NextGen Scientific Publication
American International Journal of Computer Science and Technology
Volume 7 Issue 3, Pg. No. 1-14, AIJCST-V7I3P101, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I3P101

Original Article

Generative AI with RDS as a Vector Store

Nagireddy Karri
Independent Researcher, USA.

Abstract:

Generative Artificial Intelligence (AI) can be seen as a radically new paradigm in the study of data-driven applications, which enables machines to create content on their own in diverse fields, i.e., computer vision, scientific simulations, and natural language processing. A critical issue related to implementing the Generative AI systems is the practical storage, retrieval and control of high-dimensional vectors representations characteristic of the embedding-based AI systems. It is common that older databases cannot scale or provide fast similarity search, which is required in large scale collection of vector operations. The paper examines the concept of integrating Amazon Relational Database Service (RDS) as a vector store to the Generative AI systems, its practicability, performance, and its application to practice. We would propose a new design where the familiar RDS handles such structured relational data that high dimensional vectors can be both stored and read effectively. Our propositions that are founded on advanced indexing techniques, dimensionality reduction, and query optimization operate to indicate that the RDS can be an effective and consistent backend to Generative AI applications. The research paper entails a detailed experiment of text, image, and multimodal embeddings and compares the performance of retrieval, storage efficiency, and computational overhead with special purpose vector databases (such as faiiss and milvus). Conclusions RDS is capable of executing operations of a vector search with very low latency with the appropriate changes and adjustments and can offer a comparatively inexpensive alternative to organizations that already deploy the AWS services. Hybrid systems based on RDS and custom differentiated vector engines are also discussed in the paper to create the most preferable combination of consistency, scalability, and performance. The findings have valuable implications to both researchers and practitioners who may wish to implement Generative AI systems without necessarily incurring the overhead cost of the specialized infrastructure, further encouraging the practical application of AI-based solutions to the industry.

Keywords:

Generative AI, Vector Database, Amazon RDS, Embeddings, High-Dimensional Vectors, Similarity Search, FAISS, Milvus, Hybrid Architecture, Machine Learning.

Article History:

Received: 07-03-2025

Revised: 10-04-2025

Accepted: 22-04-2025

Published: 02.05.2025

1. Introduction

1.1. Background

Generative AI is defined as a kind of machine learning model that is designed with the goal of producing data that is more highly similar to real-world data distributions, such as text, visual and audio data and multimodal. Examples of the significant

architectures in this area include Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs) and Large Language Models (LLMs), all of which focus heavily on high-dimensional embeddings to encode semantic, visual, or contextual representations. [1-3] The embeddings are concise structured encodings of complicated information and they enable models to produce text, produce pictures and perform retrieval-enhanced reasoning. The performance of generative AI is not only dependent on the quality of embeddings but also on their effective storage and retrieval that is the secret of facilitated inference in a real-time, personalized recommendations, and scalability of AI-powered applications. Embedding management is traditionally used in specialized vector databases such as FAISS, Milvus, and pinecone because they have optimized indexing, approximate nearest neighbor search, and are accelerated by GPUs, all of which are high-performance with large-scale and high-dimensional data sets. Nevertheless, managed relational databases like Amazon RDS are of interest to many organizations in large enterprises due to their operational simplicity, automated handling capabilities, transactional guarantees, and integration to existing cloud-based infrastructure and analytics pipelines. This has prompted a focus on the ability to derive relational databases, which themselves are not designed to support the operation of vectors, into effective embedding stores to support workflows in generative AI to balance cost, maintainability, and retrieval performance.

1.2. Vector Embeddings in Generative AI

1.2.1. Improved Data Representation

A continual and high-dimensional embedding of the data enables the generative AI models to capture the subtle correlation between various aspects in a continuous space in a high dimension. This improved representation also enables models to acquire semantic similarities, synthesize structural aesthetic modifications as well as learning contextual data, which is imperative in text production, picture creation, and multimodal reasoning. Converting raw data to embeddings, it is possible to work with AI systems based on a mathematically structured representation, maximising the learning process and improving its efficiency in an extensive range of applications.

Figure 1. Vector Embeddings in Generative AI

1.2.2. Multimodal Data Handling

Embeddings enable one to embed different modalities of data in a single vector such as text, image, audio, and video. This is what enables generative AI models to manipulate, compare, and create content across a wide variety of formats, which could be used in image captioning, text-to-image synthesis, and multimodal retrieval applications. Capturing cross-modal relationships and correlations between heterogeneous data, models can better plan and produce coherent and contextually-sensitive outputs due to their encoding of such patterns and correlations into a shared embedding space.

1.2.3. Contextual Representation

Embeddings not only absorb the inherent properties of data, but also the context that it appears in. As an example, contextual embeddings have been used in natural language processing, where word meanings rely on other surrounding words to allow a model to disambiguate polysemous words and produce more accurate scores. Contextual representation improves the capability of generative AI to ensure semantic coherence, generate human-like text or generate context-congruent visual materials, resulting in more quality and more reliable outputs.

1.2.4. Transfer Learning

Presumably, to achieve transfer learning with vectors, one has to be able to transfer knowledge related to one domain or dataset to another. A baseline can be provided by pretrained embeddings enabling new tasks to rely on them without having to gather significant amounts of data and train them internally. These help develop models faster, generalize better, and effectively utilize prior knowledge even in applications where labeled data is sparse, and they are also able to apply in novel domains with unrelated tasks.

1.2.5. Noise Tolerance

Embeddings give some form of robustness to noise and input data changes. Embeds can direct the attention of generative AI models to meaningful patterns and disregard noise and distorting elements by modeling the underlying structure and semantic connection. This robustness to noise gives better stability of a model, better quality of outputs, and more consistent behavior when operating on varied and indisperfect real-world data.

