Unlocking gen AI's full potential with operational databases

Your guide to generative AI
app development

Using databases to build better generative AI apps

Everyone is talking about gen AI. It can improve search. It can personalize customer interactions. It can increase your developer productivity. It can even write bedtime stories.

As a developer, you already know that successful data collection, processing, and visualization can bring an app to life. But gen AI has raised the potential of data by leaps and bounds.

In the AI era, developers have the opportunity to become bigger stakeholders in the enterprise—producing insights that can only be found with powerful models and quality code. But for many developers, there's a chasm between the promise of AI models, including large language models, and the reality of putting them to work in enterprise use cases.

That's because AI models trained on generalized data are limited to answering general questions. Answering specific questions (like *"What's my account balance?"* or *"Do you have a size 8 in stock?"*) requires integrating business-specific operational information. **Your databases can bridge this gap.**

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

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

Looking to get started right away? Jump to the four pillars for successful gen AI apps on .

Google Cloud

> **Databases are going to be the fuel for this gen AI revolution. Operational databases bridge the gap between foundation models and enterprise applications to help deliver contextual, relevant, and accurate user experiences."**
>
> Andi Gutmans, GM and VP of Engineering, Databases, Google Cloud

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 they're looking for sits in your operational databases—such as your CRM, ERP, and e-commerce 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 this on their own—they need the information contained in your operational databases. And so, 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 data. It can be used to generate and translate text and other content, as well as perform other natural language processing (NLP) tasks. These are more generally called machine learning (ML) models as AI now also understands and operates across different types of formats, such as images, audio, and video.

Foundation models are trained on generalized data. As such, they can answer general questions such as *"What's the capital of France?"*, but don't have access to any information beyond what they were trained on.

The tide is turning to developers.

**Gen AI helps tackle enterprise challenges. And it's developers leading the charge.**

For enterprises to achieve the domain-specific, intelligent interactions that customers have come to expect, developers are taking a starring role. Consider that **according to the Bureau of Labor Statistics**, there are only around 150,000 data scientists in the U.S.— but over two million software developers. While these roles overlap at times, it's clear how critical developers are for AI innovation.

How are they innovating?

They're employing techniques like retrieval augmented generation to make foundation models dependable for enterprise applications. They're using prompt engineering, embeddings for proximity searches (typically in vector-enabled databases), and frameworks that help build AI-powered applications.

Are these terms unfamiliar to you? Don't worry. They won't be 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 gen AI technologies into applications. **Let's make sure that's you.**
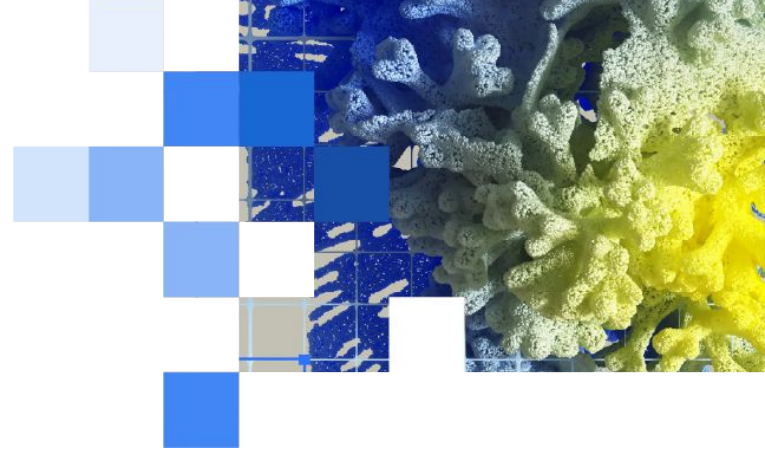
Skills required to develop gen AI apps:

- Programming
- Data modeling
- Database development
- Scalable architecting
- Integrating enterprise systems
- Using gen AI tools and frameworks
- Validating aspects of prompt engineering

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

Andi Gutmans, GM and VP of Engineering, Databases, Google Cloud

AI has raised the stakes.

AI models have come a long way, but they'll never be able to return relevant and accurate results 100% of the time. They're good, but they're not perfect.

Famously, models have an occasional tendency to "hallucinate"—AI terminology for when a model generates a factually incorrect or even nonsensical answer. AI models are notoriously poor at knowing that they don't know an answer to a question. And in that case they'll reply with whatever seems closest. The effects of hallucinations can be deleterious—most enterprise use cases can't tolerate **the shortcomings** of general purpose foundation model-based chatbots.

