Google Cloud

Unlocking AI's full potential with operational databases

Your guide to AI application and agent development.

Table of contents

Chapter 1

Using databases to build better AI apps and agents

Everyone is building with AI. It can improve search. It can personalize customer interactions. It can increase your developer productivity. It can even write bedtime stories.

But as a developer, you know the reality behind the hype. There's a chasm between a simple AI demo and a production-ready enterprise application that users can trust.

In the AI era, developers are poised to become central to enterprise innovation, not just as coders but as major stakeholders. By connecting powerful models to reliable data, they're uniquely positioned to produce key insights that drive business strategy and ensure customer expectations are met. However, this shift comes with its own set of challenges.

That's because general-purpose AI models are limited to answering general questions. But most apps need to answer specific questions (like "What's my account balance?" or "Do you have a size 8 in stock?"), and that requires integrating business-specific operational information. **Your databases can bridge this gap.**

Operational databases help you create apps and agents that are accurate in real time, contextual to the user's experience, and reliable at enterprise scale.

We'll show you how to get started in this guide.

Looking to get started right away?

Discover the six pillars for successful AI apps

Go to page 5

Level up your users' experience

Today's users expect immediate and accurate answers to their questions. So your app is under pressure to deliver.

Most of the information users are looking for sits in your operational databases—such as your CRM, ERP, and ecommerce data.

Questions like:

When will my shipment arrive?

My order arrived damaged. Can I have a replacement?

Do you have a black size 7 in stock at a store near me?

General-purpose AI models can't answer questions like these on their own—they need the information contained in your operational databases. To meet user expectations, integration of operational data into your app has moved from a "nice to have" to "complete by the end of the next sprint."

Foundation models

A foundation model is typically a large language model (LLM) that's trained on a massive amount of general data. It can be used to generate text and other content, as well as perform other natural language processing (NLP) tasks. Many foundation models can understand and operate across different types of formats, such as images, audio, and video.

As foundation models are trained on generalized data, they can easily answer general questions such as "What's the capital of France?". But their knowledge is limited to the data they were trained on. We'll look at methods to bridge that gap by integrating foundation models with your company's operational databases, providing them with the real-time, proprietary information they need.

"Databases are the fuel for the agentic era. Operational databases bridge the gap between foundation models and enterprise applications to help deliver contextual, relevant, and accurate user experiences."

Andi Gutmans
VP and GM, Data Cloud, Google Cloud

The tide is turning to developers

AI is tackling enterprise challenges, and it's developers leading the charge.

Uber is using AI agents to streamline customer service, helping users find solutions to their problems quickly. Deloitte has a "Care Finder" agent, built with Google Cloud, that helps care seekers find in-network providers—often in less than a minute.

When enterprises achieve these domain-specific, intelligent interactions, developers take a starring role. While data scientists are essential for creating the complex models that power AI, their expertise is a scarce resource. As a result, companies are relying on software developers to bridge the gap. This shift places developers at the center of the action, making them the critical engine for integrating AI models into domain-specific, intelligent applications.

"I believe we are entering a 'post-training era' in which application developers will drive the bulk of the innovation in applying AI to solve business problems."

Andi Gutmans
VP and GM, Data Cloud, Google Cloud

How are they innovating?

They're employing techniques like retrieval augmented generation to make foundation models dependable for enterprise applications. They're using open-source protocols to connect AI agents and apps to their systems and frameworks more efficiently. And they're using embeddings for proximity searches, particularly in vector-enabled databases.

Not sure how to fully use AI to your advantage as a developer? Don't worry. You will by the time you finish reading this paper.

The new era will be led by developers who build deep proficiency in how to best leverage and integrate AI technologies into applications.

Let's make sure that's you.

AI is already here

Today, AI isn't just a "nice to have," it's a fundamental part of everyday apps—from displaying real-time product availability in your online store to automatically adjusting prices to match competitors. Even when the user might not realize it, AI is working in the background to personalize their experience based on past behavior and to protect their data from potential threats.

AI models have come a long way, but it can still be challenging to return relevant and accurate results 100% of the time.

Notoriously, models have an occasional tendency to "hallucinate"—terminology for when an AI model generates a factually incorrect or even nonsensical answer. Since AI models are non-deterministic, there is always a non-zero probability they will produce an incorrect answer. As important as the quality of the input data for training and predictions is, AI agents and database integration also can make a significant difference.

The key need here is reliable, business-specific data. By ensuring that AI outputs are consistently aligned with the business's true context, you can build more reliable and trustworthy AI applications that deliver precise and relevant insights.

The potential for implementing AI models and agents across your tech stack is extraordinary. But the stakes for you as a developer skyrocket accordingly. An agent with the power to take action across your system has an inherently larger blast radius than a simple chatbot that just answers questions. **An incorrect fact is a problem, but an incorrect action in a production system is a crisis.** That's why it's essential to architect for trust, building robust safeguards at the core of your application to verify an agent's outputs—and its reasoning—before it acts.

Even with high stakes, the solution lies in familiar territory. The same principles you use for building any mission-critical application—rigorous validation and testing against a source of truth—are the keys to building safe AI. This is where your database becomes the cornerstone of your strategy. By providing verifiable, real-time data, it acts as the essential guardrail, allowing you to build intelligent, reliable, and fast applications with confidence.

Let's look at some guidelines for enterprise AI apps.

Chapter 2

Six pillars for successful AI apps

The most powerful enterprise AI apps augment foundation models by grounding them in operational data. This approach is the key to achieving the six pillars of a successful application: relevancy, reliability, observability, scalability, security, and cost-effectiveness.

An AI app that achieves these pillars will provide your users with the best experience, while minimizing risk to the business.

1 Relevancy

Deliver accurate and contextual information.

Grounding your model's responses in your operational database, which continuously stores and processes your data, means that your results will be more accurate, informative, and nuanced to the user's specific intent.

2 Reliability

Ensure your app delivers when you need it to.

To build users' trust in your organization, make sure they can rely on your app. Your AI setup should be highly available, resilient to failures, be simple to maintain without causing disruptions or downtime, and support disaster recovery.