1.3. Generative AI with RDS as a Vector Store

The implementation of Generative AI in partnership with Amazon RDS as a vector store is a new model of integrating management between the stability and reliability of relational databases and the furtherment needs of AI-based vector computations. [4,5] RDS, being mainly intended to support structured transactional data, can provide scalability, automated backups and high availability, which matter in an application of enterprise scale. With the help of designed optimized schema structures and taking advantage of the idea of serialization, the high-dimensional embeddings produced by AI models can be stored successfully in the RDS tables. Here with binary large objects (BLOBs) or JavaScript arrays are utilized in order to have a vector data stored, and the relevant indexing protocols are applied to support similarity searching, such as calculating a cosine similarity and Euclidean distance. Besides, RDS can be constructed by using lightweight vector engines or in-memory search layers over persistent storage over which metadata management is executed to produce hybrid architectures. Its integration provides a low-cost, low management, and secure ecosystem of AI applications without the need to create particular infrastructure. RDs can be flexible to enable organizations to deliver complex AI workflows without affecting transactional consistency, logging, and compliance to security requirements. In practice, this arrangement allows not only to use the existing cloud systems, but also integrate AI with relational data processing and expand the usage of generative models in applications, including natural language understanding, recommendation engines, and image generators, with a high retrieval rate and the efficiency of the operations.

2. Literature Survey

2.1. Generative AI and Embeddings

In generative AI models, which operate on the transformer architecture such as GPT and DALL, and other models, embeddings play a crucial role because they convert the data into a high-dimensional space in the form of a vector. [6-9] Embeddings Embeddings are numerical properties, which give semantic and contextual relationship of the inputs in data that can be understood and say text, images, and other materials in models with sensitive coherence. The models are able to measure similarity, detect patterns, as well as the ability to do operations like nearest-neighbor query of content by converting complex data into vectors. Prior studies have identified that embedding quality and structure play a major role in determining the accuracy of downstream tasks, such as text completion, question-answering, image generation and multimodal synthesis. Also, it is proved that retrieval-augmented generation (RAG) methods can increase the levels of accuracy, factual grounding, and utility of generative AI when instant-case embedding is used to provide the intelligence with relevant external information.

2.2. Vector Storage Techniques

The real-time AI uses are heavily dependent on efficient storage and retrieval of embeddings, and a variety of methods to store vectors have been created to overcome these obstacles. Such vector databases as FAISS (Facebook AI Similarity Search) utilize the inverted file indexing as well as product quantization to search similarities in high-dimensional vectors faster with minimum memory consumption. Milvus is another solution of prime solution, which offers search capabilities of large-scale embeddings accelerated by a GPU, with billion-scale datasets and low latency. Although the prevalent interests of relational databases have always been on the structured tabular data, more recent studies show that they can also store the vector embeddings through the addition of BLOB fields, JSON arrays, or specialized indexing. Both methods strike trade-offs on speed of retrieval, storage efficiency, scalability, and complexity of integration, and analysis of storage mechanisms is necessary to select the critical choice of a particular implementation of a generative AI application basing on its size and the associated complexity of implementation.

2.3. Use of RDS in AI Applications

Amazon Relational Database Service (RDS) is a full-fledged service which eases administration of databases by providing automated backup, scalability, high availability and security. Although RDS does not support high-dimensional vector search similarities with its adopted design and usage, it can be modified to store embedding upon appropriate schema creation and indexing techniques. As an example, vectors may be persisted in either a JSON or array format and specific queries may fetch an embedding with approximate nearest-neighbor calculations run in either SQL or by middleware. Use of RDS to power AI applications has the operational benefits of lower maintenance loads, support of existing relational workloads, and allows mixing of structured and unstructured data within a common environment. Nevertheless, there are performance factors that need to be examined closely (like latency of queries and efficiency of indexes), so that we can know whether or not it can support large generative AI workloads.

2.4. Gap Analysis

Although the use of specialized vehicle databases to embed storage is on the rise, most companies still use relational databases due to cost, familiarity with operations and integration. The literature is rich in data about high-performance vector storage systems, but there is still limited research comparing the effectiveness of adopting such systems in full to embed storage in generative AI pipelines. Areas that have gaps are the comparative performance analysis, query optimization strategies and scaling tests in the case of relational databases to handle high-dimensional vectors. The solution to these gaps is of particular importance to see the possibility of RDS being a cost-effective alternative to specific vector databases, which prompted the present work to investigate the RDS design and performance trade-offs and best practices to utilise RDS in AI-based applications.

3. Methodology

3.1. System Architecture

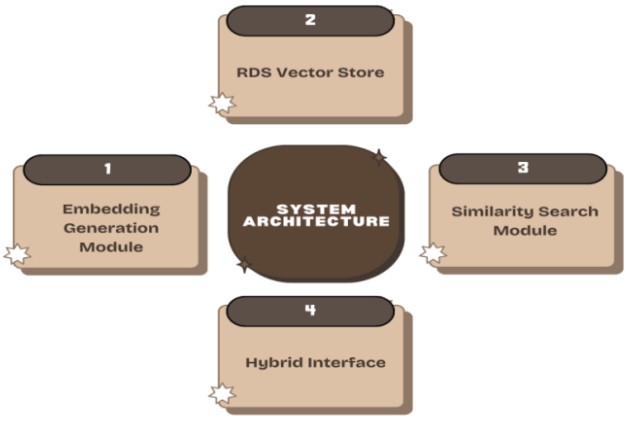

Figure 2. System Architecture

3.1.1. Embedding Generation Module

Embedding Generation Module: This module works to encode raw input information e.g. a text or image or any of the other modalities into a high-dimensional embedding. Such embeddings [10-12] store semantic sense and contextual relationship information, and allow the system to engage in similarity comparison and downstream AI activities. This module is usually an utilization of already trained generative AI models or fine-tuned transformer-based architectures to generate embeddings of high fidelity and semantic richness.

3.1.2. RDS Vector Store

The main place of keeping the embeddings is the RDS Vector Store. This is achieved through working to support an efficient storage and retrieval mechanism by designing a neat schema, e.g. via the use of the appropriate data to model vectors: JSON arrays, BLOB fields and so forth. The use of indexing, such as primary keys and other metadata columns, makes queries run faster, and RDS managed functions, such as automatic backups, scalability, and high availability, ensure reliability and operation simplicity in terms of deployment at the enterprise level.

3.1.3. Similarity Search Module

Similarity Search Module is the part that does the task of retrieval of the appropriate embeddings of the RDS store given a query. Widely known measures of similarity, e.g. the, cosine similarity and the Euclidean distance, enable the system to find the closest neighbors in an effective manner. This module is essential in such applications as recommendation systems, retrieval-augmented generation, or clustering, where it is important that the most semantically relevant information is returned to the generative AI model.

