EDB
Postgres® for the AI Generation

Intelligent Data: **Unleashing AI with PostgreSQL**

www.enterprisedb.com

TABLE OF CONTENTS

Intelligent Data: Unleashing AI with PostgreSQL

Artificial intelligence (AI) is transforming the world and changing how information is captured, processed, stored, analyzed, and used. Understanding the terminology, the different types of AI, and how the technology works is critical for leveraging its strengths and capabilities for driving business objectives. In this white paper, we'll outline the AI landscape and how a Postgres data and AI platform can unlock the near-infinite potential of this revolutionary technology.

Introduction to AI

Artificial intelligence refers to the simulation of human intelligence processes by machines, especially computer systems. AI systems are designed to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation.

Without data, there is no AI. That's why, when we talk about the core of AI, we're often talking about databases and lakehouses, which we'll discuss in detail below. Achieving enterprise-level results for AI projects calls for storing your database in an enterprise data management environment such as Postgres, which provides a solid foundation for AI workloads.

What are the different types of AI?

AI systems can be categorized into three main types based on their capabilities: descriptive AI, predictive AI, and generative AI.

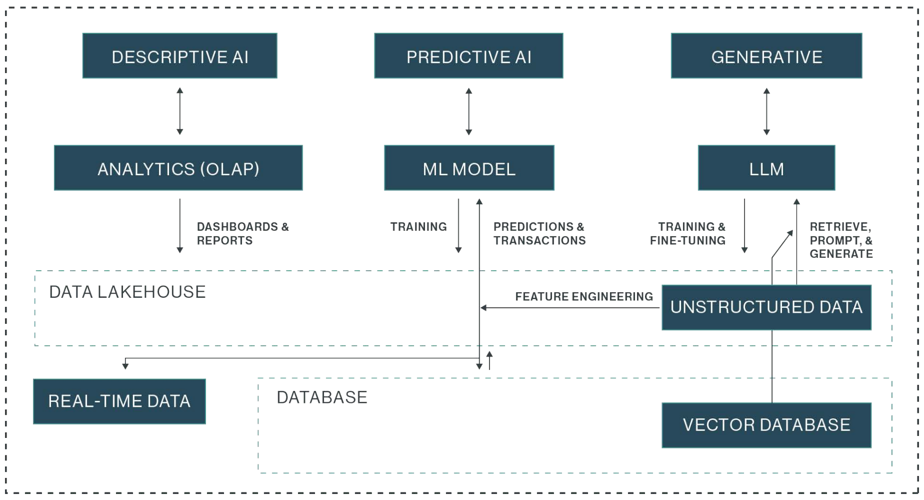
Descriptive AI, often referred to as business intelligence (BI), uses analytic technologies to run queries and perform online analytical processing (OLAP) on data to generate dashboards, reports, and more.

Predictive AI involves making predictions using machine learning (ML) models. To create these models, data from a data lakehouse is used for training. Once trained, the ML models are employed to make predictions in real time as new data arrives, whether it's a credit card transaction flagged for potential fraud or an incoming data stream that requires immediate analysis.

Generative AI also involves models, specifically large language models (LLMs) or foundation models trained on large volumes of data.

While these forms of AI are different, all three come together on the database layer, where structured and unstructured data converge. Databases like EDB Postgres are the foundation of AI, the mechanism that accelerates and enables timely data retrieval so AI can do what it does best.

THE AI LANDSCAPE



The AI data landscape

Structured and unstructured data are both leveraged in AI processes. Both forms of data contain valuable insights that organizations want to unlock. But both types need to be handled differently.

Structured data can be easily stored, accessed, filtered, and analyzed. As its name indicates, this data has some structure, like being organized in a table. Structured data can be stored in a database, in different formats such as CSV, JSON, and more. It can also be found in a data lake and in real-time feeds such as Kafka messages, which involve sending items such as a telemetry message from IoT devices to a system.

Unstructured data is a much larger volume of data. According to multiple analyst estimates, up to 90% of the data generated today is unstructured.[1] This data includes PDFs, texts, Word documents, images, photos, data on the web, HTML data, video data such as Zoom recordings and messages, meeting transcripts, social media, and other data that's being posted around the clock.

Organizations use descriptive and predictive AI or analytics approaches to unlock insights from structured data. But unstructured data such as text, images, and audio can't be directly processed by traditional analytics methods.

