller "without undue delay" after becoming aware of a breach. The controller then has a 72-hour timeline to notify the competent supervisory authority.48
   * **Response to Supervisor:** The agent returns a structured summary of the GDPR obligations.
5. **Supervisor -> State:** The Supervisor receives the GDPR findings and appends them to the intermediate\_results list in the state object. It marks the first task as complete.
6. **Supervisor -> Unstructured Query Agent:** The Supervisor reads the next task from the plan ("Find CCPA breach notification rules") and delegates it again to the Unstructured Query Agent.
7. **Unstructured Query Agent (CCPA Task):**
   * The agent repeats its process for the CCPA. Vector search identifies the CCPA node. Graph traversal finds the relevant obligations. It discovers that California's breach notification statute (Cal. Civ. Code § 1798.82) requires notification "in the most expedient time possible and without unreasonable delay".52 It also finds the requirement to notify the California Attorney General if more than 500 California residents are affected.54
   * **Response to Supervisor:** The agent returns a structured summary of the CCPA obligations.
8. **Supervisor -> State:** The state is updated with the CCPA findings. The second task is marked as complete.
9. **Supervisor -> Analytics Agent:** The Supervisor now reads the final task ("Compare GDPR and CCPA findings"). It delegates this synthesis task to the Analytics Agent, providing the accumulated GDPR and CCPA results from the intermediate\_results field in the state object as context.
10. **Analytics Agent:** The agent's LLM receives the structured findings for both regulations. It generates a comparative analysis, highlighting key differences: the processor-to-controller notification step in GDPR, the specific 72-hour timeline for controllers in GDPR versus the more ambiguous "without unreasonable delay" in CCPA, and the different thresholds for notifying regulatory bodies.
11. **Supervisor -> State:** The Supervisor receives the final comparative analysis and places it in the final\_answer field of the state. It sees the plan is now empty.
12. **Supervisor -> END:** The Supervisor transitions to the END state, and the final, synthesized answer is formatted and returned to the user.

This detailed trace illustrates how the cyclical, stateful, and collaborative nature of the Regulus architecture enables it to deconstruct and solve a complex problem that would be intractable for a linear RAG system.

## **5. Evaluation**

To empirically validate the effectiveness of the Regulus architecture, a rigorous evaluation is necessary. This section outlines a proposed methodology for assessing the system's performance against appropriate baselines, using a comprehensive set of metrics designed to measure the quality and reliability of generated answers. The goal is to demonstrate quantitatively that the novel architectural choices made in Regulus—namely, the cyclical multi-agent orchestration and the HybridRAG retrieval mechanism—lead to superior performance on complex question-answering tasks.

### **5.1 Methodology**

The evaluation will be conducted using a custom-built benchmark dataset and a comparative analysis against two baseline systems.

* **Benchmark Dataset:** A benchmark dataset will be created consisting of 100 complex, multi-hop questions related to the domain of global data privacy regulations. These questions will be designed to mirror the complexity of real-world queries from compliance officers. For each question, a "gold standard" ground-truth answer will be meticulously curated by a panel of legal and compliance experts. This ground-truth answer will serve as the reference for evaluating the generated responses.
* **Systems for Comparison:** The performance of the full **Regulus** architecture will be compared against two simplified baseline systems:
  1. **Baseline A (VectorRAG):** A single-agent system that uses a standard VectorRAG approach. This agent will retrieve information by performing a semantic similarity search over a flat index of the raw regulatory text documents. It represents the current state-of-the-art for simple RAG implementations.
  2. **Baseline B (GraphRAG):** A single-agent system that uses a pure GraphRAG approach. This agent will interact solely with the regulatory knowledge graph, using query-to-Cypher translation to retrieve context. It will not have the benefit of the initial vector search for entry-point discovery.

