# A Definitive Guide to Generative AI with Amazon Bedrock

# Chapter 7: Overview of Amazon Bedrock Knowledge Bases

In the last chapter, you learned about RAG design patterns and their importance in generative AI. Now, you might be thinking about how to create and use these patterns with a native integration with Amazon Bedrock. Enhancing large language models with external knowledge is vital for developing precise AI applications. Amazon Bedrock Knowledge Bases simplifies retrieval-augmented generation (RAG). It allows models to find relevant information before creating answers.

This chapter explores how Amazon Bedrock Knowledge Bases simplifies the integration of private data sources into generative AI workflows. It covers the end-to-end process of data ingestion, chunking strategies for efficient retrieval, semantic search capabilities, and augmenting model prompts with retrieved information.

The chapter discusses important features like multi-turn conversation support and customized retrieval. It also covers source attribution and cost-effectiveness. Key aspects include data security, access control, monitoring, and compliance. Amazon knowledge bases can securely connect to data sources like Amazon S3, Salesforce, and SharePoint.

Additionally, it provides insights into selecting the right vector database for storing and searching high-dimensional embeddings, a core component of RAG systems. Even so, you will learn the different chunking strategy. By the end of this chapter, readers will understand how the Amazon Bedrock Knowledge Bases accelerates the development of generative AI applications enriched with proprietary data.

7.1 Introduction to Amazon Bedrock Knowledge Bases

In the last chapter, you learned about retrieval augmented generation (RAG). RAG is a design pattern to improve the output of a large language model by referencing an outside knowledge bases before generating a response. RAG makes sure that the model's answers are more accurate and based on more recent, reliable data than just the data it used for training. You will learn how AWS demystifies the implementation of RAG. Even smooth integration with other features of Amazon Bedrock.

Amazon Bedrock Knowledge Bases allow you to fully leverage RAG by providing a simple way to access external data and enrich Large Language Model (LLM) outputs. Application can query this resource connecting your data sources to knowledge bases. Then, it gets the right information to add to the context and answer your question through direct quotes or natural language. This approach allows you to build applications enriched by the context provided through the knowledge bases, speeding up your time to market. By eliminating the need to manually build data pipelines, Amazon Bedrock delivers an out-of-the-box RAG solution, making application development faster and more efficient. Additionally, integrating a knowledge bases reduces costs, as there’s no need for continual retraining of the model to incorporate your private data.

**Data preparation and organization**

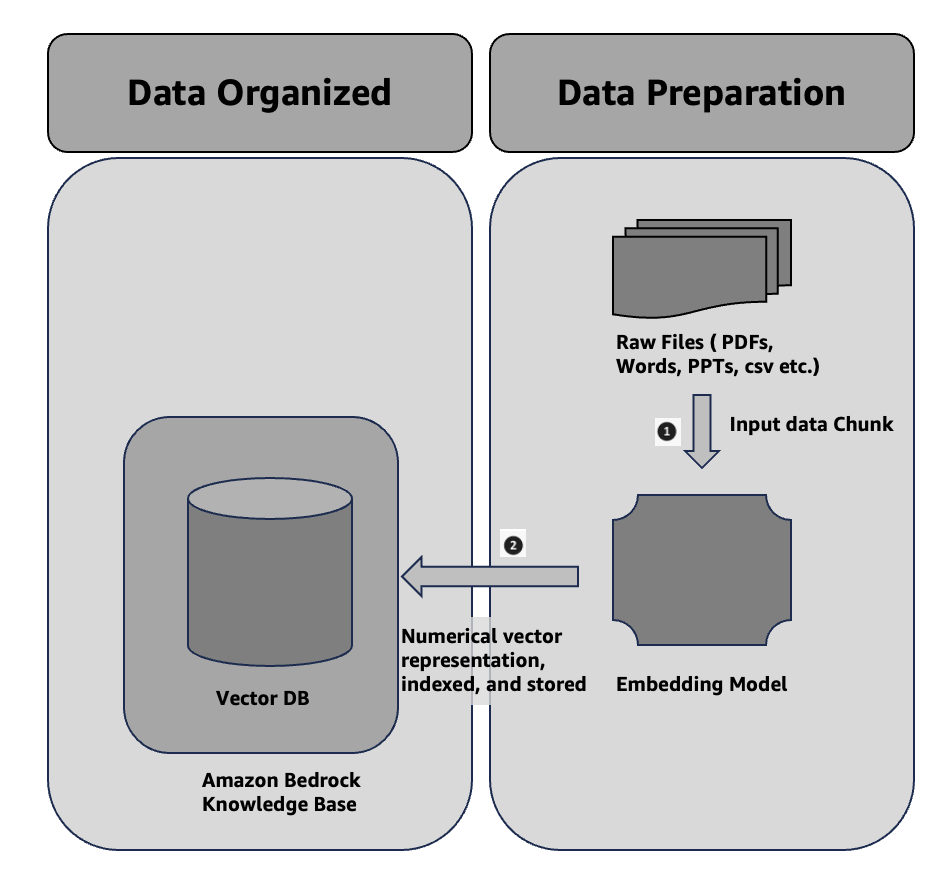


Figure 7.1 Data Preparation & Organization

Even so This is the same as bullet points 1 and 2 in Section 6.4 of the last chapter. To make it easier to get private data, one common design is to first split the documents into chunks for efficient retrieval. Next, you embed the chunks into embeddings and transfer them to a vector index, which maintains their mapping to the original text. You use these embeddings to determine the semantic relationships between queries and text from the data sources. Figure 7.1 shows how data is prepared for the vector database. Instead of the Vector DB you used and managed on your own, Amazon Bedrock Knowledge Bases now offers a variety of options for Vector DB.