3 Observability

Understand your agents' "why."

An AI agent can feel like a black box. Observability gives you the tools to look inside. It means tracing every step of an agent's decision-making process—from the initial prompt to the specific database queries it ran and the data it retrieved—to ensure reliability and build trust.

4 Scalability

Grow when you need to. Shrink, too.

Have the flexibility to scale up or down without large infrastructure changes. Look for a solution that gives you horizontal scalability (at least for read operations) and cross-region replication. Your app should be quick to deploy and scale, and operate with high throughput and low latency.

5 Security

Keep customer and company data secure.

You already ensure that your databases follow security best practices. Make sure that your AI tools do, too. Create an AI security policy and then ask about your providers' security standards. A strong defense is built from every layer having security built in.

6 Cost-effectiveness

Build smart and spend smart.

AI models, especially the most powerful ones, have significant operational costs, often tied to per-token pricing for API calls. A successful AI app must deliver value with a sustainable price tag, which requires making smart architectural choices.

A modern database gives you a solid foundation for pillars like security and scalability, but these principles aren't just features you can flip on. They have to be architected. Ultimately, it's up to you, the developer, to build on that database foundation, weaving all six principles into the core logic of your AI agents to create a truly enterprise-grade application.

Let's dig into some of the technologies that developers are using.

Chapter 3

The technologies powering AI apps

Users expect prompt, personalized interactions. But developers are finding that legacy databases aren't cutting it due to their lagging AI capabilities. Thankfully, modern databases have excellent and rapidly evolving AI support, and they can help you bring these new AI capabilities to users.

Here are a few of the technologies that modern enterprises are using to create AI apps that deliver more relevant and reliable results:

- Long context windows
- Specialized AI models
- Agent-to-agent communication
- Tools
- Vector embeddings
- Vector search
- Retrieval augmented generation (RAG)
- Orchestration frameworks

Let's have a look at each.

Long context windows

A context window is the amount of information a foundation model can consider while generating a response. So the longer the context window, the more background information the model can consider for the response.

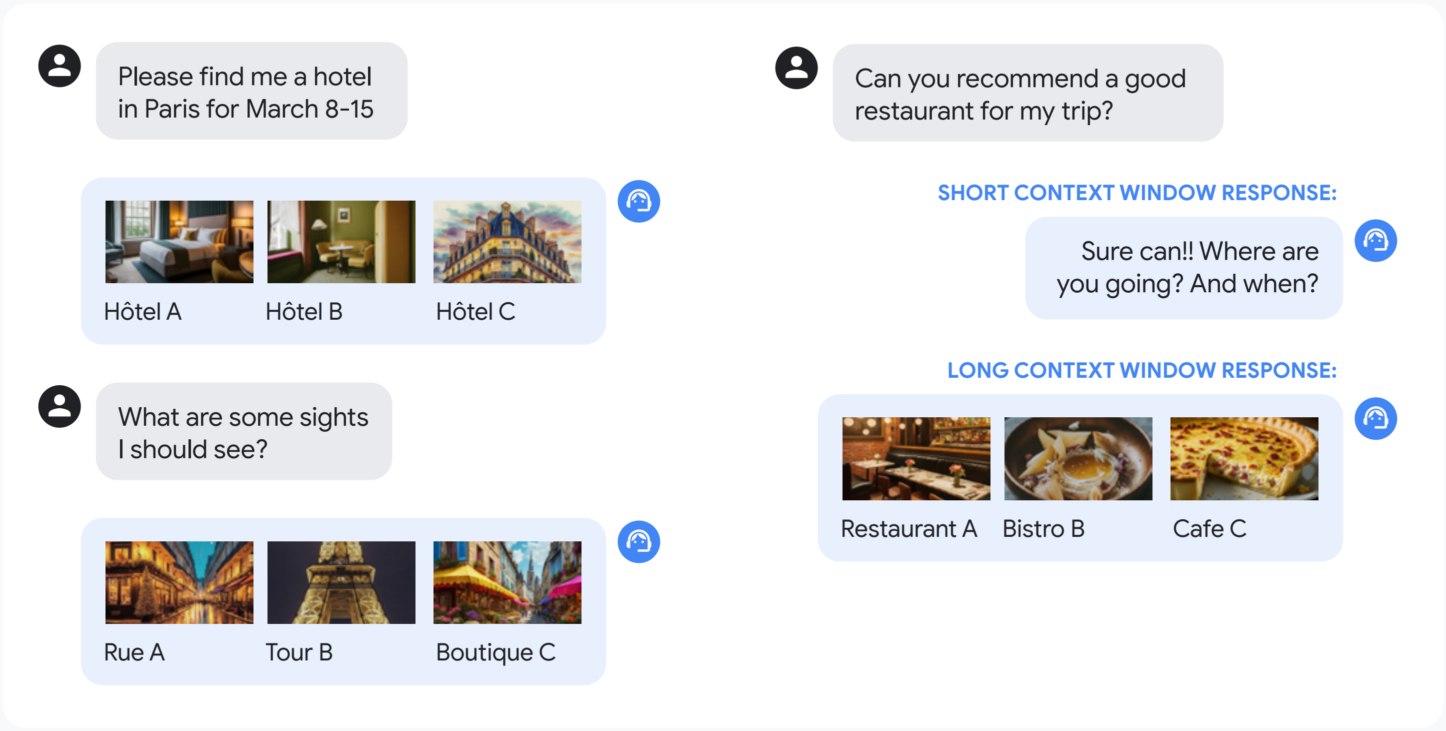
A “token” is the smallest unit of text that carries meaning for a language model. It’s often slightly smaller than the size of a word.

Let’s say a user is using a travel chatbot agent and includes the information that they’re heading to Paris in March. After a number of lines of dialog, they ask, “Can you recommend a good restaurant for my trip?”

The quality of the response will vary depending on the size of the context window. Any information outside of the model’s context window won’t be considered. So if the context window is too small, the model wouldn’t know where the trip is to. In this case, it may hallucinate or give an irrelevant response.