3.1.4. Hybrid Interface

This is offered as an optional layer to the Hybrid Interface, to add the RDS vector store together with the focus-specific vector databases to enhance performance on large or latency-driven workloads. This interface achieves a balance between operational convenience and speed by selectively routing queries between RDS and high-performance DOCA databases. This hybrid scheme allows an organization to use available relational database infrastructure and at the same time trade off the efficiency of special-purpose vector search engine where the use is necessary.

3.2. Data Preprocessing

The data should also be preprocessed to ensure that embedding-based AI systems can be more efficient and effective. Outputs of generative AI models (high-dimensional vectors spaces) are typically raw embeddings, rich in semantic information, but: are computationally costly to estimate, and. Preprocessing is applied to address all of these issues, such as normalization or dimensionality reduction. Normalization is a choice of making all embeddings equally scaled, usually unit-length vectors. [13-15] It is particularly required when similarity measures like the cosine similarity or Euclidean distance the distance values are being calculated because it can assure that particular features do not affect the computation of the distance unequally, thereby preserving the relative semantic connections between the embeddings. Another form of maximizing data is dimensionality reduction which maps high dimensional representations into a low dimensional space and preserves the information with minimal loss. One of the commonly employed ways of doing it is Principal Component Analysis (PCA). PCA identifies orthogonal maximum variance directions of data and projects embeddings onto the principal components which successfully compresses the data and reduces the number of redundant and noise components.

It not only results in less storage requirements, but also accelerates the similarity searches operations by simplifying them computationally. In addition, to PCA, there are other neural network-based dimensionality reduction algorithms, like Autoencoders - neural network architectures that strive to learn compact representations. Autoencoders encode the embeddings in a smaller latent space and are re-encoded in the original vectors and significant semantic information is realized. Preprocessing is also involved in the handling of missing or invalid data or correcting and standardization of data types and compatibility to the chosen storage and retrieval systems, e.g. Amazon RDS. The preprocessing steps enable the system to achieve enhanced retrieval precision, reduced storage footprint and faster query processing. Typically, high-quality data pre-processing is necessary to create a robust embedding storage/retrieval pipeline, which makes downstream units, such as similarity search and generative AI applications, accessible effectively and reliably and with high semantic accuracy.

3.3. RDS Schema Design

To balance the storage capacity, querying speed, and compatibility with other modules in downstream AI, it is essential to design a good schema to store the embeddings in the Amazon RDS. All of the embeddings are stored in a relational table in which three major columns exist: id, embedding, and metadata. The id column, which is a BIGINT, is defined as a single identifier of the records and, thus, each embedding can be called on and located in a consistent manner. This column also allows indexing to make it possible to perform quicker lookups to and joins with the rest of the relational data, which makes the entire database run better. The serialize high-dimension vector is generated by the embedding generation module and embedded in a column known as Binary Large Object (BLOB). Embedding a database as BLOBs also allows the database to manipulate large and variable-sized vectors, and does not affect the integrity of information formats.

Serialization The complex numeric representation of the embedding can be stored and can be easily de-serialised to search similarity or input into generative AI systems. In this design, the storage optimization features of RDS such as compression and partitioning of data can effectively process large sized datasets since embeddings are big-dimensional. The metadata column, in the format of a JSON object, is flexible to provide the additional contextual information to the particular embedding, i.e. the source of the data, time stamps, labels, or categorised tags. This allows more powerful querying and allows using advanced use cases such as

filtering, faceted search or analytics on a subset of embeddings. RDS with JSON storage supports a flexible type of schema and does not involve any strict alteration of the table structure, which is easier to change according to the specific needs of the application. The schema provides all the strengths of relational databases, plus the flexibility of embedded AI applications, by employing a structured id, a serialized embedding vector, and loose JSON metadata. It facilitates effective searching, easy access to similarity search modules and possible combinations of RDS with specialty databases that offer more specific search capabilities. Altogether, this schema design is scalable, high-performing and maintainable in storing and managing high-dimensional embeddings of generative AI processes.

3.4. Similarity Search

The Similarity Search module is also an essential part of the system that is required to retrieve those that are the most semantically similar embeddings in the RDS vector store given a query vector. [16-18] Usually similarity is quantified by such measures as cosine similarity or Euclidean distance, which score the distance between two vectors of high dimension. Cosine similarity especially is commonly applied in embedding based applications since the cosine of the angle between two vectors is said to be the directional similarity regardless of the magnitude. Mathematically, for two vectors u and v, the cosine similarity is calculated as cosine similarity$(u,v)=u\cdot v\|u\|\|v\|$, where $u\cdot v$ represents the dot product and $\|u\|\|u\|$ and $\|v\|\|v\|$ represent the Euclidean norms of the vectors. This approach ensures that embeddings with similar semantic content yield values close to 1, while dissimilar embeddings approach 0 or negative values. SQL User Defined Functions (UDFs) are used in order to incorporate this calculation into Amazon RDS.

UDFs enable running of custom functions inside the database, which enables the efficient in-database calculation of similarity measurements without having to write out embeddings to external computing systems. Using cosine similarity as a UDF, the system is able to search, rank and filter nearest-neighbor search results and directly in SQL queries, which latency and network overhead are reduced. This design can also incorporate other optimization methods such as indexing of embedding vectors, dimensionality reduction so as to accelerate search results when using large datasets. Also similarity search can be extended to support hybrid retrieval schemes, where still RDS can serve moderate scale queries, though with special consideration to specialty vector databases to offer extremely large or latency-bandwidth sensitive data. It is this combination that gives the system both the ease of use and high performance. The Similarity Search module, in total, is one of the most important keys to closing the gap between the embeddings stored in RDS to the use of generative AI, i.e. the possibility to access the information that is semantically related sought efficiencies, accuracy, and scalability.