This is where generative AI plays a crucial role. Generative AI uses technology such as LLMs to work directly with unstructured data to train models and generate data, images, text, and more. Generative AI also enables organizations to extract structured features and insights from unstructured data – a process known as feature engineering.

By using LLMs, generative AI can do things like identify sentiments, detect profanity, and extract other structured information from unstructured sources such as documents or social media. This ability to transform unstructured data into structured features enables descriptive and predictive AI and analytics to accomplish more and operate on a wider range of data sources.

Ultimately, in order to unlock the insights and potential of your entire data landscape, including your unstructured data, an integrated approach combining generative AI, descriptive analytics, and predictive modeling is essential. By seamlessly integrating these three AI disciplines, you can unlock the true value that is within your diverse data assets, regardless of their structure or format.

Revolutionizing real-time analytics and AI workloads with Postgres

The journey toward achieving seamless real-time analytics is fraught with hurdles, primarily due to the fragmented nature of data across various systems and platforms. Robust, scalable, and versatile database solutions are needed for successful real-time analytics and AI workloads, and that's why Postgres is emerging as a database of choice.

Known for its reliability and rich feature set that caters to complex analytical needs and supports advanced indexing, partitioning, query optimization techniques, and seamless data integration with innovative tools such as Kafka, FDW, logical replication, and more, Postgres provides a solid foundation for building real-time analytical platforms.

By combining Postgres' advanced analytical features with pgvector's vector data capabilities, organizations can implement machine learning models directly within their database, facilitating seamless storing and querying of data, overcoming limitations posed by data silos, and enabling a unified, data-driven approach to decision-making.

Postgres' versatility extends to its language support, making it ideal for those who work with a variety of programming languages, including Python, RUST, and R. This flexibility ensures that AI models, regardless of the language they are written in, can be easily integrated and executed within the Postgres ecosystem. Data scientists and AI/ML programmers can quickly build their ML/LLM model with Postgres' advanced features and functionality as a database store.

The importance of real-time analytics and AI in driving business success cannot be overstated. With its continuous innovation and community-driven enhancements, EDB and Postgres stand ready to support organizations in their journey towards a more integrated, intelligent, and responsive data landscape

[1] *https://mitsloan.mit.edu/ideas-made-to-matter/tapping-power-unstructured-data*

What is a lakehouse?

While data has traditionally been stored in data lakes and warehouses, **data "lakehouses"** have become increasingly popular in recent years, as they are a cost-effective and efficient way for organizations to manage and analyze large amounts of data. The data in a lakehouse can come from operational sources (such as information stored in databases and enterprise resource planning (ERP) systems such as SAP). It can also originate in real time, and be directly ingested into a data lakehouse in a streaming fashion.

Is a data lakehouse the same thing as a data lake? They're similar. A data lake is a reservoir that can store both structured and unstructured data at any scale. The lakehouse expands on this concept, offering a metadata management layer that provides data management features and tools and data warehouse-like capabilities.

What's a data warehouse? A data warehouse is a deeply integrated and highly optimized way to store and analyze data that is relational in nature. While data warehouses generally offer a higher quality of services compared to data lakes, they're more expensive. The architecture also introduces additional complexity, such as extract, transform, and load processes to move and transform data from lakes into warehouses, which may cause latency in data availability.

There are two primary methods to integrate data into a data lake:

- **Extract, transform, and load (ETL)** – ETL processes involve extracting data from source systems, applying transformations as needed, and loading the prepared data into the data lake in batches. This approach is suitable for handling large volumes of operational data from systems of record.

- **Real-time ingestion by change data capture (CDC)** – Alternatively, CDC enables capturing data changes as they occur in real time, allowing for continuous ingestion of up-to-the-moment data into the lake. This real-time data can originate from various sources, such as message buses like Kafka, which may carry Internet of Things (IoT) sensor data or other streaming data feeds.

In the case of streaming data sources, the ingestion process involves continuously piping the data streams directly into the data lake, a process known as stream data ingestion. This real-time ingestion approach enables timely analysis of rapidly evolving data sets, complementing the batch-oriented ETL processes for operational data.