This is a contrived example, since a short chat would easily fit into a typical context window. But if the relevant information is contained in a previous dialogue, tens or hundreds of thousands of tokens ago, then the chatbot might not be able to give a good answer.

A long context window comes with tradeoffs. It’s slower than other methods of improving model accuracy and may add new issues such as positional bias, sensitivity to changes in information placement, difficulty maintaining global coherence or performing complex reasoning, and high computational cost. And model vendors typically charge per token, so a longer context window will raise your costs.


Please find me a hotel in Paris for March 8-15

What are some sights I should see?

Can you recommend a good restaurant for my trip?

SHORT CONTEXT WINDOW RESPONSE:

Sure can!! Where are you going? And when?

LONG CONTEXT WINDOW RESPONSE:

Specialized AI models

In some industries, the tide is shifting away from one-size-fits-all gargantuan models to specialized AI models. These specialized models are trained on a limited, but deeper amount of information—a process often referred to as 'fine-tuning'. For certain applications, this can enable apps and agents to answer more detailed questions.

As an example, imagine a medical practice that wants to create a specialized AI model that can help diagnose whether a mole is at risk of becoming cancerous. One way to approach this is to take an off-the-shelf open AI model, then tune that foundational model on a custom data set to build specialized knowledge.

In this example, the medical practice might train the model on a database of millions of past diagnoses, including written descriptions of symptoms and visual images. In doing so, they're training the model on the kind of information it needs to look for and the patterns to notice, such as asymmetrical shapes and abnormal discoloration. Now it can make a likely prediction based on millions of past cases—and articulate that prediction.

Agent-to-agent communication

AI technology has rapidly progressed beyond single-purpose models into an era of multi-agent systems. These systems involve multiple AI agents acting, learning, communicating, and collaborating autonomously—with each specializing in a particular aspect of a task. Think of them like a team of experts, where one agent is a skilled researcher, another is an eagle-eyed fact checker, and a third is a seasoned writer who can best synthesize the results into a concise report.

This collaborative approach enables systems to handle more intricate workloads and produce more reliable outputs—at faster speeds. For this kind of system to work, the agents need to be able to understand each other, share information, and synchronize their actions to avoid bottlenecks, errors, or duplication of work. That's where the newly developed Agent2Agent Protocol (A2A) comes into play, allowing AI agents to communicate, securely exchange information, and coordinate actions—even if they were built by different vendors.

Tools

For an AI agent to take meaningful action, it needs access to data. In AI terminology, tools are specific functions that provide data, for example, by querying a product database, calling an external API, or retrieving a customer's order history.

By giving agents access to a library of reliable tools, you enable them to interact with real-time data and execute multi-step tasks. The challenge is that for agents and tools to work together, especially in a multi-agent system, they need a common language.

This is where the Model Context Protocol (MCP) becomes critical. MCP is an open-source standard that acts like a universal adapter for AI. Instead of building custom, brittle connectors for each new data source, MCP provides a universal plug. This means a developer can build a tool (like a `get_order_status function`) once and make it available to any agent that understands the MCP standard, regardless of the framework it was built with.

To make these tools available, they need to be hosted on a server that acts as a secure gateway. This server exposes your tools to the agentic world via the MCP standard. It handles the secure connection to your database, executes the function on the agent's behalf, and returns the result in a format the agent can understand.

Vector embeddings

Vector embeddings convert text, images, or other content into vectors (numerical representations), enabling a foundation model to understand semantic similarities between words and phrases through a mathematical formula. In linear algebra, a vector represents a position in a virtual space, and if two objects are similar, then the vectors will be neighbors in that virtual space.

Once converted, vector search enables retrieval of the most relevant information by looking at the distance between the vectors—a short distance means two pieces of content are semantically similar. This means that matches can be meaningful, rather than focusing on surface-level, keyword-based similarity to the input question.

The foundation model defines your vector space, so the analysis of whether two vectors are near or far from another depends on the version of your specific model.

This need to compare semantic meaning is why recommendation engines, search algorithms, and many other natural language processing applications rely on vector embeddings. They use vector search results to ground foundation model responses, minimize hallucinations, and provide more reliable information.

Google Cloud databases include support for vectors, meaning you can streamline your embedding creation and access processes using a database you already know.

Working with vector embeddings

Databases like AlloyDB can generate and store billions of vector embeddings directly alongside your existing operational data, all within a single database. Learn how AlloyDB can:

- Generate and store vector embeddings based on a model
- Index and query embeddings using the pgvector extension
- Work with embeddings generated outside the database

Get started

The pgvector extension for PostgreSQL databases enables you to store, index, and query vectors—and run vector similarity searches.

Vector search

Vector search is the engine that powers semantic understanding in modern applications. If you search for music, movies, or books like the ones you already love—which are conceptually similar—that's where vector search comes in.

Instead of matching keywords, vector search queries the numerical representations of your data—the embeddings—to find the closest semantic (conceptual) matches. It's particularly useful when dealing with large datasets of unstructured data like images, text, or audio, where traditional search methods (based on exact matches) might not be effective.

Vector search plays a crucial role in generative AI applications because it can also be used to ground AI models via prompts or tools, increasing relevance and reducing hallucinations. Having this functionality built into a general-purpose database means you can search without the need to manage a specialized vector database or ETL pipeline.

An example search workflow

Generate embeddings with your data and pgvector functionality. This example takes plain-text input and fetches the nearest neighbors from a database, relying on model-driven semantic parsing of the text's meaning.

```
SELECT id, name FROM items

  ORDER BY complaint_embedding

  <-> embedding('gemini-embedding-001',
'wind and rain protecting red jacket
for women')::vector LIMIT 10;
```

Read more

Choosing a search approach

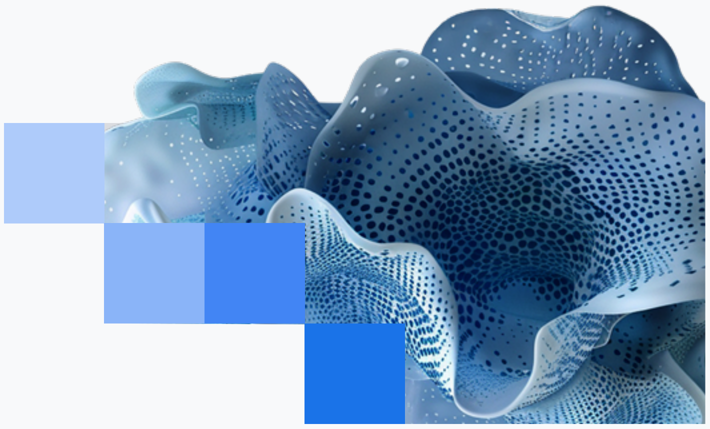
Deciding which search algorithm to use is a classic engineering trade-off between accuracy and performance. Your choice between an exact or an approximate algorithm will depend on the specific requirements of your application.