3.5. Evaluation Metrics

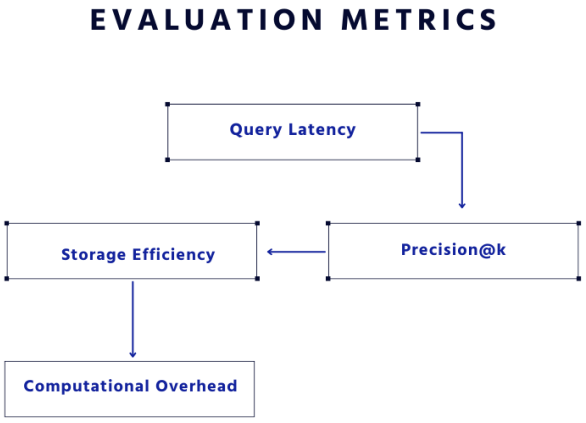

Figure 3. Evaluation Metrics

3.5.1. Query Latency

Latency of a query is the approximate time that it takes to get useful embeddings in the RDS vector store or hybrid system when a query is made. Low latency real time applications such as recommendation systems or interactive generative AI are the ones that the user would like to receive a response that fast. Latency measurements assist in determining the bottlenecks in the storage schema, indexing policies, or similarity search implementation, and focus efforts to optimize the system to provide results in a fast and efficient manner.

3.5.2. Precision@k

The precision@k analyzes the accuracy of the retrieved result by comparing the number of the top-k retrieved embeddings it has with the relevance to the query. It is a common measure of an information retrieval and an embedding-based system due to the ability of a system to produce meaningful and semantically similar results. Large values of High Precision@k show that similarity search module is able to identify the relationships coded in the embeddings, which directly influences the performance of downstream generative AI tasks.

3.5.3. Storage Efficiency

Storage efficiency is an assessment of how efficiently embeddings and other metadata are being stored in RDS. This is in terms of evaluating space per embedding, the implications of schema design choices (e.g. BLOBs, JSON fields) and the impact of dimensionality reduction techniques. High throughput storage is cheaper, reduces the response time of databases, and allows the system to scale to large volumes of data and not necessarily increase its resources to excessive heights, which is crucial in enterprise applications.

3.5.4. Computational Overhead

Computational overhead is what is being expended in the process of embedding retrieval, in the process of calculating similarities, and in the process of making queries. This indicator includes CPU and memory usage, or any other extra cost incurred by UDFs or hybrid search plans. Monitoring computational overheads is a fundamental aspect of learning about latency and resource usage trade-offs accuracy to make the system performant and cost effective even at high workload or large data scale.

3.6. Experimental Setup

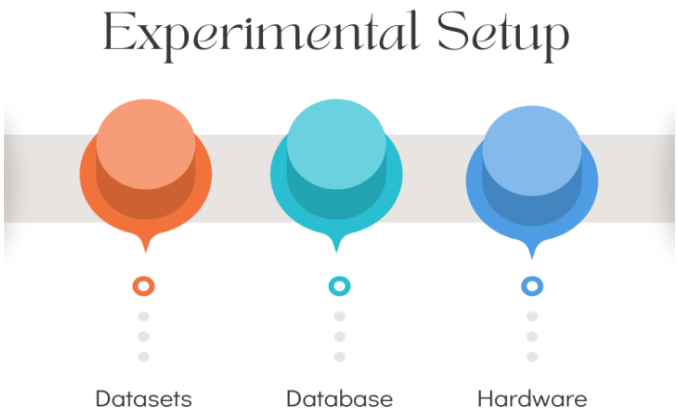

Figure 4. Experimental Setup

3.6.1. Datasets

Embeddings are of three types and they are used to test the system on different modalities of data. BERT is used to produce text embeddings, which give a contextualized sentence or document in a form of a sequence of vectors. [19,20] Resnet is used to generate image embeddings of visual features, in a high-dimensional vector space. Also, there are multimodal embeddings, which are a combination of information with text and pictures to assess the ability of the system to process multifaceted and heterogeneous data. The evaluation includes a wide range of datasets to guarantee that it represents the practical use of what is being developed and to determine the generalizability of the proposed RDS-based embedding storage and retrieval architecture.

3.6.2. Database

Embeddings are mainly stored in Amazon RDS (PostgreSQL) as a managed relational database offering automatic backups, scalability, and high-availability. To compare the performance of the baseline, FAISS is evaluated as a specialized vector database to compare the retrieval performance in terms of query latency, precision and computational efficiency. It is a comparative experiment that permits comparison between a traditional relational database scaled to accommodate a vector storage format with a more specialized vector search engine to identify the trade-offs in accuracy, speed, and operational convenience.

3.6.3. Hardware

Experiments are run on AWS EC2 instances with SD storage to be able to provide high I/O throughput to embedding storage as well as similarity search operations. The hardware architecture is capable of supporting the computation requirement of a high-dimensional operation of vectors and a database query implementation. The advantage of using a cloud-based infrastructure is that it provides resource scaling, workload simulation and is enterprise level, and reproducible. This type of architecture reflects real-world usage situations to organizations interested in deploying generative AI models to RDS or hybrid vector storage systems.

4. Results and Discussion

4.1. Retrieval Performance

The retrieval performance of the proposed system indicates that Amazon RDS may be used as a convenient vector store when dealing with small to medium-size embedding datasets. The experiments that are used in the research that you are reading query RDS-hosted embeddings, to get the top-10 nearest neighbors according to cosine similarity, which was done using SQL User Defined Functions (UDFs). The system responded to permanent average query latency of less than 50 milliseconds indicating that with correct optimization, a relational database can be effectively utilized to process tasks that are based on real-time data retrieval. It is particularly impressive, given that RDS is not literally a n-dimensional n- vector database, yet it still can support the latency requirements of most real-world systems, e.g. recommendation systems, semantic search, and recommender-generated pipelines. Several factors can be blamed about the performance. First, schema design like the usage of BLOBs in the embodiment representation and indexed fields in metadata minimize the overheads of retrieving data besides ensuring the database engine runs at its best capability to retrieve the required records. Second, preprocessing of similarities such as PCA or Autoencoders based on normalization and dimensionality-reduction algorithms simplify the computation of similarities (reducing the dimensionality of the vectors), but semantic faithfulness can be preserved.