**Experiences and Information Retrieval and Generation**

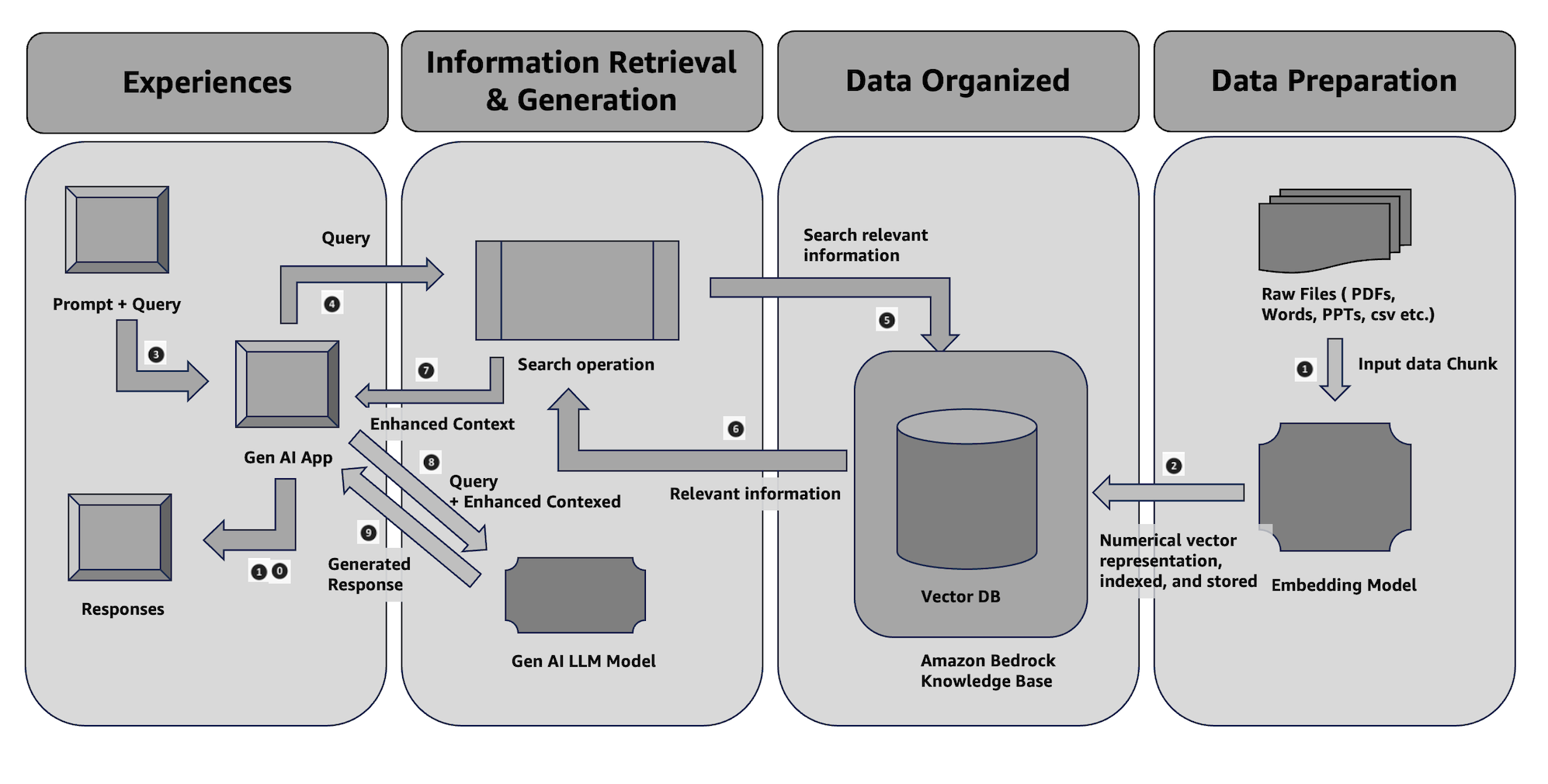
****

Figure 7.2 Experiences and Information Retrieval & Generation

Even so This is the same as bullet points 5–10 in Section 6.4 of the last chapter. You use an embedding model to transform your query (prompt) into a vector. You then query the vector index, comparing document vectors to your query vector, to identify chunks that share semantic similarities with your query. The last step improves your prompt by incorporating additional information from the chunks retrieved from the vector index. You then pass the prompt and additional information to the model, allowing it to generate a response got you. Figure 7.2 below illustrates how RAG works in runtime to improve responses to your questions. Additionally, in the sections below, you will learn some specific APIs to communicate with the Amazon Bedrock Knowledge Bases.