K-nearest neighbors (KNN):
The exact method

The brute-force approach guarantees perfect recall (accuracy) by comparing your query vector against every other vector in your dataset. While this provides the mathematically precise ground truth, the computational cost is immense. It is best suited for offline or academic workloads where absolute precision is the top priority.

Approximate nearest neighbor (ANN):
High-performance

This is the approach used by virtually all large-scale production systems. ANN algorithms use sophisticated indexing techniques to intelligently partition the vector space. Their ability to navigate to the most likely search region means they can find highly relevant neighbors with incredible speed, trading a small, often negligible, amount of precision for massive gains in performance and scalability.



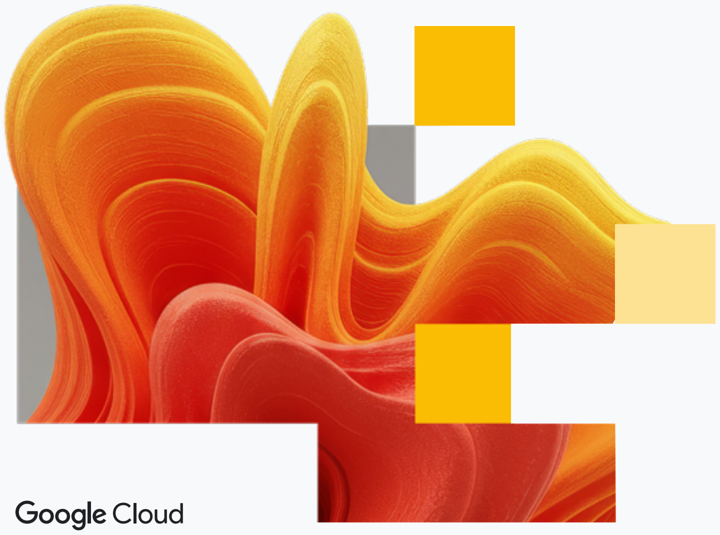
Retrieval augmented generation (RAG)

Retrieval augmented generation is an AI technique that combines the strengths of traditional information retrieval systems—such as databases—with the capabilities of AI models. It grounds your foundation model with fresh or domain-specific data from your database to power more accurate, up-to-date, and relevant responses.

This grounding can be approached in more than one way. In a classic RAG workflow, the application first retrieves relevant data and then **augments the prompt** by "stuffing" this context directly into the model's input. A more modern and flexible approach, however, is to provide the AI agent with **tools**. Instead of pre-fetching data, the developer gives the agent a function (e.g., `search_customer_orders`) that it can choose to call when it needs information. This tool-based method allows the agent to reason more effectively and handle more complex, multi-step queries.

Regardless of the method, RAG is highly efficient. By retrieving only the necessary data, it reduces the reliance on large context windows and, most importantly, provides the specific context needed to generate highly accurate and relevant responses.

Critically, RAG enables you to use the foundational model as-is, without asking the vendor to update it every time you update your database.



Meet AlloyDB

AlloyDB is Google Cloud's PostgreSQL-compatible database that offers superior performance, availability, and scale. It's optimized for enterprise AI apps that need real-time and accurate responses. It delivers superior performance for transactional, analytical, and vector workloads—and offers the same vector search algorithm (ScaNN) used by Google Search.

And with Gemini, you get AI-powered assistance through every stage of the database journey, from development and performance optimization to fleet management, governance, and migration.

AlloyDB can deliver up to 4 times faster vector queries and up to 10 times faster filtered vector search queries than the HNSW index in standard PostgreSQL.

AlloyDB AI helps developers more easily and efficiently combine the power of foundation models with real-time operational data by providing built-in, end-to-end support for vector embeddings, natural language processing, and other AI needs.

AlloyDB Omni is a downloadable edition of AlloyDB built with portability and flexibility in mind. Run enterprise-grade, AI-enabled applications anywhere: on premises, at the edge, across clouds, or even on your laptop.

Build AI apps with AlloyDB AI.

Get started

Let's see how RAG works in a common scenario: a chatbot agent for a retailer.

A customer wants a suggestion for a toy for a five-year-old child. Using only a foundation model, the chatbot could reply about general trends and age appropriateness, but could not align those suggestions with products the retailer actually sells.

By augmenting the standard foundation model with real-time inventory and product information from the retailer's operational database, the chatbot can use RAG to answer a wide range of questions about availability, pricing, and return policies. It can even provide a pointer to the store closest to the customer that has the recommended toy in stock. That's the type of personalized response that helps close a sale.

What's a prompt?

You're probably already familiar with the idea of a prompt: the input that you provide to a model to elicit a response. When using consumer-oriented AI, it's the question or statement that you type into the text field: "Gemini, what's the largest stadium in the world?"