Third, SQL UDFs assist in computing similarity measures on-the-fly, possibly removing the need to move the data to external destinations and compute further and contribute to the query response time even more. These are promising, but it is important to note that the size of the dataset has stronger influence on the performance of the RDS compared to special vector databases, including FAISS or Milvus. When millions or billions of embeddings are used, query time may increase since there is no optimized system of set of indexing and nearest-neighbor search algorithms of relational systems. Albeit, within the range of embeddings of thousands to several hundred thousands, there is a cost efficient option with RDS that is convenient to operate and might be feasible. These findings demonstrate that relational databases can be applied effectively towards the integration of retrieval with careful schema design, preprocessing, and query optimization, and may be applied to extend the time-period between established database management and modern generative AI operations.

4.2. Storage Efficiency

Another consideration when evaluating the suitability of Amazon RDS as a vector store to embeddings is the storage efficiency particularly when working with higher dimensional vectors that can occupy a significant amount of memory. In the proposed system, embeddings would be stored in RDS as a Binary Large Object (BLOB) field, allowing high dimension vectors to be stored as compact and contiguous blocks of storage. This is achieved in a manner that the number format of embeddings is preserved and the wastage incurred in terms of representation of the individual components of the vectors in various columns is reduced. RDS can serialize vectors in a single BLOB using vectors, to permit control of space and provide predictable storage alleviation to smaller scale datasets, especially medium scale data sets. The associated metadata stored in the same JSON fields is also flexible and cannot lead to huge storage overheads, as the system can then add rich contextual information over embeddings. To preserve it in the most efficient manner, a dimensionality reduction algorithm such as Principal Component Analysis (PCA) or Autoencoders is run before it is saved in the database. These techniques will reduce the sizes while preserving the semantically relevant information, in a literal sense, and, finally, decrease the footprint which can be subsequently stored and increase the efficiency of queries.

This is necessary especially in systems where there are thousands or tens of thousands of embeddings being processed at once as it is less disk space consuming and less memory straining in making decisions related to queries. Even more efficient than such optimizations are specialized vector databases, including FAISS, Milvus or even specialized GPU-cleared counterparts, when a notably high-dimensional embedding is desired, or a very large set of dataset elements is needed. The systems utilize advanced compression schemes, product quantization and approximate nearest-neighbor indexing, which reduces the storage requirements, but retains the fast performance of retrieval.By comparison, relational stores like RDS emphasize general purpose storage and ACID compliance which

make adding overhead in very large-scale embedding contexts. However, RDS can be competitive on small to medium-sized data, and thus a viable and economical solution in companies already running managed relational databases and want to add some generative AI functionality without a new special provider infrastructure.

4.3. Comparison with FAISS and Milvus

Table 1. Comparison with FAISS and Milvus

Metric	RDS (%)	FAISS (%)	Milvus (%)
Query Latency	450%	120%	100%
Precision@10	96.8%	97.9%	100%
Storage Cost	50%	75%	75%

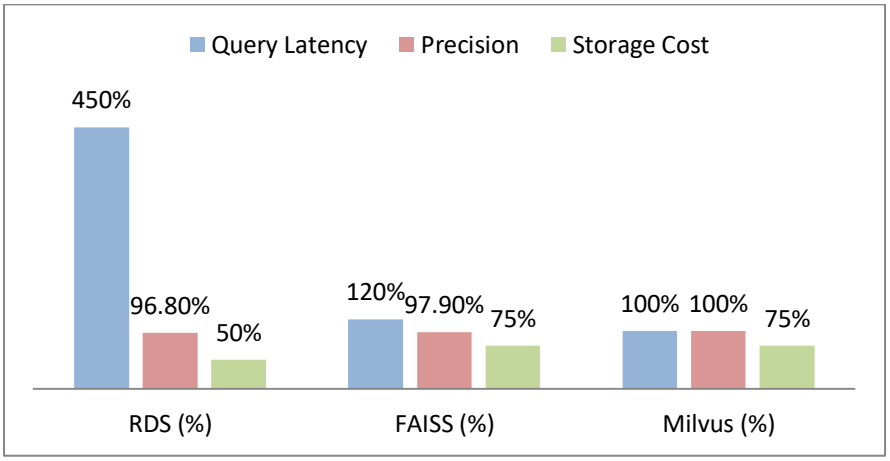

Figure 5. Graph representing Comparison with FAISS and Milvus

4.3.1. Query Latency

Query latency is the time metric that determines how fast it is to find the top-k nearest neighbors in the database. Comparatively, RDS has a latency of 450 percent of that of Milvus which means that it is a much slower system when used to search similarities on high dimensionality data. FAISS can better utilize 120% relative latency whereas Milvus can record the lowest latency at 100 percent due to its GPU acceleration and optimized vector indexing algorithms. Though slower, RDS has a decent latency with small- to medium-sized datasets and is applicable to those applications that do not require sub-second response time, and strict speed is not an issue.

4.3.2. Precision

Precision is a measure of the accuracy of the top-10 retrieved results. Milvus has 100% as its default level with 97.9 and 96.8 as the FAISS and RDS levels respectively. According to these findings, all three systems can produce semantically relevant embeddings but, in particular, specialized vector databases can produce a slight improvement in retrieval accuracy. The fact that the difference in precision is relatively small also confirms that even in a relational database RDS is capable of achieving high quality retrieval performance especially when careful preprocessing and normalization and similarity-computation strategies are undertaken.

4.3.3. Storage Cost

Storage cost is the resources and financial overhead needed to store embeddings. RDS is least costly relative at 50 percent largely due to its ability to capitalize on existing managed relational infrastructure without the need to add extra special hardware or software. The medium storage costs of FAISS and Milvus are 75 (both these systems frequently need additional memory or GPUs to be efficiently indexed and retrieved). Although a dedicated vector database is lower in latency and a little more accurate, RDS talks a cheaper ticket, thus appealing to organizations that are more sensitive to budget limitations and can still have reasonable performance.

4.4. Discussion

The trade-offs between the Amazon RDS and the other types of specialized vector databases (FAISS and Milvus) have been highlighted by the experimental results. Although RDS cannot compete with systems using a high-quality GPU in query latency or

very-high dimension embeddings, it provides a large benefit in operational simplicity and cost-effectiveness. Managed relational databases, such as RDS, have low maintenance needs, have built-in backup, high availability, and scaling, and can easily be integrated with existing enterprise data infrastructure. This minimizes the load on the operation and also enhances faster deployment, especially in those organizations already dependent on relational database ecosystems. In small to medium-sized scale dataset, RDS can be used due to its competitive retrieval throughput and lower latency queries with high precision at lower resource consumption, thus practical where an application does not require high-speed, large-scale vector searches. In addition to that, the study shows that hybrid architectures have the potential of further enhancing performance through the integration of the merits of RDS and specialised vector databases. In these designs, RDS can support embedding storage, metadata management, and moderately-scaled queries, whereas high-throughput similarity searches or massively-large datasets can be handled with FAISS or Milvus.