7.2 Why Amazon Bedrock Knowledge Bases

In this section, you will discover the significance of Amazon Bedrock Knowledge Bases in the development of your generative AI application.

**Seamless RAG Workflow**: Fully managed support for retrieval augmented generation (RAG), eliminating the need for custom integrations and manual data handling.

**Contextual AI with Proprietary Data**: Enables foundation models (FMs) and agents to access your company’s private data, delivering more relevant, accurate, and customized responses. You will learn about the agent in the next chapter.

**Secure Data Connectivity**: This feature securely connects to data sources such as Amazon S3, Salesforce, Confluence, and SharePoint, automatically ingesting and indexing content. Amazon Knowledge Bases natively support some sources to connect. (Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/data-source-connectors.html> )

**Flexible Data Ingestion**: It supports several different ingestion methods, such as handling complex unstructured data (PDFs, images), and you can change the chunking settings to make it easier to find information.

**Supports multi-turn conversations**: Built-in session context management allows your app to handle ongoing conversations, maintaining coherence across interactions.

**Customized Retrieval**: You can improve the accuracy of retrieval by enhancing queries and using advanced processing to make them work best for your business.

**No External Vector Database is Needed**: This solution offers a managed vector store, like Amazon OpenSearch Serverless and others, or the flexibility to connect to your existing vector databases, such as Pinecone or Redis.

**Augmented prompts**: These automatically enrich your queries with relevant, up-to-date information to improve response quality.

**Source Attribution**: It provides citations for retrieved data, ensuring transparency and minimizing AI hallucinations.

**Cost-effective**: By dynamically augmenting models with real-time, proprietary data, it reduces the need for constant retraining of models.

**Quick Time-to-Market**: This solution simplifies the process of constructing pipelines, providing a ready-to-use RAG solution that expedites the development of AI applications.

7.3 Sample Applications of Amazon Bedrock Knowledge Bases

To get the GitLab details, refer to the **appendix** section of this book. In GitLab, locate the repository named **genai-bedrock-book-samples** and click on it.

Inside the **genai-bedrock-book-samples** repository is an AWS CloudFormation template that resides in the **cloudformation** folder. If you already executed the AWS CloudFormation template in Chapter 3 and didn't delete the stack afterward, you can skip the paragraph highlighted in grey below.

The task requires the execution of an AWS CloudFormation template, which should be performed once for all exercises in this book. A detailed guidance on how to manually execute the AWS CloudFormation template can be found in a file called **README** located within a directory named **cloudformation**. For more information about AWS CloudFormation template refer <https://aws.amazon.com/cloudformation/>.

***Disclaimer****: It is advisable to delete the AWS CloudFormation template if you are not actively participating in any exercises for some longer duration. Clear instructions for deleting the AWS CloudFormation template are provided within the README file itself.*

However, in the **genai-bedrock-book-samples** folder there’s another subfolder titled **chapter7**. The **README** file within **chapter7** folder provides clear instructions on launching a **Notebook** on Amazon SageMaker.

|  |  |
| --- | --- |
| File Name | File Description |
| simple\_knwl\_bases\_building.ipynb | 1. Create a vector store using OpenSearch Serverless collection and index.  2. Create Amazon Knowledge Bases.  3. Ingest data into the Amazon Knowledge Bases.  4. Test the Amazon Knowledge Bases.  **Dependency**: simple-sageMaker-bedrock.ipynb at Chapter 3 should work properly. |
| simple\_knwl\_bases\_retrival.ipynb | 1. Use the RetrieveAndGenerate API for Amazon Bedrock integration.  **Dependency**: simple\_knwl\_bases\_building.ipynb at Chapter 7 should executed properly. |
| simple\_knwl\_bases\_retrival\_langchain.ipynb | 1. Use LangChain retrieve and generate integration with Amazon Bedrock.  **Dependency**: simple\_knwl\_bases\_building.ipynb at Chapter 7 should executed properly. |
| simple\_knwl\_bases\_chunking\_strategy.ipynb | 1. Example of variety of Chunking Strategy  **Dependency**: simple-sagemaker-bedrock.ipynb at Chapter 3 should work properly. |
| simple\_knwl\_bases\_clean\_up.ipynb | 1. Cleaning resources helps reduce unnecessary expenses.  **Dependency**: All the above code of Chapter 7 |

# 3.8 Bedrock Interaction Sample Application

***Disclaimer****: Charges will apply upon executing above files. Therefore, it is important not to forget to clean up the kernel after studying the topic. Refer to the clean-up section for instructions on how to properly clean up the kernel.*

7.4 Overview of Chunking Strategy

Chunking is a key part of retrieval-augmented generation (RAG). It splits the data into smaller, easier-to-handle "chunks" to make retrieval faster and more accurate. Amazon Bedrock Knowledge Bases supports different chunking strategies. You will learn in this section in detail along with their advantages, drawbacks, and use cases.