The stakes are incredibly high for you as a developer, because you know how critical it is for the app to return correct results. That's why it's essential to put safeguards at the core of your application to verify the outputs of the model.

Even with high stakes, there's no need to worry. Creating a robust AI application relies on some of the skills you already have: conscientious planning and testing. With databases doing much of the heavy lifting, it's important that you select the right database tools and integrations to make app development as intelligent, simple, and fast as possible.

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

Four pillars for successful gen AI apps

The most powerful enterprise gen AI apps augment generic foundation models, grounding these models in operational data to achieve relevancy, reliability, scalability, and security.

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

⚡ Relevancy

Deliver accurate and contextual information.

Grounding your model 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.

👍 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 gen AI setup should be highly available, resilient to failures, be simple to maintain without causing disruptions or downtime, and support disaster recovery.

⤬ 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. All of this should be quick to deploy and scale, and operate with high throughput and low latency.

🔒 Security

Keep customer and company data secure.

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

Some of these capabilities come out of the box from your database vendor, but you also need to architect for them within your app—including in any gen AI capabilities that the app provides.

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

The technologies powering gen AI apps

Users expect prompt, personalized interactions. But developers are finding that legacy databases aren't cutting it, due to their lagging gen AI capabilities. Thankfully, modern databases have excellent and rapidly evolving gen AI support. And in today's competitive landscape, this speed is a top priority.

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

- Long context windows
- Vector embeddings
- 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. And 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.
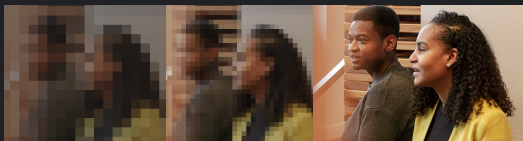
Non-text formats

The size of the context window is also a key issue for videos. Using typical settings, **one frame of video is 258 tokens**. Gemini 1.5 Pro has a context window of 2 million tokens, so it can process 2 hours of video or 1.4 million words. It's easy to see that a small context window won't get you very far once you move to non-text formats.

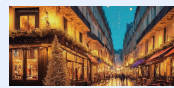Please find me a hotel in Paris for March 8-15
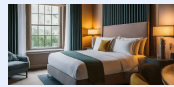
Hôtel A   Hôtel B   Hôtel C

What are some sights I should see?

Rue A   Tour B   Boutique C

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:

Restaurant A   Bistro B   Café C

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.

Google Cloud

Vector embeddings

Vector embeddings convert text, images, or other objects 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, and so the analysis of whether two vectors are near or far from another depends on your particular model.

This need to compare semantic meaning is why recommendation engines, search algorithms, and many other natural language processing applications rely on vector embeddings. Retrieval augmented generation workflows (which we'll cover on page 12) can use these vector embeddings to retrieve relevant data into foundation model prompts to refine them. This minimizes hallucinations, gives more context to the foundation model's answers, and provides more reliable information.

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

Working with vector embeddings

Some databases, such as AlloyDB, can directly generate vector embeddings. Learn how it can:

- Generate and store vector embeddings based on a model
- Index and query embeddings using the pgvector extension
- Work with embeddings from another source

Get started →

An example embedding workflow

Generate embeddings with your table-stored data and **pgvector functionality.** This example uses plain-text input to fetch a result from a database that relies on model-driven semantic parsing of the text's meaning.

```
Unset

SELECT id, name FROM items
  ORDER BY complaint_embedding
  <-> embedding('text-embedding-004', 'It
was the wrong color')::vector LIMIT 10;
```

Read more →

Retrieval augmented generation

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

RAG workflows can use vector embeddings to retrieve relevant data and augment the prompt to the model. Since RAG queries your operational database and retrieves only the data you need, it reduces the need for large context windows. And even more importantly, it gives context to the foundation model's answers and ultimately generates more accurate 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.

Minimize hallucination risks.

Generative models rely on statistical relationships learned during training, which sometimes causes them to generate plausible-sounding outputs that are factually incorrect. By connecting the model to reliable knowledge sources, grounding enables the model to cross-reference its responses against trustworthy data or verifiable facts, significantly increasing the accuracy and relevance of outputs.

Google Cloud databases include support for vectors, meaning you may not need a specialized vector database. Instead, you can streamline your embedding creation and access processes using a database you already know.

Meet AlloyDB.

AlloyDB is Google Cloud's PostgreSQL-compatible database that offers superior performance, availability, and scale. It's optimized for enterprise gen 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 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.

AlloyDB can scale to more than a billion vectors with typically less than 25 ms query latency.

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.