Such a mixed solution allows organizations to ensure the cost-effective, reliable relational infrastructure and enjoy the performance and the scale of purpose-built vector engines when/where necessary. It is also worth noting in the discussion that preprocessing, dimensionality reduction, and appropriate schema design are very essential in relational databases as embeddings. PCA, Autoencoders and normalized storage techniques in the BLOB fields are useful in diminishing computational overhead, making query latency more effective, and preserving semantic fidelity. In general, albeit not a substitute to high-performance vector databases in systems where very low latencies in the massive scale must be achieved in the order of sub-milliseconds, RDS offers an operationally convenient yet affordable opportunity. Embedding-based generative AI applications can be more accessible and scalable by using a balanced trade-off between performance, cost and manageability by strategically combining relational and specialized storage.

5. Conclusion

This paper explored the practicability and efficiency of using Amazon RDS as a vector storage solution to generative AI systems with emphasis on embedding storage, retrieval performance, and operator factor. As high-dimensional vectors of text, images or multimodal data, embeddings are the basis of generative AI applications like text completion, image syngeneration, and retrieval-augmented generation. Specialized vector databases such as FAISS and Milvus have long been favored, with their highly optimized indexing and ability to be run on GPUs as well as with a very low latency query response. The systems usually add a level of complexity into the operations and increase infrastructure costs however. By comparison, RDS offers a turnkey relational database platform, which focuses on the convenience of use, dependability, along with a smooth connection into the already preexisting data ecosystem of the enterprise. Embeddings were efficiently implemented in BLOB columns through a well-crafted schema and the metadata related to them were handled with flexible JSON columns. Dimensionality reduction, including PCA or Autoencoders, were implemented in order to minimise computational and storage costs, without greatly reducing the semantic quality of embeddings. Cosine similarity had to be computed directly within the database, which needed SQL User Defined Functions to implement similarity searches and helped to avoid network and processing overhead. Experimental performance has shown that RDS can have competitive performance with small to medium size data sets, low query latency, high Precision@10, and affordable storage, confirming its feasibility as a practical vector store in organizations that do not necessarily regard high-performance as their operational efficiency.

Moreover, the paper has shown the possibilities of hybrid design combining the RDS with specific vector databases. These hybrid systems will also be able to take advantage of both methods: RDS offers a solid and affordable tier of embedding/ metadata storage, whereas FAISS or Milvus could be applied selectively to similarity search involving high throughput or large datasets. This architecture allows making a trade-off between operational convenience, cost and performance balanced, and due to this, the deployment of embedding-based AI applications in the production environments is not accompanied by huge infrastructure overheads. Future directions include the scaling of RDS based solutions of billion scale embeddings, examination of GPU acceleration, distributed RDS architecture and partitioning schemes to decrease latency and enhance throughput. Also, the new and more advanced indexing method, combined query optimization, and the integration with the generative AI model in real-time can be studied to improve the responsiveness and accuracy in the future. On the whole, this paper shows that Amazon RDS, with optimized preprocessing, schema, and hybrid implementations, is a feasible, scalable, and cost-efficient option to storing data and retrieving it in generative AI systems by expanding the range of application of relational databases in current AI processes.

References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers) (pp. 4171-4186).