Once data is ingested into the lakehouse, analytics can be applied to extract valuable insights. **Analytics workflows typically include these steps:**

1. Data discovery to understand data quality, semantics, structure, and more
2. Data preparation, including cleansing, filtering, normalization, and deduplication
3. Optimization by transforming data into formats suitable for efficient analytics, such as columnar storage or analytical table structures
4. Batch analytics execution, running queries or models on the optimized data
5. Automated reporting by packaging analytics results into scheduled reports for consumers

While data lakes excel at cost-effective storage of massive data volumes, they don't always provide the capabilities needed for certain advanced analytics use cases requiring low latency, interactive queries for real-time dashboards, transactional consistency guarantees for continuous data ingestion, or fine-grained access controls for data security and compliance.

Data lakehouses solve these issues, combining all the benefits of data warehouses and data lakes into a single, open architecture. As a unified, one-stop solution, they merge the cost-effective, scalable storage capabilities of a data lake with the advanced analytics features and quality of service typically associated with data warehouses.

Key benefits of lakehouses

- Cost-effective and scalable data storage like a data lake
- Transactional consistency, low-latency queries, and fine-grained access controls like a data warehouse
- Support for diverse data modalities, including structured, semi-structured, and unstructured formats
- Interactive response times
- Elimination of separate ETL processes between lakes and warehouses
- Ingestion of data from other systems or from a real-time feed, enabling successful data discovery, preparation, and optimization

Data lakehouses are able to offer all these benefits by adopting a set of optimized open table formats and open source standards that enable high-performance analytics while retaining an open, data lake–like architecture. No wonder so many organizations are using lakehouses.

EDB Postgres AI offers a lakehouse that puts Postgres at the center of analytics workflows, with an eye toward future AI workflows. This new data lakehouse stack utilizes object storage, an open table format, and query accelerators to enable customers to query data through their standard Postgres interface, but in a highly scalable and performant manner.

What is generative AI?

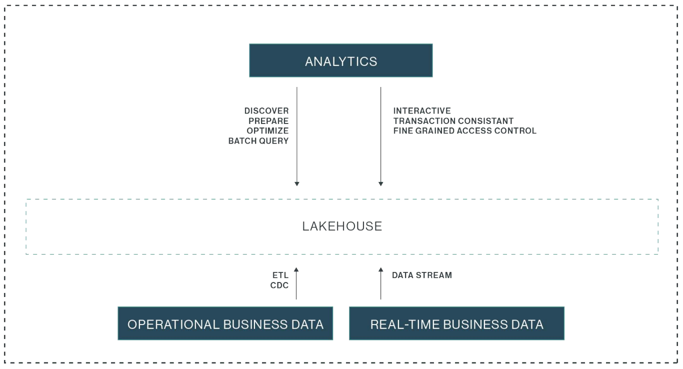
Today, there is a lot of buzz around **generative AI**, which is a form of deep learning. While deep learning isn't new, generative AI is unique because it combines multiple deep learning techniques to generate new data. The term that describes this best is **"transformer architecture,"** which refers to the ability to transform input data via various steps into output data.

Transformer architecture, powered by LLMs, is a driving force behind the generative AI industry. While there are multiple vendors offering commercial variants of LLMs, there is also an open ecosystem that has evolved around the Hugging Face platform where models and datasets are available for free.

Transformer architecture includes two phases:

- **Encoder Phase:** Input data (text, images, audio) is encoded into embeddings (an array of floating point numbers), capturing the semantic meaning of the input data.
- **Decoder Phase:** The embeddings are fed into a decoder that generates new output data based on the encoded semantics.

WHAT IS A LAKEHOUSE



A wide range of generative AI applications are powered by the encoder-decoder process:

- **Machine translation:** Encode text in one language into embeddings, decode into another language.
- **Summarization:** Encode lengthy text into embeddings, decode into a concise summary.
- **Classification:** Encoder-only models like classification models don't generate other unstructured data, but encode input into embeddings, and output a classification score (e.g., detecting profanity).
- **Language generation:** Decoder-only models such as GPT (Generative Pre-trained Transformer) generate new data and human-like text continuations.

To understand the generative AI landscape, it's crucial to recognize the role of embeddings in the transformer architecture. Embeddings act as the bridge connecting the encoder and decoder components. As numerical representations of input data that capture its essential features, they are the secret sauce that ties the entire generative AI process together.

Overall, the transformer architecture which combines encoders, decoders, and embeddings underpins generative AI, allowing input data to be transformed into novel generated outputs leveraging large AI models' capabilities.