**Fixed-Size Chunking**

Entire data is split into chunks of a predetermined size, such as 500 or 1000 characters or tokens in this approach. Each chunk is treated as a separate unit for embedding and retrieval.

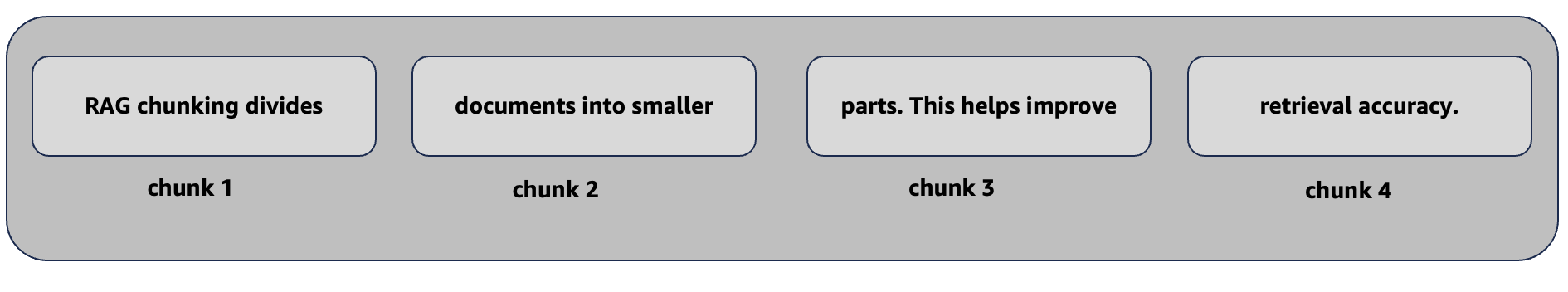


Figure 7.3 Example of Fixed-Size Chunking

|  |  |  |
| --- | --- | --- |
| Advantages | Drawback | Use cases |
| **Simplicity**: Easy to implement and comprehend. The chunks are uniformly sized. This makes retrieval easy.  **Efficient for Structured Data**: Works well when dealing with uniform, structured data, like logs, technical manuals, or data tables. | **Lack of Context**: Fixed-size chunks can cut off meaningful content in the middle of sentences or paragraphs, leading to potential loss of context.  **Reduced Relevance**: The retrieval process might miss important details if the information is spread out over different sections. | This approach is ideal for data with uniformly structured content, like user manuals, FAQs, and log files. In such cases, losing a bit of context between chunks has minimal impact. |

Table 7.1 Advantages, Drawback and use cases of Fixed-Size Chunking

**No Chunking**

This method treats the entire document as a single chunk and indexes it for retrieval.

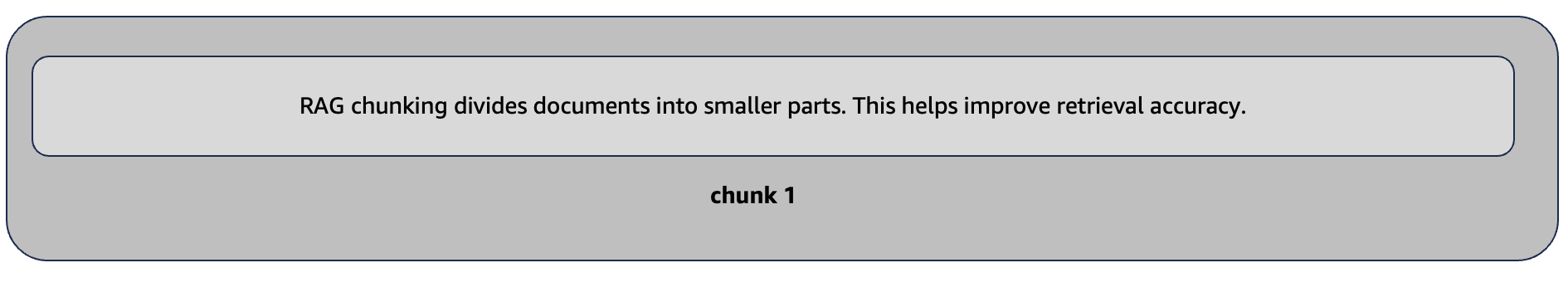
****

Figure 7.4 Example of No Chunking

|  |  |  |
| --- | --- | --- |
| Advantages | Drawback | Use cases |
| **Preserves Full Context:** Processing the document in its entirety eliminates the risk of losing crucial information that could arise from segmenting it into smaller parts.  **Ideal for Short Texts:** Works well with smaller documents, where dividing the content could dilute its meaning or context. | **Inefficient for Large Documents:** Processing or retrieving large documents in this manner can be computationally expensive and hindered by the limited context window of generative AI models with slower retrieval. | This approach highly effective for obtaining summaries from separate documents. It demonstrates significant efficacy for concise documents, including legal contracts. It maintains the complete context for accurate replies. |

Table 7.2 Advantages, Drawback and use cases of No Chunking

**Hierarchical Chunking**

This method systematically arranges segments within a hierarchical framework. The retrieval process looks at specific details and the overall context. It organizes smaller parts within larger frameworks.