### **5.2 Metrics**

To provide a multi-faceted assessment of performance, we will employ a suite of evaluation metrics inspired by recent RAG evaluation frameworks.31 The answers generated by each system will be evaluated against the ground-truth answers using the following criteria:

* **Faithfulness:** This metric measures the factual accuracy of the generated answer with respect to the retrieved context. It is designed to quantify the degree of hallucination. An LLM-as-a-judge approach will be used to break down the generated answer into a set of individual statements and verify whether each statement is directly supported by the provided source context.31 A higher score indicates a more factually grounded answer.
* **Answer Relevance:** This metric assesses how well the generated answer addresses the user's original question. It is possible for an answer to be faithful to its context but fail to actually answer the query. This will be measured by using an LLM to score the relevance of the answer to the question on a numerical scale.31
* **Context Precision:** This metric evaluates the signal-to-noise ratio of the retrieved context. It measures what proportion of the retrieved context was actually useful and necessary for constructing the ground-truth answer.31 High precision indicates an efficient retrieval process that does not overwhelm the generator with irrelevant information.
* **Context Recall:** This metric evaluates the completeness of the retrieved context. It measures whether the retrieval step successfully found all the necessary pieces of information from the knowledge base that are required to formulate the ground-truth answer.31 Low recall is a primary failure mode for RAG systems, as information not retrieved cannot be included in the final answer.

### **5.3 Expected Results and Discussion**

We hypothesize that the Regulus architecture will demonstrate superior performance across these metrics, particularly on the complex, multi-hop questions that constitute our benchmark.

* **Hypothesis 1: Regulus will exhibit significantly higher Faithfulness and Context Recall compared to Baseline A (VectorRAG).** The standard VectorRAG approach is prone to missing context because relevant information may be scattered across multiple document chunks that are not all retrieved by a single similarity search. The HybridRAG agent in Regulus, by traversing the knowledge graph, can gather a more complete and interconnected set of facts, leading to higher recall. This richer context, in turn, reduces the likelihood of the LLM hallucinating or making incorrect inferences, thus improving faithfulness.27
* **Hypothesis 2: Regulus will achieve higher Answer Relevance than Baseline B (GraphRAG).** A pure GraphRAG system can struggle when a query is abstract or does not contain specific entities that map directly to nodes in the graph. The HybridRAG agent's initial vector search acts as a powerful semantic grounding step, identifying the most relevant starting points for graph traversal even for abstract queries. This ensures that the subsequent retrieval is focused on the correct area of the knowledge graph, leading to a more relevant final answer.31
* **Hypothesis 3: The Regulus architecture will be capable of answering a class of complex, compositional queries that are intractable for the single-agent baselines.** For questions that require a clear decomposition of steps (e.g., "First find X, then find Y, then compare them"), the single-agent baselines are likely to fail, either by only addressing the first part of the query or by producing a confused, incomplete response. The Supervisor agent in Regulus is explicitly designed to handle this compositionality. By breaking the problem down and sequentially invoking different agents while maintaining state, Regulus can systematically construct an answer that the baselines cannot. We will present qualitative examples of these successes and failures to support this hypothesis.

The results of this evaluation are expected to provide strong empirical evidence for the advantages of a cyclical, multi-agent, hybrid-retrieval architecture for tackling complex enterprise question-answering.

## **6. Conclusion and Future Work**

### **Recapitulation**

In this paper, we have introduced Regulus, a novel architecture for enterprise question-answering designed to overcome the limitations of linear, monolithic AI systems. Regulus is a supervisor-driven multi-agent system that leverages the LangGraph framework to create cyclical, stateful workflows, enabling complex, iterative reasoning. Its core components—a dynamic Supervisor agent that orchestrates a team of specialized worker agents, including a sophisticated HybridRAG agent that synergizes vector and graph retrieval—work in concert to deconstruct and solve multi-faceted queries.

We have provided a formal specification of the architecture, grounding its design in established principles of multi-agent systems and modern software engineering. We have detailed how the specific capabilities of LangGraph, particularly its support for cycles and explicit state management, are not merely technical features but are the fundamental enablers of true agentic collaboration and re-planning. Through a detailed use case in the complex domain of global data privacy compliance, we have demonstrated how Regulus can navigate a knowledge graph to answer multi-hop questions that would be intractable for simpler RAG systems. Finally, we have outlined a robust evaluation methodology designed to empirically validate the superior performance of our proposed architecture.

### **Implications for Enterprise AI**

The Regulus architecture represents a significant step forward in the development of practical, enterprise-grade AI systems. Its implications extend beyond the specific use case presented.