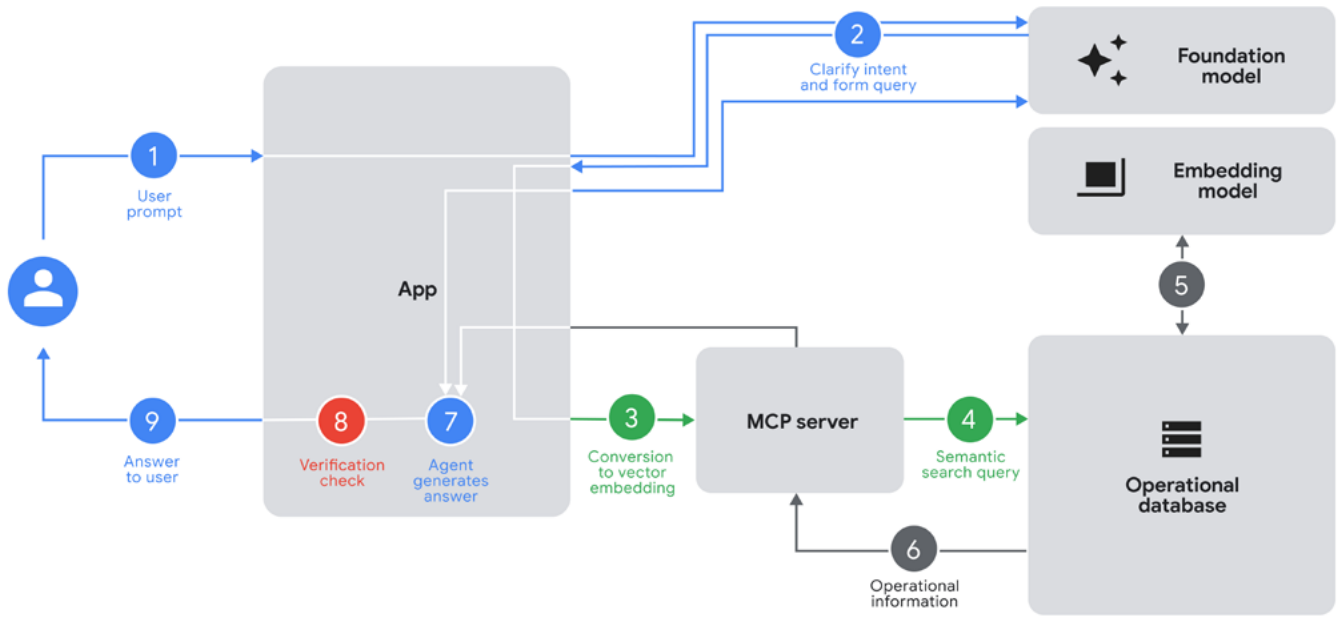
Inside AI apps, the prompt becomes much more powerful. It evolves in two key ways:

1. **Context-augmented prompts:** In a classic RAG system, the application first retrieves relevant data from a database, then augments the user's question with this data. The final prompt sent to the model includes both the original question and the retrieved context.

2. **Instructional agent prompts:** In modern agentic systems, the prompt evolves further into a mission briefing. Instead of just stuffing in raw data, the prompt provides instructions that tell an agent how to find the answer. This prompt might include the user's goal, a description of available tools (like a database query function or an API), and rules for how the agent should behave.

So, while prompts are changing, they are still critical. They're the primary mechanism through which developers guide and control the behavior of powerful AI agents.

How does RAG typically work?



1. The user enters a question (prompt) into the app
2. The application uses a foundation model to understand user intent and formulate a data query
3. The app calls the required tools, which are hosted on the MCP server
4. The MCP server executes the tools, which query the operational database
5. The database uses the embedding model and vector search to get results
6. Operational information is returned from the database
7. The agent receives the data from the tool/ database and synthesizes it with the original query and calls the foundation model to generate a final, fact-based answer
8. An application-specific verification check on the resultant output takes place, and the process is repeated if necessary to improve the answer
9. The app returns a response to the user

A verification step, often using the database as a source of truth, is a critical safeguard against model hallucinations and ensures enterprise-grade reliability.

The more enterprises can ground foundation models in their real-time information and enterprise data, the more accurate their apps become.

Orchestration frameworks

While a single call to a foundation model can be powerful, the frontier of AI development lies in building autonomous agents. An agent isn't just a simple prompt-and-response loop—it's a system that can plan, reason, and execute complex tasks like managing conversations, using tools, and handling API calls. Building these capabilities from scratch requires a lot of repetitive "plumbing" code.

This is where orchestration frameworks are key. They provide the operating system for your agents by managing intricate workflows and streamlining essential tasks. For example, a simple but critical task is maintaining conversation history. A foundation model is inherently stateless—it doesn't remember past interactions. Instead of manually coding the logic to store, retrieve, and append chat history to every prompt, an orchestration framework provides pre-built instructions to handle this automatically.

By providing this essential scaffolding, orchestration frameworks make it possible to build sophisticated, reliable, and scalable agents much more quickly, while keeping your code simple.

Some leading orchestration frameworks are LangChain, LlamaIndex, and Google's own Agent Development Kit.

Learn how to build with:

- LangChain
- LlamaIndex
- Agent Development Kit

Not all AI technologies are created equal

We've covered a number of different techniques for achieving more accurate responses from AI models. How do they stack up? Let's compare.

Consider an HR chatbot agent that employees can use to ask questions about policies. A foundation model could reference general HR information, but would be unable to reference particular company policies.

And it goes without saying that when an employee wants to know how much annual leave they have available, they want to know the exact number of hours in their leave balance, not a statement about legal requirements or a general average.

It's possible to input an employee's leave history into a **long context window,** but this method can be slow, prohibitively expensive, and may not produce the right result if the context window isn't long enough.

You can attempt to use **vector search** over the employee's leave history, but again, this may not produce accurate results because a semantically relevant answer isn't what the employee is interested in.

With **RAG** accessing the employee's documents and company policy, the agent can respond with the employee's exact leave balance. Guided by an orchestration framework, the agent would select the right tool—like a `get_leave_balance` function. This tool, exposed via a server using a standard like MCP, would securely query the HR database and return the exact number, allowing the agent to provide a precise and trustworthy answer.

Chapter 4

Putting these technologies to work

As an AI developer using techniques like RAG, there are a number of decisions you need to make. Each choice comes with tradeoffs that you'll want to consider depending on your particular use case.