The path to success with a vector database

Vector databases play a crucial role in AI, especially in generative AI and building AI applications. As AI becomes mission-critical, vector databases also become mission-critical. So it's important to understand what to look for when selecting a vector database.

Vector databases store **vector embeddings** (lists of numbers) representing the semantics of data, like documents or images, in a numeric space. The core function of a vector database is to enable vector searches to retrieve vectors that are semantically similar (numerically close) to a given query vector.

When working with high-dimensional data, especially in applications like recommendation engines, image search, and natural language processing, vector similarity search is a critical capability. Many AI applications involve finding similar items or recommendations based on user behavior or content similarity. The powerful Postgres pgvector extension is designed to perform vector similarity searches quickly and easily, making it the ideal choice for recommendation systems, content-based filtering, and similarity-based AI tasks.

Since vector searches often occur in interactive AI applications like chatbots or copilots, vector databases must perform fast, efficient vector searches to ensure responsive user experiences.

To achieve high-performance vector searches, vector databases rely on **vector indexes**. These indexes accelerate and streamline searches. However, the time it takes to build vector indexes must be taken into consideration.

Beyond performance in vector searches, several other factors are crucial when selecting a mission-critical vector database:

Accuracy: Vector indexes enable approximate nearest-neighbor searches, but the accuracy of results can vary based on the indexing technique employed.

Dynamic data handling: As data changes, new vector embeddings need to be generated and ingested; the database must support real-time ingestion and transactional updates to vector data and indexes.

Scalability: Mission-critical vector databases must be able to handle increasing volumes of vector data, searches, and concurrent workloads as the AI application grows.

Security: Fine-grained access control mechanisms can ensure only authorized data is retrieved for specific users or applications.

High availability: Robust backup, restore, and failover capabilities are necessary to maintain uninterrupted service for mission-critical AI applications.

Hybrid search: The ability to combine vector similarity searches with structured data filters or operations enables more complex queries that leverage both vector and tabular data.

A mission-critical vector database must address all of these factors to support demanding AI workloads in an AI environment while ensuring data security, availability, and the flexibility to integrate with existing data sources and structured information.

Because Postgres supports vectors, developers and engineers no longer have to deal with complex data transfer methods in and out of Postgres. The benefit of having vector data in Postgres is that you can still use your domain knowledge and use vectors to enhance your search experience.

Customizing generative AI

Public data, referring to the vast amount of information available on the internet, is used to train LLMs. Before this training can occur, however, the data needs to be prepared through processes like cleaning, formatting, and preprocessing.

Because training these massive language models is extremely costly and computationally intensive, many commercial vendors offer generic pre-trained models to reduce the barriers to entry. There are also open initiatives and ecosystems, such as Hugging Face, that provide services at a low cost or for free.

Customization refers to tailoring a generative AI model to your own private or domain-specific data not available in public training datasets used by commercial vendors. Just like public data, if you have proprietary data that is unique to your organization or industry, it needs to be prepared first before it can be incorporated into a generative AI solution.

Once your data has been prepared, there are two fundamental paths for customizing it:

Fine-tuning approach:

- An existing generic LLM is fine-tuned (further trained) on the prepared private data.
- This produces a custom LLM that can be prompted for custom outputs.
- Retraining is required for new data, which is time-consuming and computationally expensive.

Retrieval-augmented generation (RAG) approach:

- Prepared private data is converted to vector embeddings and stored in a vector store.
- During prompting, relevant embeddings are retrieved from the vector store and used to augment the prompt to the generic LLM.
- This allows incorporating new data in real time without retraining the LLM.

These two approaches can be combined, with fine-tuning for a base custom model and RAG for incorporating the latest data. However, due to RAG's flexibility, it is not surprising that it has become the dominant approach for generative AI solutions. We will discuss RAG in more detail below.

Understanding retrieval-augmented generation

RAG is a technique used in generative AI to enhance model performance and outputs. It involves retrieving and utilizing an organization's private data to augment the generative AI model's knowledge.

The process begins with collecting private data, which can include unstructured data such as images, videos, documents, PDFs, and other binary files. Once collected, this data is prepared to ensure its usability within the generative AI application workflow.

Data preparation may involve filtering out unwanted or low-quality data (e.g., profanity, noise), cleansing the data to remove irrelevant information, and condensing large volumes of data into more digestible fragments by summarizing it. If the data consists of lengthy documents, it may be chunked into smaller, self-contained nuggets of information to make it more manageable.