* **A Blueprint for Collaborative AI:** Regulus provides a tangible blueprint for building systems where multiple specialized AI components collaborate to solve problems, much like human expert teams. This modular "cognitive microservices" approach is essential for building systems that are scalable, maintainable, and adaptable to evolving business needs.1
* **Enhanced Auditability and Trust:** By forcing agents to follow the explicit "Thought-Action-Observation" ReAct pattern within a stateful, logged workflow, the system's reasoning process becomes transparent and auditable.2 This is a non-negotiable requirement for deploying AI in high-stakes, regulated environments like finance, where "black box" solutions are unacceptable.
* **Unlocking Complex Knowledge Domains:** The architecture is domain-agnostic. The same pattern used to model data privacy regulations can be applied to other complex knowledge domains prevalent in finance, such as anti-money laundering (AML) protocols, risk management frameworks, trade finance rules, and investment research analysis. Regulus provides a general-purpose framework for turning complex enterprise knowledge into a queryable, intelligent asset.

### **Future Work**

This work opens several promising avenues for future research and development. We identify four key directions:

1. **Formal Human-in-the-Loop Integration:** While the current architecture allows for implicit human oversight by reviewing logs, future versions will focus on integrating formal **human-in-the-loop (HITL)** checkpoints directly into the LangGraph workflow.2 For high-stakes decisions, the Supervisor could be configured to pause the workflow and route a proposed action or a synthesized answer to a human expert for approval before proceeding. This would combine the speed and scale of AI with the judgment and accountability of human experts, a critical feature for production deployment.56
2. **Dynamic Agent Spawning and Management:** The current implementation uses a fixed collective of worker agents. A more advanced Supervisor could be developed with the capability to **dynamically instantiate** or "spawn" new, temporary agents tailored to novel sub-tasks it encounters. This would move the system from a fixed team structure to a more fluid, on-demand "gig economy" of agents, further enhancing its flexibility and problem-solving power.
3. **Automated Knowledge Graph Maintenance:** The regulatory knowledge graph is the foundation of the system's intelligence, but regulations evolve. A key area of future work is to create a feedback loop where the agentic system itself is used to **maintain its own knowledge base**. An agent could be tasked to periodically scan regulatory sources for updates, use its understanding to identify changes, and automatically propose updates to the knowledge graph, thus creating a self-maintaining, "living" knowledge system.58
4. **Cross-Domain Reasoning:** The current system excels within a single, complex domain. The next frontier is to expand the agent collective to include specialists for entirely different domains (e.g., an agent for internal HR policies, an agent for real-time market data). The challenge and opportunity will be to enhance the Supervisor's reasoning capabilities to orchestrate complex queries that require synthesizing information across these disparate domains, enabling true holistic business intelligence.