Here are some questions to ask:

- **Which foundation model(s) should I use?** Different models have distinct strengths—for example, some have built a reputation for exceptional coding abilities and others for rigorous logical reasoning. Your choice should align with your application's primary function, and you can easily integrate multiple models, such as Gemini and others.
- **Which database should I use?** This decision will be driven not only by your AI needs, but by your database needs more broadly.
- **How should I retrieve information from the database: by creating custom APIs or using SQL queries?** Custom APIs can execute any database query, including structured SQL queries and vector search, on behalf of the model—and are accurate and secure. However, they're not flexible enough to handle the full range of end-user questions. SQL statements are fast but take skill to write, including the skill to make them secure.
- **Should I use an orchestration framework?** This depends on the complexity of what you're building. Typically, as soon as your app needs to function like an agent, using multi-step chains of reasoning, deciding between different tools (like calling an API vs. querying a database), or implementing a complex RAG workflow, a framework provides the essential scaffolding.
- **Do you need a specialized vector database?** Not necessarily. Many popular databases include support for vectors, and your existing database may allow you to easily add a vector column or index. This way, you don't need to learn and install a new database.

All Google Cloud databases offer low-latency vector search, so you get instantaneous, relevant responses no matter how large the dataset or how many parallel requests hit the system. This makes for a better user experience across applications like search, conversational AI, and personalized recommendations.

Although vector search is important, your app probably also needs to store non-vector data and be scalable, secure, and highly available. A specialized vector database may or may not be the right fit for your core considerations, depending on the requirements of your app.

Codelabs: Putting RAG to work

Even the best off-the-shelf RAG techniques require some learning. Below, we've included a couple of tutorials to help you get started implementing RAG in your application.

Google Developers Codelabs are guided tutorials that give you hands-on coding experience. In addition to the labs we highlight in this paper, you can go deeper and search the entire Codelabs library to find articles about the technologies you're looking at, such as RAG or AlloyDB.

Explore all AI & Machine Learning codelabs

Join the Google Developer Program

If you haven't already, create a Google developer profile and select topics like AI and databases so you're always up to date.

Colab: Use RAG and the Netflix movie dataset using LangChain

If you've decided that a framework is right for your application, try this codelab.

Learn how to create a powerful, interactive AI application using RAG powered by AlloyDB for PostgreSQL and LangChain. This application is grounded in a Netflix Movie dataset, enabling you to interact with movie data in exciting new ways.

You'll learn how to:

- Deploy an AlloyDB instance
- Use AlloyDB as a VectorStore
- Use AlloyDB as a DocumentLoader
- Use AlloyDB for ChatHistory storage

Get the notebook

```
# create the ConversationalRetrievalChain
rag_chain = ConversationalRetrievalChain.from_llm(
   lm=llm,
   retriever=retriever,
   verbose=False,
   memory=memory,
   condense_question_prompt=condense_question_
   prompt_passthrough,
   combine_docs_chain_kwargs={"prompt": prompt},
)
```

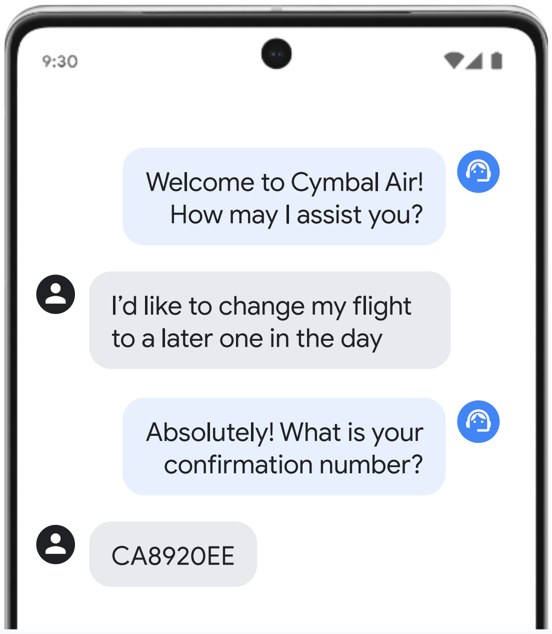
Codelab: AlloyDB Agentic RAG Application with MCP Toolbox

Learn how to create an AlloyDB cluster, deploy the MCP toolbox, and build a RAG-based chat application that uses the deployed toolbox to ground its requests.

You'll learn how to:

- Deploy an AlloyDB Cluster
- Connect to AlloyDB
- Configure and deploy the MCP Toolbox Service
- Deploy a sample application using the deployed service

Start the codelab



Welcome to Cymbal Air! How may I assist you?

I'd like to change my flight to a later one in the day

Absolutely! What is your confirmation number?

CA8920EE

Powering AI applications with AlloyDB AI

Increasingly, every app is an AI app. Some industries already have decades of experience incorporating advanced statistical and ML models into their applications, but the pace and breadth are accelerating. Let's look at an example of how intelligent agents have already transformed customer-facing apps.

For years, an auto insurer's app would have a customer fill out a form, then make a single, rigid call to a risk model to generate a quote. But now, instead of a form, a customer has a natural conversation: "Hi, I'm looking for car insurance for my 2025 Volkswagen Golf in Minneapolis."

On Google Cloud, an AI agent built on AlloyDB AI might then orchestrate the following series of actions:

- Use a tool to query the customer's profile in AlloyDB, retrieving their driving history and any loyalty discounts.
- Call an external API to fetch the latest safety ratings and typical repair costs for a Volkswagen Golf.
- Use another tool to run a risk model on Vertex AI, but now with a much richer, more complete set of data.
- Rather than just returning a price, the agent would analyze the result and explain it in plain language, suggesting optional add-ons: "For an extra $5 a month, you could add windshield coverage, which is common for this vehicle model."

In this scenario, AlloyDB AI isn't just feeding data into a model—it's the agent's dynamic source of truth, its long-term memory, and a key part of its toolkit. The agent's ability to orchestrate database lookups, API calls, and model predictions is what makes it fundamentally more powerful and useful.

Meet your data science team

In this new agentic world, collaboration is key. Your data science team might build and maintain the core predictive model (like the risk engine), but you, the developer, would build the agent that brings it to life. You orchestrate how that model is used alongside databases and APIs to create a complete, interactive user experience.