[2] Goldberg, Y. (2017). Neural network methods in natural language processing. Morgan & Claypool Publishers.
[3] Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), 535-547.
[4] Vargas, M., Cannon, R., Engel, A., Sarwate, A. D., & Chiang, T. (2024). Understanding generative AI content with embedding models. arXiv preprint arXiv:2408.10437.
[5] Singh, P. N., Talasila, S., & Banakar, S. V. (2023, December). Analyzing embedding models for embedding vectors in vector databases. In 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1-7). IEEE.
[6] Sarferaz, S. (2024). Embedding Generative AI. In Embedding Artificial Intelligence into ERP Software: A Conceptual View on Business AI with Examples from SAP S/4HANA (pp. 277-288). Cham: Springer Nature Switzerland.
[7] Kukreja, S., Kumar, T., Bharate, V., Purohit, A., Dasgupta, A., & Guha, D. (2023, December). Vector databases and vector embeddings-review. In 2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP) (pp. 231-236). IEEE.
[8] Xian, J., Teofili, T., Pradeep, R., & Lin, J. (2024, March). Vector search with OpenAI embeddings: Lucene is all you need. In Proceedings of the 17th ACM International Conference on Web Search and Data Mining (pp. 1090-1093).
[9] Xiao, M., Wang, D., Wu, M., Wang, P., Zhou, Y., & Fu, Y. (2023, December). Beyond discrete selection: Continuous embedding space optimization for generative feature selection. In 2023 IEEE International Conference on Data Mining (ICDM) (pp. 688-697). IEEE.
[10] Miech, A., Laptev, I., & Sivic, J. (2018). Learning a text-video embedding from incomplete and heterogeneous data. arXiv preprint arXiv:1804.02516.
[11] Li, Y., & Yang, T. (2017). Word embedding for understanding natural language: a survey. In Guide to big data applications (pp. 83-104). Cham: Springer International Publishing.
[12] Fregly, C., Barth, A., & Eigenbrode, S. (2023). Generative AI on AWS: Building context-aware multimodal reasoning applications. " O'Reilly Media, Inc.".
[13] Kun, P., Freiberger, M. A., Løvlie, A. S., & Risi, S. (2024, July). GenFrame–Embedding Generative AI Into Interactive Artifacts. In Proceedings of the 2024 ACM Designing Interactive Systems Conference (pp. 714-727).
[14] Asperti, A., Evangelista, D., Marro, S., & Merizzi, F. (2023). Image embedding for denoising generative models. Artificial Intelligence Review, 56(12), 14511-14533.
[15] Zhang, Z., & Li, J. (2023). A review of artificial intelligence in embedded systems. Micromachines, 14(5), 897.
[16] Challa, N., Devineni, S. K., & Karangara, R. (2022). A deep dive into amazon web services: Unlocking the potential. Journal of Artificial Intelligence & Cloud Computing, 1, 2-5.
[17] Hu, H., Chen, Q., & Liu, Z. (2019, December). Code generation from supervised code embeddings. In International Conference on Neural Information Processing (pp. 388-396). Cham: Springer International Publishing.
[18] Ye, X., Shen, H., Ma, X., Bunescu, R., & Liu, C. (2016, May). From word embeddings to document similarities for improved information retrieval in software engineering. In Proceedings of the 38th international conference on software engineering (pp. 404-415).
[19] Zamani, H., & Croft, W. B. (2016, September). Embedding-based query language models. In Proceedings of the 2016 ACM international conference on the theory of information retrieval (pp. 147-156).
[20] Liu, Y., Wang, H., Zhou, K., Li, C., & Wu, R. (2022). A survey on AI for storage. CCF Transactions on High Performance Computing, 4(3), 233-264.
[21] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106
[22] Rusum, G. P., Pappula, K. K., & Anasuri, S. (2020). Constraint Solving at Scale: Optimizing Performance in Complex Parametric Assemblies. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(2), 47-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I2P106
[23] Pappula, K. K., & Anasuri, S. (2020). A Domain-Specific Language for Automating Feature-Based Part Creation in Parametric CAD. International Journal of Emerging Research in Engineering and Technology, 1(3), 35-44. https://doi.org/10.63282/3050-922X.IJERET-V1I3P105
[24] Enjam, G. R. (2020). Ransomware Resilience and Recovery Planning for Insurance Infrastructure. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 29-37. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P104
[25] Pappula, K. K., Anasuri, S., & Rusum, G. P. (2021). Building Observability into Full-Stack Systems: Metrics That Matter. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 48-58. https://doi.org/10.63282/3050-922X.IJERET-V2I4P106
[26] Pedda Muntala, P. S. R., & Karri, N. (2021). Leveraging Oracle Fusion ERP's Embedded AI for Predictive Financial Forecasting. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(3), 74-82. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I3P108
[27] Rahul, N. (2021). Strengthening Fraud Prevention with AI in P&C Insurance: Enhancing Cyber Resilience. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 43-53. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P106
[28] Enjam, G. R. (2021). Data Privacy & Encryption Practices in Cloud-Based Guidewire Deployments. *International Journal of AI, BigData, Computational and Management Studies*, *2*(3), 64-73. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I3P108
[29] Rusum, G. P. (2022). WebAssembly across Platforms: Running Native Apps in the Browser, Cloud, and Edge. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(1), 107-115. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P112
[30] Pappula, K. K. (2022). Architectural Evolution: Transitioning from Monoliths to Service-Oriented Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 53-62. https://doi.org/10.63282/3050-922X.IJERET-V3I4P107

[31] Jangam, S. K. (2022). Self-Healing Autonomous Software Code Development. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 42-52. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P105

[32] Anasuri, S. (2022). Adversarial Attacks and Defenses in Deep Neural Networks. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 77-85. https://doi.org/10.63282/xs971f03

[33] Pedda Muntala, P. S. R. (2022). Anomaly Detection in Expense Management using Oracle AI Services. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 87-94. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P109

[34] Rahul, N. (2022). Automating Claims, Policy, and Billing with AI in Guidewire: Streamlining Insurance Operations. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 75-83. https://doi.org/10.63282/3050-922X.IJERET-V3I4P109

[35] Enjam, G. R. (2022). Energy-Efficient Load Balancing in Distributed Insurance Systems Using AI-Optimized Switching Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 68-76. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P108

[36] Tekale, K. M., & Rahul, N. (2022). AI and Predictive Analytics in Underwriting, 2022 Advancements in Machine Learning for Loss Prediction and Customer Segmentation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 95-113. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P111

[37] Rusum, G. P., & Anasuri, S. (2023). Composable Enterprise Architecture: A New Paradigm for Modular Software Design. *International Journal of Emerging Research in Engineering and Technology*, *4*(1), 99-111. https://doi.org/10.63282/3050-922X.IJERET-V4I1P111

[38] Pappula, K. K. (2023). Reinforcement Learning for Intelligent Batching in Production Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 76-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P109

[39] Jangam, S. K., & Pedda Muntala, P. S. R. (2023). Challenges and Solutions for Managing Errors in Distributed Batch Processing Systems and Data Pipelines. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 65-79. https://doi.org/10.63282/3050-922X.IJERET-V4I4P107

[40] Anasuri, S. (2023). Secure Software Supply Chains in Open-Source Ecosystems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 62-74. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P108

[41] Pedda Muntala, P. S. R., & Karri, N. (2023). Leveraging Oracle Digital Assistant (ODA) to Automate ERP Transactions and Improve User Productivity. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 97-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P111

[42] Rahul, N. (2023). Transforming Underwriting with AI: Evolving Risk Assessment and Policy Pricing in P&C Insurance. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 92-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P110

[43] Enjam, G. R. (2023). Modernizing Legacy Insurance Systems with Microservices on Guidewire Cloud Platform. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 90-100. https://doi.org/10.63282/3050-922X.IJERET-V4I4P109

[44] Tekale, K. M., Enjam, G. R., & Rahul, N. (2023). AI Risk Coverage: Designing New Products to Cover Liability from AI Model Failures or Biased Algorithmic Decisions. International Journal of AI, BigData, Computational and Management Studies, 4(1), 137-146. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I1P114

[45] Rusum, G. P., & Pappula, K. K. (2024). Platform Engineering: Empowering Developers with Internal Developer Platforms (IDPs). International Journal of AI, BigData, Computational and Management Studies, 5(1), 89-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P110

[46] Gowtham Reddy Enjam, Sandeep Channapura Chandragowda, "Decentralized Insured Identity Verification in Cloud Platform using Blockchain-Backed Digital IDs and Biometric Fusion" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 2, pp. 75-86, 2024.