#### Works cited

1. Understanding Agents and Multi Agent Systems for Better AI Solutions - HatchWorks, accessed July 4, 2025, <https://hatchworks.com/blog/ai-agents/multi-agent-systems/>
2. AI agent architecture and design | Deloitte US, accessed July 4, 2025, <https://www.deloitte.com/us/en/services/consulting/articles/ai-agent-architecture-and-multiagent-systems.html>
3. A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2408.05141v1>
4. LangGraph - LangChain Blog, accessed July 4, 2025, <https://blog.langchain.dev/langgraph/>
5. Pseudo-Knowledge Graph: Meta-Path Guided Retrieval and In-Graph Text for RAG-Equipped LLM - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2503.00309v1>
6. Multi-Agent Architecture Explained: What It Is and Why It Works - Lyzr AI, accessed July 4, 2025, <https://www.lyzr.ai/blog/multi-agent-architecture/>
7. Supervisor-Driven Multi-Agent Systems: A Blueprint for Scalable AI ..., accessed July 4, 2025, <https://medium.com/@mohitbasantani1987/supervisor-driven-multi-agent-systems-a-blueprint-for-scalable-ai-workflows-96b8b78e440f>
8. A Beginner's Guide to LangGraph in Agentic AI | by Shikha Singh ..., accessed July 4, 2025, <https://medium.com/@shiqs90/a-beginners-guide-to-langgraph-in-agentic-ai-33a60807cc3b>
9. HybridRAG. Combining the strengths of VectorRAG… | by Bijit Ghosh | Medium, accessed July 4, 2025, <https://medium.com/@bijit211987/hybridrag-0a48228dd97c>
10. Exploring RAG and GraphRAG: Understanding when and how to ..., accessed July 4, 2025, <https://weaviate.io/blog/graph-rag>
11. LangChain vs LangGraph: Which Framework Wins? - ProjectPro, accessed July 4, 2025, <https://www.projectpro.io/article/langchain-vs-langgraph/1123>
12. LangChain vs. LangGraph: Choosing the Right Framework | by Tahir | Medium, accessed July 4, 2025, <https://medium.com/@tahirbalarabe2/langchain-vs-langgraph-choosing-the-right-framework-0e393513da3d>
13. LangGraph - cycling through multi-agent LLM applications, accessed July 4, 2025, <https://blog.scottlogic.com/2024/05/13/langgraph-cycling-through-llm-applications.html>
14. LangGraph: A Framework for Building Stateful Multi-Agent LLM Applications | by Ken Lin, accessed July 4, 2025, <https://medium.com/@ken_lin/langgraph-a-framework-for-building-stateful-multi-agent-llm-applications-a51d5eb68d03>
15. langgraph - PyPI, accessed July 4, 2025, <https://pypi.org/project/langgraph/0.0.25/>
16. ReAct agent from scratch with Gemini 2.5 and LangGraph, accessed July 4, 2025, <https://ai.google.dev/gemini-api/docs/langgraph-example>
17. Comparing Open-Source AI Agent Frameworks - Langfuse Blog, accessed July 4, 2025, <https://langfuse.com/blog/2025-03-19-ai-agent-comparison>
18. Implementing ReAct Agentic Pattern From Scratch - Daily Dose of Data Science, accessed July 4, 2025, <https://www.dailydoseofds.com/ai-agents-crash-course-part-10-with-implementation/>
19. Hierarchical multi-agent systems with LangGraph - YouTube, accessed July 4, 2025, <https://www.youtube.com/watch?v=B_0TNuYi56w>
20. Multi-Agent System Tutorial with LangGraph - FutureSmart AI Blog, accessed July 4, 2025, <https://blog.futuresmart.ai/multi-agent-system-with-langgraph>
21. Structuring the Unstructured: A Multi-Agent System for Extracting and Querying Financial KPIs and Guidance - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2505.19197v1>
22. Structuring the Unstructured: A Multi-Agent System for ... - arXiv, accessed July 4, 2025, <https://arxiv.org/abs/2505.19197>
23. SQL-Factory: A Multi-Agent Framework for High-Quality and Large-Scale SQL Generation - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2504.14837v1>
24. QUITE: A Query Rewrite System Beyond Rules with LLM Agents - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2506.07675v1>
25. Which RAG is More Superior: Graph RAG or Vector RAG? | by Jillani SofTech - Medium, accessed July 4, 2025, <https://jillanisofttech.medium.com/which-rag-is-more-superior-graph-rag-or-vector-rag-55974f50b9c9>
26. Graph RAG vs Vector RAG: Solving Gartner's Challenges - FalkorDB, accessed July 4, 2025, <https://www.falkordb.com/blog/graph-rag-vs-vector-rag-solving-gartner-challenges/>
27. GraphRAG Analysis, Part 2: Graph Creation and Retrieval vs Vector Database Retrieval - Blog | MLOps Community, accessed July 4, 2025, <https://home.mlops.community/public/blogs/graphrag-analysis-part-2-graph-creation-and-retrieval-vs-vector-database-retrieval>
28. Which is better: HybridRAG, VectorRAG, or GraphRAG? : r/Rag - Reddit, accessed July 4, 2025, <https://www.reddit.com/r/Rag/comments/1eyrdy4/which_is_better_hybridrag_vectorrag_or_graphrag/>
29. TigerVector: Supporting Vector Search in Graph Databases for Advanced RAGs - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2501.11216v3>
30. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction - arXiv, accessed July 4, 2025, <https://arxiv.org/html/2408.04948v1>
31. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval ..., accessed July 4, 2025, <https://arxiv.org/abs/2408.04948>
32. Knowledge Graphs | Tom Sawyer Software, accessed July 4, 2025, <https://www.tomsawyer.com/knowledge-graphs>
33. Graph data model - Memgraph, accessed July 4, 2025, <https://memgraph.com/docs/data-modeling/graph-data-model>
34. arxiv.org, accessed July 4, 2025, <https://arxiv.org/abs/2302.01842>
35. PrivComp-KG: Leveraging Knowledge Graph and Large Language ..., accessed July 4, 2025, <https://arxiv.org/abs/2404.19744>
36. Exploring Agentic Workflows with Langgraph and Neo4j | by Raajas Sode - Medium, accessed July 4, 2025, <https://medium.com/@RaajasSode/exploring-agentic-workflows-with-langgraph-and-neo4j-c4cd031814e9>
37. Understanding State in LangGraph: A Beginners Guide | by Rick Garcia | Medium, accessed July 4, 2025, <https://medium.com/@gitmaxd/understanding-state-in-langgraph-a-comprehensive-guide-191462220997>
38. LangGraph: Orchestrating Agent Workflows with State Machines - Articles, accessed July 4, 2025, <https://sidharthr.pythonanywhere.com/article/langgraph-article>
39. LangGraph: Build Stateful AI Agents in Python, accessed July 4, 2025, <https://realpython.com/langgraph-python/>
40. Complete Guide to Building LangChain Agents with the LangGraph Framework - Zep, accessed July 4, 2025, <https://www.getzep.com/ai-agents/langchain-agents-langgraph>
41. Comprehensive Guide to ReAct Prompting and ReAct based Agentic Systems - Mercity AI, accessed July 4, 2025, <https://www.mercity.ai/blog-post/react-prompting-and-react-based-agentic-systems>
42. CCPA vs GDPR. What's the Difference? [With Infographic] - CookieYes, accessed July 4, 2025, <https://www.cookieyes.com/blog/ccpa-vs-gdpr/>
43. What is a labeled property graph? - Oxford Semantic Technologies, accessed July 4, 2025, <https://www.oxfordsemantic.tech/faqs/what-is-a-labeled-property-graph>
44. RDF vs LPG for GraphRAG : r/KnowledgeGraph - Reddit, accessed July 4, 2025, <https://www.reddit.com/r/KnowledgeGraph/comments/1i297lb/rdf_vs_lpg_for_graphrag/>
45. Graphs to Graph Neural Networks: From Fundamentals to Applications — Part 2c: RDF vs. LPG Knowledge Graphs - Isaac Kargar, accessed July 4, 2025, <https://kargarisaac.medium.com/graphs-to-graph-neural-networks-from-fundamentals-to-applications-part-2c-rdf-vs-43f246764e39>
46. 6 Best Practices for Applied Ontology Development - KaDSci, accessed July 4, 2025, <https://kadsci.com/applied-ontology-development/>
47. 11 best practices for data privacy and compliance | Strengthen security with structured data privacy governance | Improve compliance with a data privacy framework | Reduce risk exposure with data privacy best practices | Lumenalta, accessed July 4, 2025, <https://lumenalta.com/insights/11-best-practices-for-data-privacy-and-compliance>
48. Understanding the GDPR breach notification timeline: A step-by-step guide - Thoropass, accessed July 4, 2025, <https://thoropass.com/blog/compliance/gdpr-breach-notification-timeline/>
49. What is a data breach and what do we have to do in case of a data breach? - European Commission, accessed July 4, 2025, <https://commission.europa.eu/law/law-topic/data-protection/rules-business-and-organisations/obligations/what-data-breach-and-what-do-we-have-do-case-data-breach_en>
50. Art. 33 GDPR – Notification of a personal data breach to the supervisory authority, accessed July 4, 2025, <https://gdpr-info.eu/art-33-gdpr/>
51. What are GDPR's rules on data breaches? - TrueVault, accessed July 4, 2025, <https://www.truevault.com/learn/what-are-gdprs-rules-on-data-breaches>
52. Don't Delay! California Likely to Soon Require Data Breach Notifications to be Provided to Consumers Within 30 Days, accessed July 4, 2025, <https://www.fisherphillips.com/en/news-insights/california-likely-to-soon-require-data-breach-notifications-to-be-provided-to-consumers-within-30-days.html>
53. Understanding Breach Notification Obligations Under California Law: What Does the CCPA Require? | Consumer Financial Services Law Monitor, accessed July 4, 2025, <https://www.consumerfinancialserviceslawmonitor.com/2024/07/understanding-breach-notification-obligations-under-california-law-what-does-the-ccpa-require/>
54. Data Security Breach Reporting | State of California - Department of Justice - CA.gov, accessed July 4, 2025, <https://oag.ca.gov/privacy/databreach/reporting>
55. CCPA & Data Breach Notification Law FAQ's - foreUP, accessed July 4, 2025, <https://www.foreupgolf.com/ccpa-2/>
56. LangGraph - LangChain, accessed July 4, 2025, <https://www.langchain.com/langgraph>
57. LangGraph, accessed July 4, 2025, <https://langchain-ai.github.io/langgraph/>
58. Construction of Knowledge Graphs: State and Challenges - arXiv, accessed July 4, 2025, <https://arxiv.org/abs/2302.11509>