Get started building gen AI apps with AlloyDB AI →

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

In the case of RAG, the model receives prompts that are generated by a combination of a prompt template and a variety of input data. These prompts serve the same function (eliciting a response from an AI model) as user-generated prompts, but they are augmented with context and additional information.

How does RAG typically work?
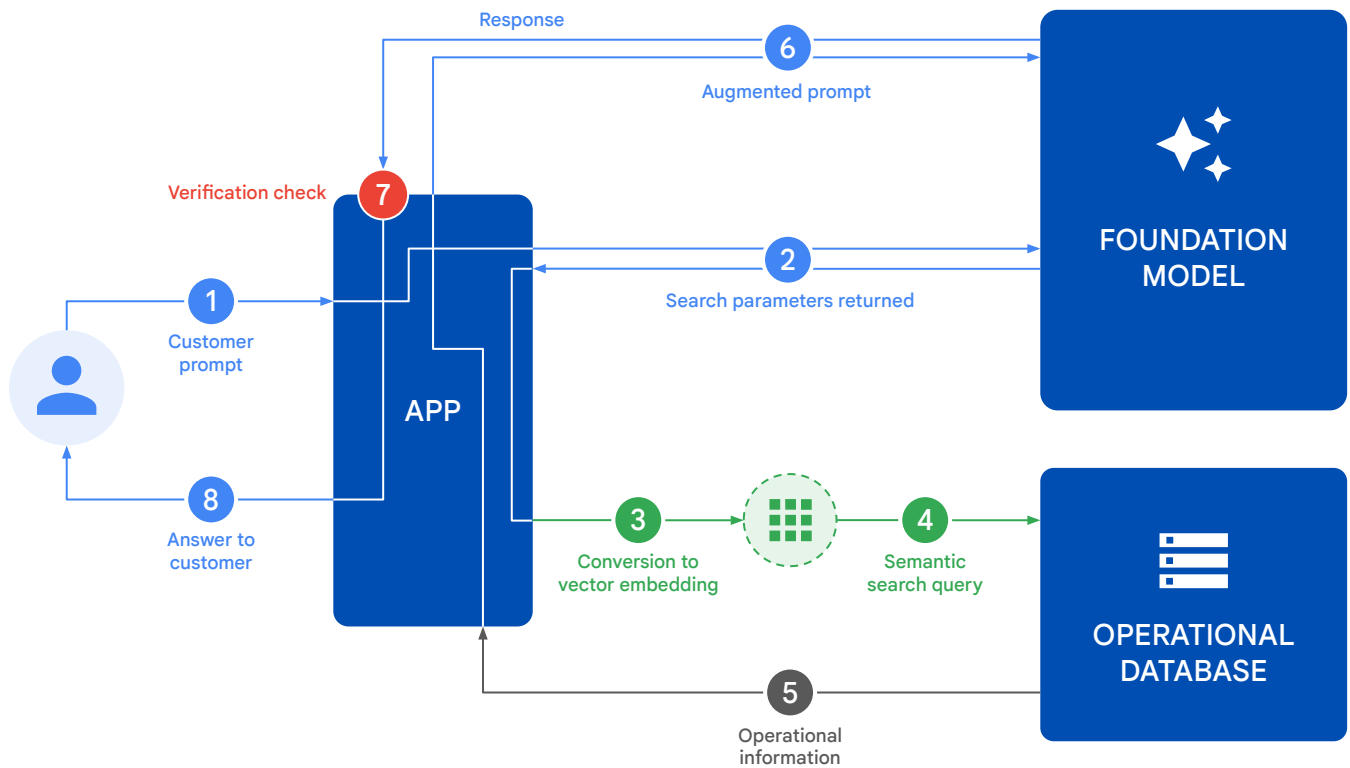
1 The customer enters a question (prompt) into the app

2 The foundation model identifies the intent of the question and returns search parameters

3 The app or database converts the request into a vector embedding

4 Semantic search queries the operational database

5 Operational information is returned from database

6 The app augments the prompt to the foundation model, the output is an answer from the foundation model

7 An application-specific verification check on the resultant output takes place

8 The app returns a response to the customer

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

Google Cloud

Skills Boost: RAG-based chat application with AlloyDB and Vertex AI

Use AlloyDB to store and search by vector embeddings, using a semantic search that retrieves the data that's the best match for a user's natural language query. The retrieved data is then passed to the model in the prompt.