Explore some examples of developers working with data scientists to incorporate similarity search, sentiment analysis, bot detection, and healthcare predictions.

Building enterprise AI apps faster

Google Cloud databases are simple to integrate with your developer ecosystem. They support popular open source database standards like PostgreSQL, making it easier to migrate from legacy databases.

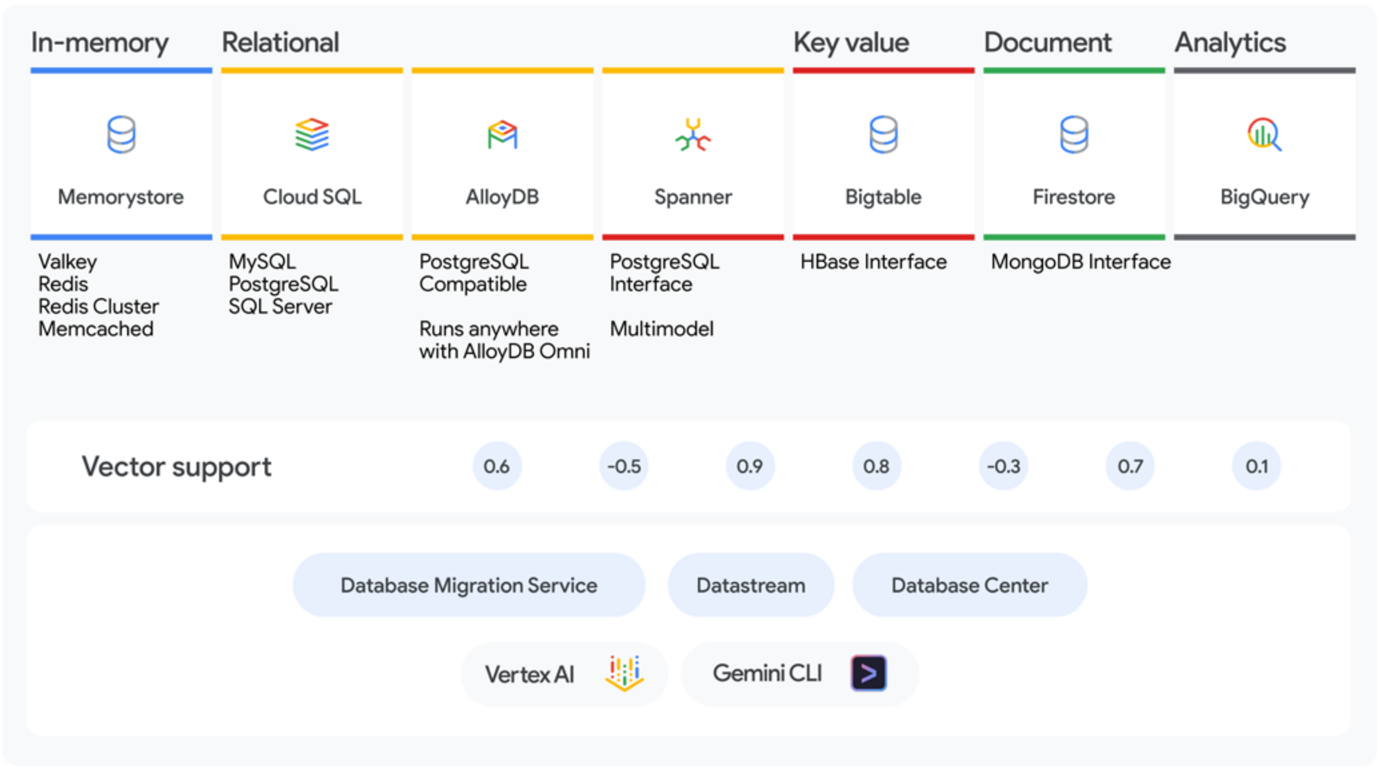
AlloyDB is optimized for enterprise AI apps that need real-time and accurate responses, delivering world-class vector embedding and search capabilities. It delivers superior performance for transactional, analytical, and vector workloads, and it runs anywhere, including on-premises and on other clouds, enabling you to modernize and innovate wherever you are.

AlloyDB AI is an integrated set of capabilities built into AlloyDB to help you build performant and scalable AI applications using your operational data. It helps you more easily and efficiently combine the power of foundation models with your real-time operational data.

“All of our databases have vector search capabilities natively available. That means you don’t have to deal with complex data pipelines to move your data to specialized vector stores. Furthermore, you can easily perform filter and join operations on your relational data with your familiar database interface. To top it all, you get strong performance, scalability, data protection, availability, security, and compliance from your database, which are core needs for any application, AI or not.”

Yannis Papakonstantinou
Distinguished Engineer, Query Processing,
Google Cloud

Infusing gen AI across Google Cloud databases



For applications requiring global scale, transactional consistency, and a 99.999% availability SLA, consider Spanner, a fully managed multi-model database. Spanner brings together relational, key-value, graph, full-text search, and vector search capabilities to enable a new class of AI-enabled applications that rely on interconnected data and semantic search—such as smarter recommendation engines in retail and sophisticated fraud detection in financial services.

Tutorial: Personalized recommendations using Spanner and Vertex AI

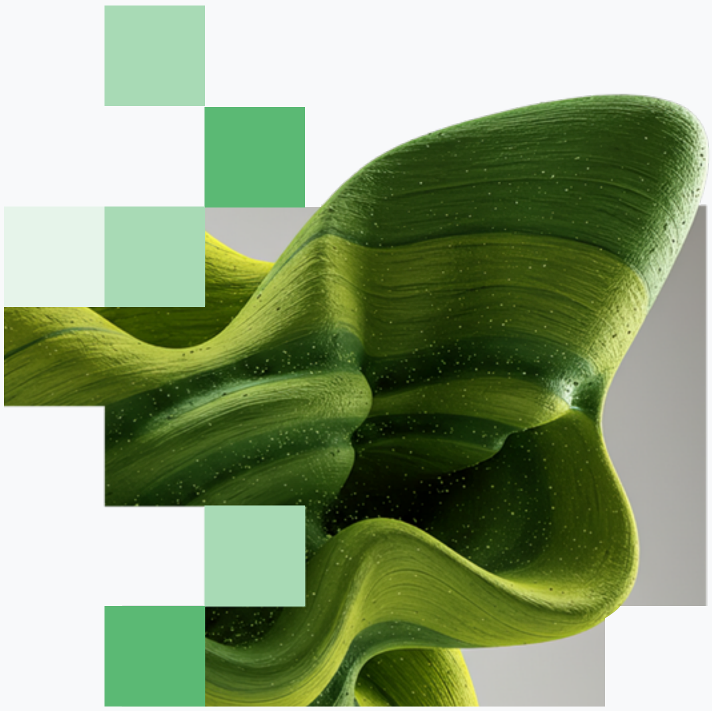
Learn how to use AI to provide personalized product recommendations in a sample ecommerce app.