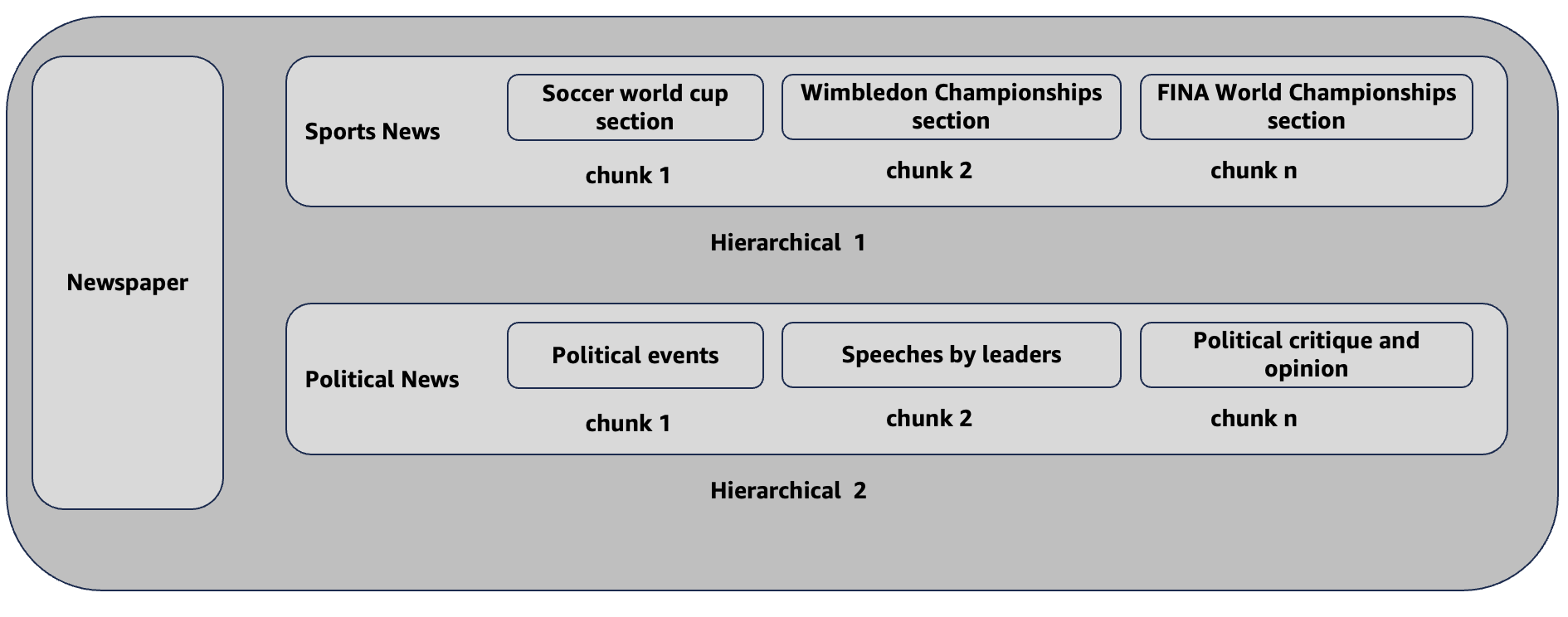
****

Figure 7.5 Example of Hierarchical Chunking

|  |  |  |
| --- | --- | --- |
| Advantages | Drawback | Use cases |
| **Preserves Structure**: The hierarchy ensures the document's logical flow while splitting it for effective retrieval.  **Context-Aware**: The hierarchy allows for retrieval based on both high-level overviews and detailed sections. | **Complex to Implement**: This is more difficult to build and manage. It requires better indexing and retrieval methods and more development work. **Computational overhead**: Managing hierarchical relationships can raise the system's computational needs. | It works best for intricate technical documents, books, or research papers where it's crucial to maintain both a detailed and high-level context. |

Table 7.3 Advantages, Drawback and use cases of Hierarchical Chunking

**Semantic Chunking**

Semantic chunking organizes content by meaning rather than size. It employs natural language understanding to form chunks that represent complete ideas or related sections, like paragraphs with similar themes.