In this lab, you'll learn how:

- RAG enhances foundation model capabilities by retrieving relevant information from a knowledge base
- AlloyDB can be used to find relevant information using semantic search
- You can use Vertex AI and Google's foundation models to provide powerful gen AI capabilities to applications

```
cd ~/genai-databases-retrieval-app/llm_demo
pip install -r requirements.txt
```

Learn how →

You'll notice that the retrieval service is run separately, not in the app itself. This method has a few benefits:

Better recall

Foundation models perform better when given smaller, discrete tasks they can use to accomplish larger goals. By mapping a specific action to a specific, predetermined query, the model can more successfully leverage the information.

Better scalability

Running the retrieval as a separate service allows multiple foundation models to leverage it and allows this service to scale independently. This allows for production best practices such as connection pooling or caching.

Better security

Foundation models are susceptible to attacks—such as "jailbreaking"—to circumvent safety measures. Using an intermediary service allows the application to handle authentication and authorization through more standard and secure channels, like existing authentication frameworks.

Orchestration frameworks

A simple gen AI app using a foundation model may have a straightforward prompt-to-response format that can be actioned with a single line of code.

However, some gen AI apps, such as those using more complex RAG, can have many branches and components. Orchestration frameworks allow you to streamline tasks such as API interactions, prompt engineering, data retrieval, state management, and workflows, so the app can work well even at high scale.

LangChain is an orchestration framework that helps you build gen AI applications or RAG workflows—providing the structure, tools, and components to streamline complex workflows.

Learn how to build with LangChain →

Google Cloud databases support three commonly used LangChain features: vector store, document loaders, and chat message history.

Learn more about these integrations →

Not all AI technologies are created equal.

We've covered four 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 specially trained 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. And that's 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.

Putting these technologies to work

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

Here are some questions to ask:

- **Which foundation model should I use?** Several excellent ones are available on Google Cloud, such as Gemini.

- **Which database should I use?** This decision will be driven not only by your gen AI needs, but by your database needs more broadly.

- **What type of nearest neighbor search algorithm should I use**: exact k-nearest neighbors algorithm (KNN) or approximate nearest neighbor search (ANN)? ANN is a performance-optimized approach that may be preferred when approximate search provides sufficient accuracy.

- **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 (such as LangChain)?** Orchestration is commonly used to build apps that use RAG.

Do you need a specialized vector database?

Not necessarily. Many popular databases include support for vectors, and your existing database may allow you to simply add a vector column or index. This way you don't need to learn and install a new database.

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 engines, conversational AI, and personalized recommendations.

Although vector search is important, your app probably needs more than that: relational (or non-relational) data, scalability, security, disaster recovery, and more. 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 can 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 gen 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

```python
# create the ConversationalRetrievalChain
rag_chain = ConversationalRetrievalChain.from_llm(
    llm=llm,
    retriever=retriever,
    verbose=False,
    memory=memory,
    condense_question_prompt=condense_question_prompt_passthrough,
    combine_docs_chain_kwargs={"prompt": prompt},
```
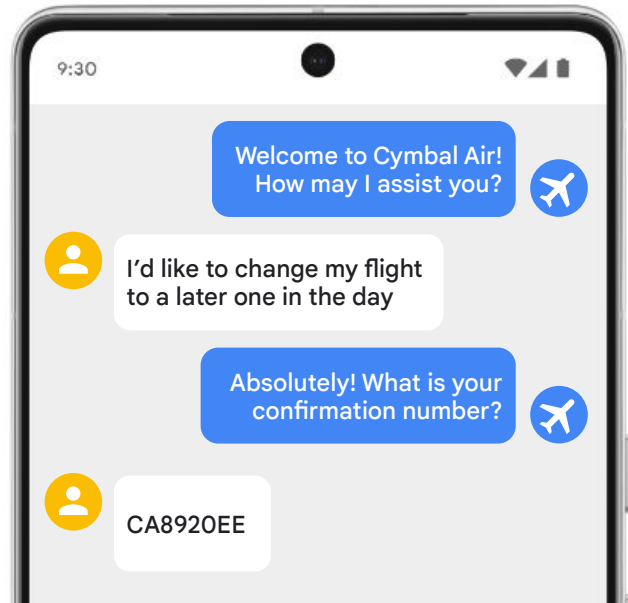
[Get the notebook →]

Codelab: RAG-based chat application using AlloyDB AI and LangChain

Learn how to build a foundation model- and RAG-based chat application by deploying the GenAI Databases Retrieval Service and create a sample interactive application using the deployed environment.