Try the tutorial

Chapter 5

Build with Google's innovation

At Google, we have over a decade of experience innovating on real-world vector algorithms to support our most popular services, including Google Search and YouTube. We had to invent new ways of indexing and searching vectors to meet the most demanding use cases.



Connecting agents to your data

A powerful agent is one that can reliably access your data. To make this connection seamless for developers, Google Cloud supports two key requirements—deep integration with popular agent frameworks and robust support for open standards.

1. **Deep framework integration**

   For developers building within established ecosystems, we offer deep integrations with frameworks like LangChain, LlamaIndex, and Google's Agent Development Kit. These provide pre-built components that simplify using Google Cloud databases for everything from vector stores to chat message history. This path enables rapid development by leveraging a rich set of existing tools and patterns, like the built-in RAG workflows used for personalized recommendations, question answering, and customer service automation.

2. **Open standards**

   For universal, long-term interoperability, we're also strong supporters of open standards. For example, the MCP protocol provides a common, framework-agnostic language for agents and tools to communicate. To make this practical, our MCP Toolbox provides the necessary components to expose your Google Cloud database as a fully compliant MCP endpoint.

This open approach prevents framework lock-in and creates a more flexible, future-proof architecture for your entire AI ecosystem, giving you the flexibility to choose the best architectural approach for your project.

Use AI to supercharge database development and management

Database technology is evolving fast, so it's important for database professionals to stay up to date.

Your operational database is key to your organization's applications. You want to ensure that data can flow in and out smoothly and keep your application performing well. Database management comes with a lot of challenges—many platform engineers, database administrators, and developers juggle ill-fitting tools, complex scripts, and error-prone workflows to complete their tasks.

Google Cloud provides a unified, AI-powered experience, driven by Gemini, to assist you across the entire database journey. An integrated suite of tools helps you migrate legacy databases, accelerate development, optimize performance, and enforce data policies.

- **Migration:** Leverage foundation models to assess and convert the schema or database resident code when migrating databases. Easily learn new PostgreSQL dialects, optimize SQL code, and enhance readability for better productivity, easier migration, and higher efficiency.
- **Development:** Build and deploy applications faster. The Gemini command line interface (CLI) and Code Assist give you the power to perform complex tasks with simple natural language instructions—all from your terminal. Generate code, fix bugs, summarize documentation, and seamlessly query data with database extensions. When you're ready to build and deploy the AI applications themselves, Gemini Enterprise provides the environment to create and manage the agents that will interact with your data.
- **Unified management and optimization:** Administrators and platform engineers can manage their entire fleet of diverse databases from a single dashboard in Database Center. Integrated directly into this interface is Cloud Assist, an AI assistant for operations. Cloud Assist proactively monitors your fleet, identifies performance bottlenecks, highlights potential security or compliance issues, and provides actionable recommendations—all accessible through natural language conversation.
- **Data governance:** Set data policies to improve security, regulatory compliance, and control. Manage all your data across data silos in one centralized location. Use built-in data intelligence to check data validity and compliance.

Continue your AI journey with Google Cloud

The AI era is truly here. And when organizations scramble to implement new technologies, you have the opportunity to chart your career path.

The more AI skills you develop, the more valuable you'll be. As companies become more heavily data-driven, developers like you are increasingly taking on stakeholder roles. Where will your path lead? What skills will you invest in?

30-day free AlloyDB trial

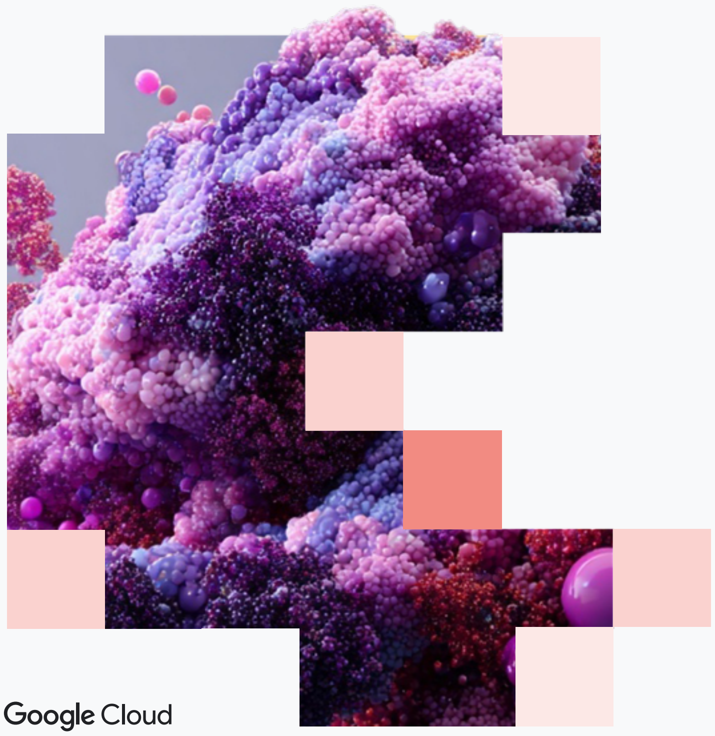
Start now

90-day free Spanner trial

Start now

90-day free Cloud SQL trial

Start now



Google Cloud

Start your transformation with Google Cloud

Talk to an expert