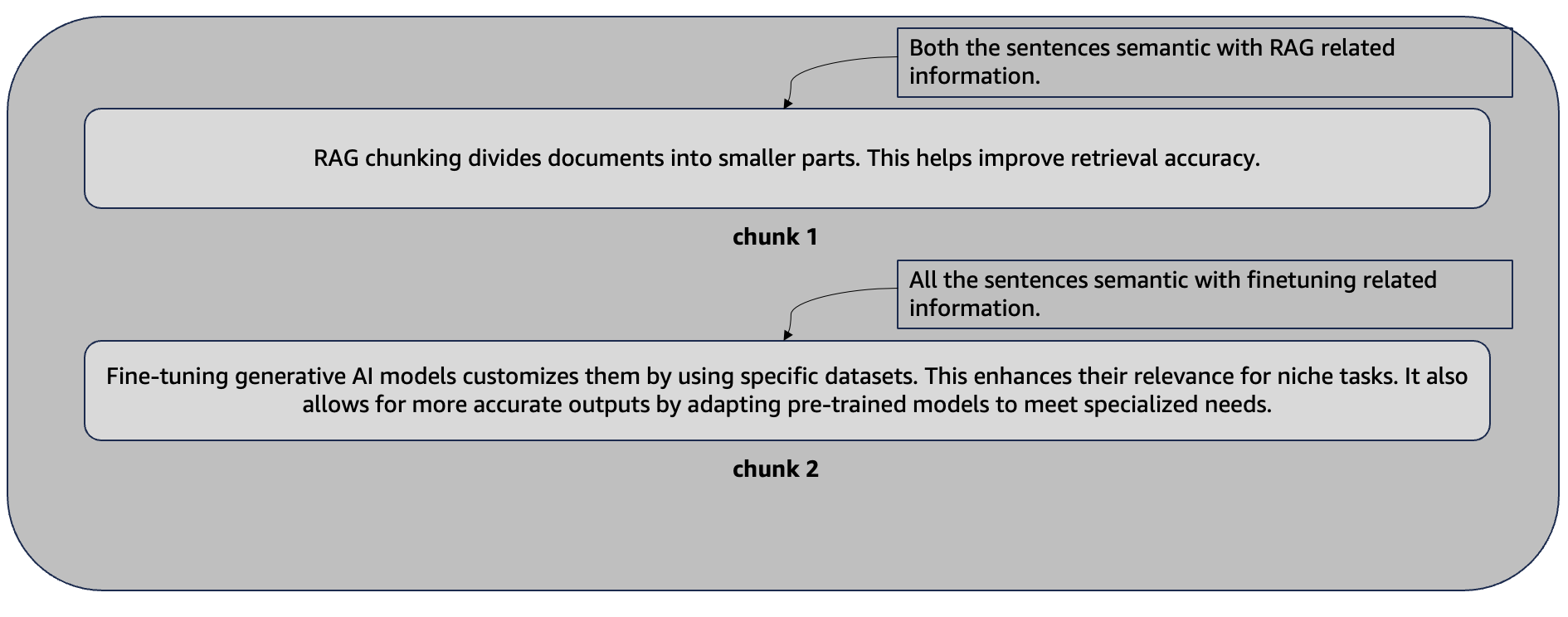


Figure 7.6 Example of Semantic Chunking

|  |  |  |
| --- | --- | --- |
| Advantages | Drawback | Use cases |
| **High Relevance**: Each chunk captures a complete thought, ensuring the retrieval of meaningful sections.  **Context Retained**: Semantic chunking preserves the natural flow of information, making it ideal for conversational AI or customer support tools. | **Complex Parsing:** Requires advanced natural language processing (NLP) techniques to identify appropriate chunk boundaries.  **Slower Processing:** The system may take more time to parse and chunk documents accurately, leading to slower indexing times. | This feature helps customer service databases and knowledge articles. It works well with complex documents. It ensures the information retrieved is accurate and relevant to the context. |

Table 7.4 Advantages, Drawback and use cases of Semantic Chunking

**Custom Chunking**

Custom chunking lets you create your own data splitting logic. You can use tools like frameworks such as LangChain and LlamaIndex. This gives you full control and flexibility to optimize for specific needs.