After preparing the private data, it is stored for further processing. Vector embeddings are then computed for these prepared documents. This process converts each data chunk into a numerical vector representation, capturing its semantic meaning as arrays of floating-point numbers.

Once computed, these vector embeddings are stored in a database or data store. Merely storing them is not enough, however. To enable rapid access, vector indexing techniques are employed to create an index specifically designed for efficiently searching and retrieving relevant vector embeddings.

Here's how generative AI applications consume and interact with this data:

1. The application sends a text prompt (query) for data generation.
2. The prompt is encoded into a vector embedding, representing its semantics.
3. This query embedding is used to perform a similarity search in the vector store (database) containing the application's private data embeddings.
4. The similarity search, accelerated by vector indexes, returns the most relevant data embeddings matching the query.
5. The actual data (documents, images, etc.) corresponding to the retrieved embeddings is fetched from the store.
6. The retrieved data is used to augment the original prompt.
7. The augmented prompt is sent to an LLM to generate the final output, which is returned to the application.

This RAG process has led to the development of integrated systems like AI databases and vector databases which store, index, and enable similarity searches on vector embeddings.

AI databases expand on vector databases by storing the actual data (documents, images, etc.), computing embeddings, and automating the entire RAG process for applications. An AI data platform encapsulates all these capabilities into a unified solution, allowing developers to build generative AI applications by treating the entire RAG workflow as a database workload.

This integration of RAG capabilities into database systems highlights the critical role databases play in enabling efficient and scalable generative AI solutions.

From vector DB to AI data platform

From capturing the data to fine-tuning and storing it, there are a lot of steps in creating generative AI applications.

Building a generative AI application requires:

- Data capture and preparation, often using AI models for tasks such as classification and hate speech detection
- Storing prepared data securely and efficiently while enabling retrieval for the application
- Computing and indexing vector embeddings of the data for efficient vector search
- Implementing RAG by integrating with LLMs to generate data
- Fine-tuning LLMs in some cases
- Managing context for language model prompts, including instructions, conversational history, and retrieved data

Accelerating AI development

The capabilities for AI data management have evolved to streamline the work involved in building intelligent apps and extracting valuable insights. These processes rely on databases and platforms such as:

Vector database: Vector databases are table stakes for storing and indexing vector embeddings, enabling vector search and retrieval.

AI database: These more advanced databases store the actual AI data, not just vector embeddings, allowing retrieval of the data itself.

AI data platform: This comprehensive solution handles all aspects of building generative AI applications out of the box, including data preparation, vector embedding, RAG, language model integration and fine-tuning, and context management.

Typically, developers who build generative AI applications require support from experts such as data scientists who understand generative AI models and and data engineers adept in preparing and wrangling data. By taking on some of this work, AI databases and data platforms empower developers to build generative AI solutions more independently and with shorter time-to-market.

Harnessing the power of Postgres for AI workloads

The object-relational database system Postgres is uniquely suited for AI tasks. Its ease of use, open source architecture, and extensibility enable it to work with new AI innovations that extend its functionality. For example, the popular pgvector extension enables you to efficiently store your vectors and quickly perform similarity searches. Features such as Common Table Expressions (CTEs) and Window Functions allow analysts and developers to craft complex queries that efficiently process large volumes of data, deriving actionable insights in real-time. There are also tools such as EvaDB that connect to your relational database and perform SQL queries on pre-trained models such as Hugging Face, OpenAI, YOLO, and PyTorch.

At EDB, we are actively contributing to the evolution of Postgres, so companies worldwide can realize the full potential of their data and achieve true innovation. We are building a Postgres data and AI platform that delivers on the full workloads needed for this new generation of intelligent applications.

Today, it's no longer just about database functionality and transactional processing. Radical transformation lies within the systems that understand the data, know how to process it, and anticipate the use of that data. Helping organizations move from traditional application development to AI application is our vision for the next phase of Postgres. We invite you to explore our solutions and reach out to discuss how you can be a part of the AI transformation.

Watch EDB's chief architect for analytics and AI walk through the new data and AI landscape in our lightboard video series.

Watch Now »

EDB

EDB provides a data and AI platform that enables organizations to harness the full power of Postgres for transactional, analytical, and AI workloads across any cloud, any time.

For more information, visit www.enterprisedb.com.