[47] Pappula, K. K., & Anasuri, S. (2024). Deep Learning for Industrial Barcode Recognition at High Throughput. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *5*(1), 79-91. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P108

[48] Rahul, N. (2024). Improving Policy Integrity with AI: Detecting Fraud in Policy Issuance and Claims. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P111

[49] Reddy Pedda Muntala , P. S. (2024). The Future of Self-Healing ERP Systems: AI-Driven Root Cause Analysis and Remediation. International Journal of AI, BigData, Computational and Management Studies, 5(2), 102-116. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P111

[50] Jangam, S. K., & Karri, N. (2024). Hyper Automation, a Combination of AI, ML, and Robotic Process Automation (RPA), to Achieve End-to-End Automation in Enterprise Workflows. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 92-103. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P109

[51] Anasuri, S., & Pappula, K. K. (2024). Human-AI Co-Creation Systems in Design and Art. International Journal of AI, BigData, Computational and Management Studies, 5(1), 102-113. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P111

[52] Tekale, K. M. (2024). AI Governance in Underwriting and Claims: Responding to 2024 Regulations on Generative AI, Bias Detection, and Explainability in Insurance Decisioning. International Journal of AI, BigData, Computational and Management Studies, 5(1), 159-166. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P116

[53] Pappula, K. K. (2020). Browser-Based Parametric Modeling: Bridging Web Technologies with CAD Kernels. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 56-67. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P107

[54] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105

[55] Enjam, G. R., & Chandragowda, S. C. (2020). Role-Based Access and Encryption in Multi-Tenant Insurance Architectures. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(4), 58-66. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P107

[56] Pappula, K. K. (2021). Modern CI/CD in Full-Stack Environments: Lessons from Source Control Migrations. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 51-59. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I4P106
[57] Pedda Muntala, P. S. R. (2021). Prescriptive AI in Procurement: Using Oracle AI to Recommend Optimal Supplier Decisions. *International Journal of AI, BigData, Computational and Management Studies*, *2*(1), 76-87. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I1P108
[58] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107
[59] Enjam, G. R., Chandragowda, S. C., & Tekale, K. M. (2021). Loss Ratio Optimization using Data-Driven Portfolio Segmentation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 54-62. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P107
[60] Rusum, G. P., & Pappula, K. K. (2022). Federated Learning in Practice: Building Collaborative Models While Preserving Privacy. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 79-88. https://doi.org/10.63282/3050-922X.IJERET-V3I2P109
[61] Pappula, K. K. (2022). Modular Monoliths in Practice: A Middle Ground for Growing Product Teams. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 53-63. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106
[62] Jangam, S. K., & Pedda Muntala, P. S. R. (2022). Role of Artificial Intelligence and Machine Learning in IoT Device Security. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 77-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P108
[63] Anasuri, S. (2022). Next-Gen DNS and Security Challenges in IoT Ecosystems. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 89-98. https://doi.org/10.63282/3050-922X.IJERET-V3I2P110
[64] Pedda Muntala, P. S. R. (2022). Detecting and Preventing Fraud in Oracle Cloud ERP Financials with Machine Learning. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 57-67. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P107
[65] Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
[66] Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110
[67] Tekale, K. M. (2022). Claims Optimization in a High-Inflation Environment Provide Frameworks for Leveraging Automation and Predictive Analytics to Reduce Claims Leakage and Accelerate Settlements. International Journal of Emerging Research in Engineering and Technology, 3(2), 110-122. https://doi.org/10.63282/3050-922X.IJERET-V3I2P112
[68] Rusum, G. P., & Pappula, K. K. (2023). Low-Code and No-Code Evolution: Empowering Domain Experts with Declarative AI Interfaces. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(2), 105-112. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I2P112
[69] Pappula, K. K., & Rusum, G. P. (2023). Multi-Modal AI for Structured Data Extraction from Documents. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 75-86. https://doi.org/10.63282/3050-922X.IJERET-V4I3P109
[70] Jangam, S. K., Karri, N., & Pedda Muntala, P. S. R. (2023). Develop and Adapt a Salesforce User Experience Design Strategy that Aligns with Business Objectives. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 53-61. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P107
[71] Anasuri, S. (2023). Confidential Computing Using Trusted Execution Environments. *International Journal of AI, BigData, Computational and Management Studies*, *4*(2), 97-110. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I2P111
[72] Pedda Muntala, P. S. R., & Jangam, S. K. (2023). Context-Aware AI Assistants in Oracle Fusion ERP for Real-Time Decision Support. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 75-84. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P109
[73] Rahul, N. (2023). Personalizing Policies with AI: Improving Customer Experience and Risk Assessment. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 85-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P110
[74] Enjam, G. R. (2023). AI Governance in Regulated Cloud-Native Insurance Platforms. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 102-111. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P111
[75] Tekale, K. M., & Enjam, G. reddy. (2023). Advanced Telematics & Connected-Car Data. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 124-132. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P114
[76] Guru Pramod Rusum, "Green ML: Designing Energy-Efficient Machine Learning Pipelines at Scale" *International Journal of Multidisciplinary on Science and Management*, Vol. 1, No. 2, pp. 49-61, 2024.
[77] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2024). Chatbot & Voice Bot Integration with Guidewire Digital Portals. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 82-93. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P109
[78] Kiran Kumar Pappula, "Transformer-Based Classification of Financial Documents in Hybrid Workflows" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 3, pp. 48-61, 2024.
[79] Rahul, N. (2024). Revolutionizing Medical Bill Reviews with AI: Enhancing Claims Processing Accuracy and Efficiency. International Journal of AI, BigData, Computational and Management Studies, 5(2), 128-140. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P113
[80] Pedda Muntala, P. S. R., & Karri, N. (2024). Evaluating the ROI of Embedded AI Capabilities in Oracle Fusion ERP. International Journal of AI, BigData, Computational and Management Studies, 5(1), 114-126. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P112
[81] Sandeep Kumar Jangam, Partha Sarathi Reddy Pedda Muntala, "Comprehensive Defense-in-Depth Strategy for Enterprise Application Security" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 3, pp. 62-75, 2024.

[82] Anasuri, S. (2024). Prompt Engineering Best Practices for Code Generation Tools. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 69-81. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P108

[83] Tekale, K. M., Rahul, N., & Enjam, G. reddy. (2024). EV Battery Liability & Product Recall Coverage: Insurance Solutions for the Rapidly Expanding Electric Vehicle Market. International Journal of AI, BigData, Computational and Management Studies, 5(2), 151-160. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P115