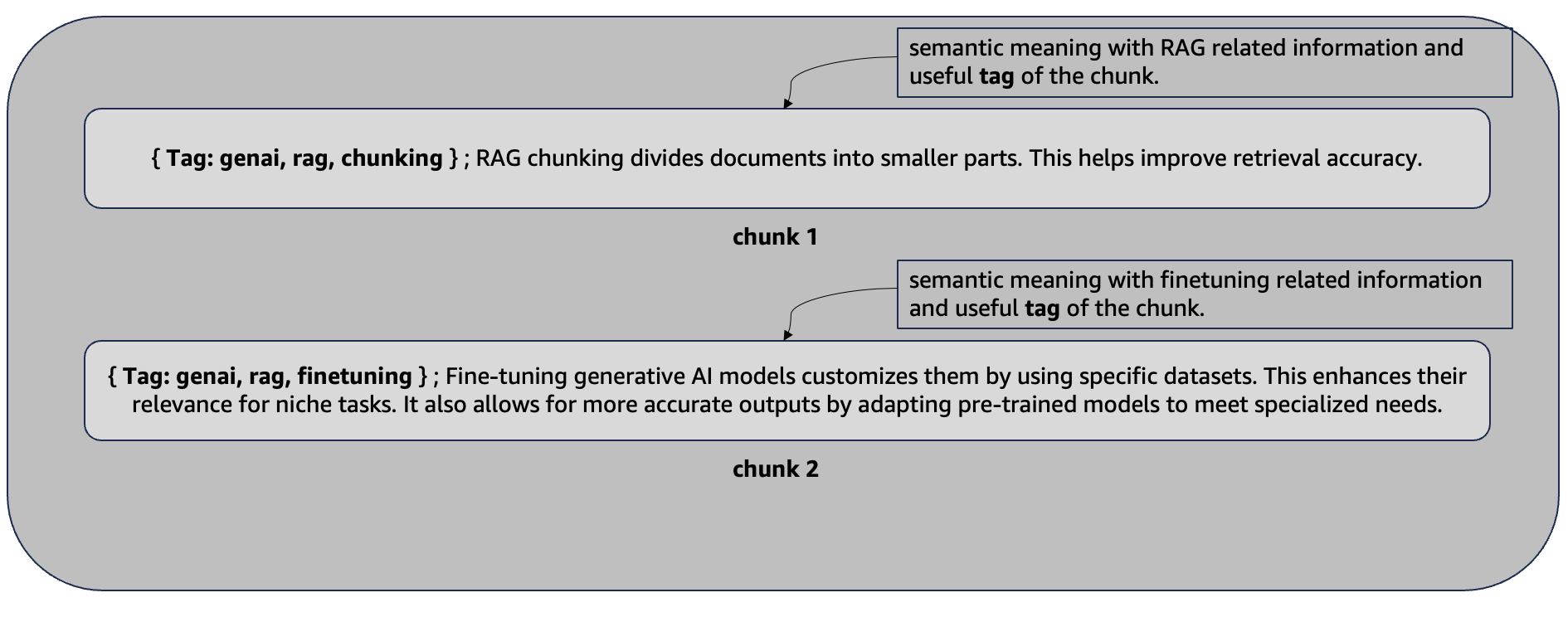


Figure 7.6 Example of one of the Custom Chunking

|  |  |  |
| --- | --- | --- |
| Advantages | Drawback | Use cases |
| **Highly Tailored**: You can modify chunking to meet specific business requirements, guaranteeing the most effective division of data.  **Flexibility**: Enables advanced methods and third-party tools for better chunking of various data types. | **Requires Expertise**: Custom chunking involves more development work and requires knowledge of how to optimize chunking for retrieval performance.  **Increased Maintenance**: As data structures or use cases change, custom implementations require maintenance and updates. | This method works well for specialized fields like healthcare and finance. These industries have unique data formats, such as medical records and financial reports. This requires specific chunking strategies. |

Table 7.5 Advantages, Drawback and use cases of Custom Chunking

**Choosing the Right Strategy**

Each chunking strategy has advantages and limitations. The best choice depends on your data and goals. Fixed-size chunking works well for simple, structured data. For complex documents needing context, semantic or hierarchical chunking is better. Custom chunking is for cases where standard methods don't meet your needs, giving you full control over data processing. (Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking-parsing.html> )

Organizations can evaluate the advantages and limitations of each strategy. This helps them select the best chunking method. The goal is to enhance their Amazon Bedrock Knowledge Bases. This will lead to better information retrieval.

7.5 Governance & Monitoring

Amazon Bedrock Knowledge Bases provide robust governance and monitoring functionalities to guarantee data integrity, security, compliance, and operational efficiency for generative AI applications. It supervises governance and oversight by evaluating these characteristics.

**Data Security & Access Control**: Amazon Bedrock Knowledge Bases integrate with AWS Identity and Access Management (IAM), allowing administrators to define detailed permissions for who can access, manage, or retrieve data from the knowledge bases. This guarantees that only authorized systems and individuals may access proprietary information, thereby protecting sensitive data. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/security-iam.html)

**Audit Trails**: Amazon Bedrock Knowledge Bases logs all interactions, including data ingestion, retrievals, and modifications. These audit logs help track your activities, providing visibility into who accessed or altered the knowledge bases and when. Compliance with industry regulations and internal security policies requires this level of traceability. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/logging-using-cloudtrail.html)

**Monitoring & Alerts**: AWS CloudWatch provides real-time monitoring of system performance and data flow within Bedrock Knowledge Bases. You can set up custom alerts to notify administrators about any unusual activity, system errors, or performance bottlenecks. Proactive monitoring ensures the swift resolution of any problems. It minimizes downtime. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-bases-logging.html)

**Failure Handling**: You can implement automated retries and error recovery mechanisms to effectively manage failures. If there’s an issue during ingestion or retrieval, APIs will attempt to resolve it without disrupting the overall workflow.

**Encryption**: Amazon Bedrock Knowledge Bases uses AWS-managed encryption services to encrypt data stored and transmitted. It supports encryption of data at rest and in transit. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/encryption-kb.html)

**Compliance**: Amazon Bedrock Knowledge Bases adhere to a variety of compliance standards, such as GDPR, HIPAA, and SOC. This makes it appropriate for sectors that must comply with rigorous data protection standards. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/compliance-validation.html)

By combining these governance and monitoring features, Amazon Bedrock Knowledge Bases provide organizations with full control, flexibility, and visibility over their data, ensuring secure, compliant, and reliable operations.