You'll learn how to:

- Deploy an AlloyDB Cluster
- Connect to AlloyDB
- Configure and deploy the GenAI Databases Retrieval Service
- Deploy a sample application using the deployed service

Start the codelab →

9:30

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

As applications become more intelligent, we'll reach the point where every app is an AI app. In fact, some industries already have decades of experience incorporating advanced statistical and ML models into their applications.

Take insurance, for example. Estimating risk is at the core of an insurance company, so it needs good risk models in order to issue policies and process claims.

Let's say you're an auto insurance company that offers online quotes. Every time a customer applies for a policy, you need to evaluate the risk based on their data (e.g., driving record) and run a risk model to generate a quote.

If customer input and other data are stored in AlloyDB, you could use AlloyDB AI to access the data and run a risk model hosted on Vertex AI. The same approach would work for claims processing, where a model determines the validity of the claim, and for other types of fraud detection in insurance and finance.

In a Google Cloud environment, the AlloyDB database would make calls to a risk or a claims model running on Vertex AI. Google makes it easy for you to make these calls.

Meet your data science team.

If your organization uses complex custom models, you probably don't need to train them yourself—they're likely to be developed by data scientists, and you just need to call those models from your app. Explore some examples of developers working with data scientists to incorporate similarity search, sentiment analysis, bot detection, and healthcare predictions.

Build enterprise gen AI apps faster.

AlloyDB delivers world-class vector embedding and search capabilities. Across the Google Cloud portfolio, relational databases and non-relational databases alike offer gen AI features to provide a deeper, more meaningful understanding of your data.

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

It's also easy to connect your database to external services that provide additional AI inferencing services, such as Vertex AI, and integrate with orchestration frameworks such as LangChain.

> "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, gen AI or not."

Yannis Papakonstantinou, Distinguished Engineer,
Gen AI and Query Processing, Google Cloud Databases

Industry-leading databases for your next project

In-memory	Relational				Key-value	Document
Memorystore	Bare Metal Solution	Cloud SQL	AlloyDB	Spanner	Bigtable	Firestore
Redis Cluster Redis Memcached Valkey	Oracle	MySQL PostgreSQL SQL Server	PostgreSQL Compatible Runs anywhere with AlloyDB Omni	PostgreSQL Interface Multimodel	HBase Interface	

| Managed third-party database engines | | | Google's cloud-first database engines | | | |

🗄 Database Migration Service

🗄 Datastream

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 or sophisticated fraud detection in financial services.

Tutorial: Personalized recommendations using Spanner and Vertex AI

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

Try the tutorial →

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.

In addition to support for open-source pgvector for our PostgreSQL databases, we're bringing the next generation of tree-based vector capabilities to relational databases. **The ScaNN index**, available in AlloyDB and Spanner, is a pgvector-compatible index based on Google's state-of-the-art approximate nearest neighbor algorithms.

To enhance choice and flexibility, Model Endpoint Management is available in several Google Cloud databases to facilitate access to any model on any platform. It's easy to call remote Vertex AI, third-party, and custom models, including third-party services such as Anthropic and Hugging Face.

Accelerating ecosystem support for LangChain

Integration with specific LangChain components simplifies the process of incorporating Google databases into apps. By leveraging the power of LangChain with our databases, you can create context-aware gen AI applications, faster.

Our LangChain integration provides built-in RAG workflows across your preferred data sources, using the Google Cloud database of your choice. Example use cases include personalized product recommendations, question answering, document search and synthesis, and customer service automation.

Supercharge database development and management with AI.

Database technology is evolving fast, and database professionals are finding it hard to stay up to date—which hampers both programming quality and productivity.

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 databases provide an AI-powered assistant that helps you in every aspect of the database journey across migration, development, performance optimization, fleet management, and governance. It helps you move away from legacy databases and migrate your data to the cloud, so you can become more efficient at what you do.

Migration

Leverage foundation models to assess and convert the schema or database resident code before migrating data. 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 while meeting security and high availability needs with Gemini's ability to generate, fine-tune, and summarize SQL code with simple natural language instructions.

Performance optimization

Administrators and developers can address database performance issues through an easy-to-use interface that highlights problems and provides recommendations. It provides visibility into all metrics in a single view to save time and enhance productivity.

Fleet management

Administrators and platform engineers can manage an entire fleet of diverse databases using intelligent dashboards, proactively assessing availability, data protection, security, and compliance issues without any custom tools or processes. With the integrated AI assistant in Database Center, you can interface with the system using natural language, making it easier to troubleshoot problems.

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.

Google Cloud

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?

Read more about Google Cloud databases →

Start a 30-day AlloyDB free trial →

Start a 90-day Spanner free trial →

Start a 90-day Cloud SQL free trial →