7.6 Design principals of right Vector DB

Choosing the right vector database is important for generative AI applications. It must efficiently store and retrieve high-dimensional vector embeddings. These embeddings represent the meaning of text, images, video, and audio. Here are some key design guidelines for selecting or creating a vector database. For further details, check the link. ( Refer: https://superlinked.com/vector-db-comparison )

**Semantic Search Capabilities**: The core function of the vector database is to perform semantic search—retrieving the most relevant text chunks or documents based on their vector embeddings' proximity to the query vector in high-dimensional space. This text provides important context for the prompt. It enhances result accuracy.

**Scalability for Vector Datasets**: Generative AI relies on large datasets containing billions of vector embeddings. Scalability is essential for these datasets. The database must support ongoing data ingestion, rebuild indexes, and efficiently search through large vector datasets. It should also ensure high performance and resilience.

**High Dimensionality Support**: A large number of modern embedding models generate high-dimensional vectors (1024, for example). The database needs to enable efficient, large-scale ingesting and searching on these high-dimensional vectors.

**Optimized Indexing Techniques**: To enable rapid nearest neighbour search in high-dimensional spaces, the database should implement advanced indexing algorithms like Hierarchical Navigable Small World (HNSW) or Inverted File with Flat Compression (IVFFlat). These methods reduce latency and performance.

**Configurable Relevance and Recall**: This is also important for generative AI applications. You should be able to configure the desired trade-off between these factors in the vector database, taking into account their specific needs and preferences and ensuring the retrieved results are sufficiently relevant and complete.

**Hybrid search and filtering**: Hybrid search and filtering combines traditional search methods with vector similarity search. You can utilize keyword matching, phrase matching, full-text search, and structured filtering. This approach enhances the precision and targeting of information retrieval.

**Strong Integration with ML/LLM Frameworks**: The vector database must easily connect with popular machine learning and large language model frameworks. This will support generative AI applications and make development and deployment simpler.

**Serverless and Fully Managed**: Serverless and fully managed vector database services are helpful for generative AI workloads. These simplify operations by automatically adjusting resources based on demand.

Some of the vector database offering from AWS which could be consider during design.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Properties | Memory | Document | Graph | Search | RDBMS |
|  | Amazon MemoryDB | Amazon DocumentDB | Amazon Neptune | Amazon OpenSearch | Amazon Aurora/ RDS with pgvector |
| Index | HNSW, FLAT | IVFFLAT, HNSW | HNSW | IVFFLAT, HNSW | IVFFLAT, HNSW |
| Max Dimensionality | 32768 | 16k, 2k index | 65535 |  | 16k, 2k index |
| Max Vectors | Millions | Billions | Billions | Billions | Billions |
| Serverless | No | No | No | Yes | Yes ( Aurora) |
| Full Text Search | No | No | No | Yes | Yes |
| Hybrid Search | No | No | No | Yes | No |
| Quantization | No | No | No | PQ, SQ | SQ |

Table 7.5 Type of Vector DB and properties

By following these design rules, a vector database can become the best way to store and retrieve information for adding relevant external knowledge to generative AI models. This opens up new possibilities for smart and aware applications in many areas.. This enables more refined and targeted retrieval of relevant information.

7.7 Summary

The chapter emphasizes Amazon Bedrock Knowledge Bases. This is crucial for generative AI applications. It shows how it simplifies retrieval-augmented generation (RAG) by managing external data access and enhancing Large Language Model (LLM) outputs. You learned that the benefits include a smooth RAG workflow, contextual AI with proprietary data, secure data connectivity, flexible data ingestion, support for multi-turn conversations, customized retrieval, and no need for an external vector database.

You also learned chunking strategies for efficient information retrieval from large datasets. Strategies include fixed-size, no chunking, hierarchical, semantic, and custom chunking, each with its pros, cons, and use cases. Choosing the right strategy depends on the data type.

The chapter covers governance and monitoring aspects, emphasizing data security, access control, audit trails, monitoring, failure handling, encryption, and compliance.

The chapter emphasizes designing an effective vector database for storing and retrieving high-dimensional vector embeddings. Key features include semantic search, scalability, high dimensionality support, optimized indexing, configurable relevance, hybrid search, and strong ML/LLM integration.

The design process should consider AWS vector database options such as Amazon MemoryDB, Amazon DocumentDB, Amazon Neptune, Amazon OpenSearch, and Amazon Aurora/RDS with pgvector.

Consider AWS's vector database offerings during the design process.

By adhering to these design guidelines, a vector database can emerge as the optimal method for storing and retrieving information, thereby incorporating pertinent external knowledge into generative AI models. This opens up new possibilities for smart and aware applications in many areas. This enables more refined and targeted retrieval of relevant information.