

# Distributed AI: global and geo-partitioned RAG architecture with Google Cloud Spanner for geo-location aware similarity search

Creating and querying a geo-partitioned global database for embedding-based similarity search



Christoph Bussler

10 min read · Just now



## Introduction

With the newly introduced geo-partitioning functionality of Google Cloud Spanner (<https://cloud.google.com/spanner/docs/geo-partitioning>) a distributed RAG system is supported by Cloud Spanner that enables geo-local as well as global embedding-based similarity search.

This blog outlines a basic global geo-partitioned RAG (Retrieval Augmented Generation) architecture based on Cloud Spanner. I provide all `gcloud` commands to create a global geo-partitioned database, embedding retrieval and similarity search queries for geo-local and global searches.

Vector databases are all the Rage (<https://chbussler.medium.com/vector-databases-are-all-the-rage-872c888fa348>). Cloud Spanner provides the

vector data type as well as similarity search

(<https://cloud.google.com/spanner/docs/find-k-nearest-neighbors>) and hence also belongs to the category of vector databases.

## Use case: international news

The use case for this blog is search of international news based on their publishing geo-location: Europe, Asia, Americas. The news is made accessible to similarity search based on embeddings represented as vectors.

The news is stored by geo-location in corresponding geo-partitions in Cloud Spanner. This enables global search across all geo-locations as well as geo-location specific search within partitions.

The blog shows queries for both, global and geo-local similarity search.

**Note:** for cost saving reasons, I use regional Cloud Spanner configurations. In a production deployment that might be insufficient — multi-region configurations are possibly a better alternative.

**Note:** In order to run the commands in your cloud project, I used `<project-id>` wherever you have to substitute it with your cloud project.

## Overview

In the following the following steps are executed to setup a global geo-partitioned database for global and geo-local similarity search:

- Create a Cloud Spanner instance and database
- Create partitions (one each for Europe, Asia, Americas)

- Create placements (one each for the three geo-locations)
- Create a table `newschunk`
- Create a model (for retrieving embeddings from within the database)
- Insert embeddings into the table
- Execute geo-local as well as global similarity search queries

## Instance and database creation

```
gcloud spanner instances create news-international-instance
--config=regional-us-central1
--description="Instance for NewsInternational"
--nodes=1
```

A configuration choice for a global production system could be: `nam-eur-asia3`.

A default partition is required to set up a partitioned database. The initial database is going to be the `default` partition.

```
gcloud spanner databases create news-international-database
--instance=news-international-instance
```

Since geo-partitioning is in preview, the following configuration has to be applied to the database.



```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="ALTER DATABASE db
      SET OPTIONS (opt_in_dataplacement_preview = true);"
```

The following command is added just in case you need it for your deployment to start over. This will delete the partitions and placements as well in addition to deleting the tables and the data.

```
gcloud spanner instances delete news-international-instance
```

At the completion of the deployment and the creation of the partitions (`gcloud` commands are yet to come below), the console will show that the default partition plus the three non-default partitions are running in 4 nodes.

 **Filter** Enter property name or value

<input type="checkbox"/>	Name 	ID	Partitions	Configuration	Processing units 	Nodes 
<input type="checkbox"/>	<a href="#">Instance for NewsInternational</a>	news-international-instance	4	us-central1 (lowa) 	4,000	4

Overview of instance

The following console representation shows additional details (like the partition identifiers).

## Instance for NewsInternational

Instance ID	news-international-instance
Partition ID	default
Configuration	us-central1 (low) ⓘ 99.99% availability SLA 3 read-write replicas in 3 separate zones within the region us-central1 <a href="#">^ HIDE DETAILS</a>
Scaling mode	Manual allocation
Additional partition IDs	americas-partition, asia-partition, europe-partition

Detailed overview of instance

## Partition creation

One partition each for Europe, Asia and Americas is created in the following.

```
gcloud beta spanner instance-partitions create americas-partition
--config=regional-us-east1
--description="americas-partition"
--instance=news-international-instance
--nodes=1
```

A configuration choice for a global production system could be: `nam3` .

```
gcloud beta spanner instance-partitions create europe-partition
--config=regional-europe-west1
--description="europe-partition"
--instance=news-international-instance
--nodes=1
```

A configuration choice for a global production system could be: `eur3`.

```
gcloud beta spanner instance-partitions create asia-partition
--config=regional-asia-southeast2
--description="asia-partition"
--instance=news-international-instance
--nodes=1
```

A configuration choice for a global production system could be: `asia1`.

The following command lists the specified partitions. You can use it to check that they are in place and correct.

```
gcloud beta spanner instance-partitions list
--instance=news-international-instance
```

The output is as follows:

```
NAME: americas-partition
DISPLAY_NAME: americas-partition
CONFIG: regional-us-east1
NODE_COUNT: 1
PROCESSING_UNITS:
STATE: READY

NAME: asia-partition
DISPLAY_NAME: asia-partition
CONFIG: regional-asia-southeast2
NODE_COUNT: 1
PROCESSING_UNITS:
STATE: READY
```

```
NAME: default
DISPLAY_NAME: Default Instance Partition
CONFIG: regional-us-central1
NODE_COUNT: 1
PROCESSING_UNITS:
STATE: READY
```

```
NAME: europe-partition
DISPLAY_NAME: europe-partition
CONFIG: regional-europe-west1
NODE_COUNT: 1
PROCESSING_UNITS:
STATE: READY
```

The cloud console shows the following:

Filter	Filter partitions					
●	Partition ID ↑	Configuration	Availability	Processing units	Nodes	Databases
✓	americas-partition	us-east1 (South Carolina) ⓘ	99.99%	1,000	1	1
✓	asia-partition	asia-southeast2 (Jakarta) ⓘ	99.99%	1,000	1	1
✓	default	us-central1 (Iowa) ⓘ	99.99%	1,000	1	1
✓	europe-partition	europe-west1 (Belgium) ⓘ	99.99%	1,000	1	1

List of partitions in cloud console

## Placement creation

For each partition one placement is created. A placement specifies a unique value that is used in tables to specify the partition the data is stored in ( `PLACEMENT KEY` ). The same value can be used in queries for geo-location specific queries on partitions.

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="CREATE PLACEMENT americas
      OPTIONS (instance_partition='americas-partition')"
```

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="CREATE PLACEMENT europe
      OPTIONS (instance_partition='europe-partition')"
```

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="CREATE PLACEMENT asia
      OPTIONS (instance_partition='asia-partition')"
```

Check that placements are in place and correct with the following commands:

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='SELECT * FROM information_schema.placements'
```

The output is

```
PLACEMENT_NAME: default
IS_DEFAULT: True

PLACEMENT_NAME: americas
IS_DEFAULT: False
```



```
PLACEMENT_NAME: asia
```

```
IS_DEFAULT: False
```

```
PLACEMENT_NAME: europe
```

```
IS_DEFAULT: False
```

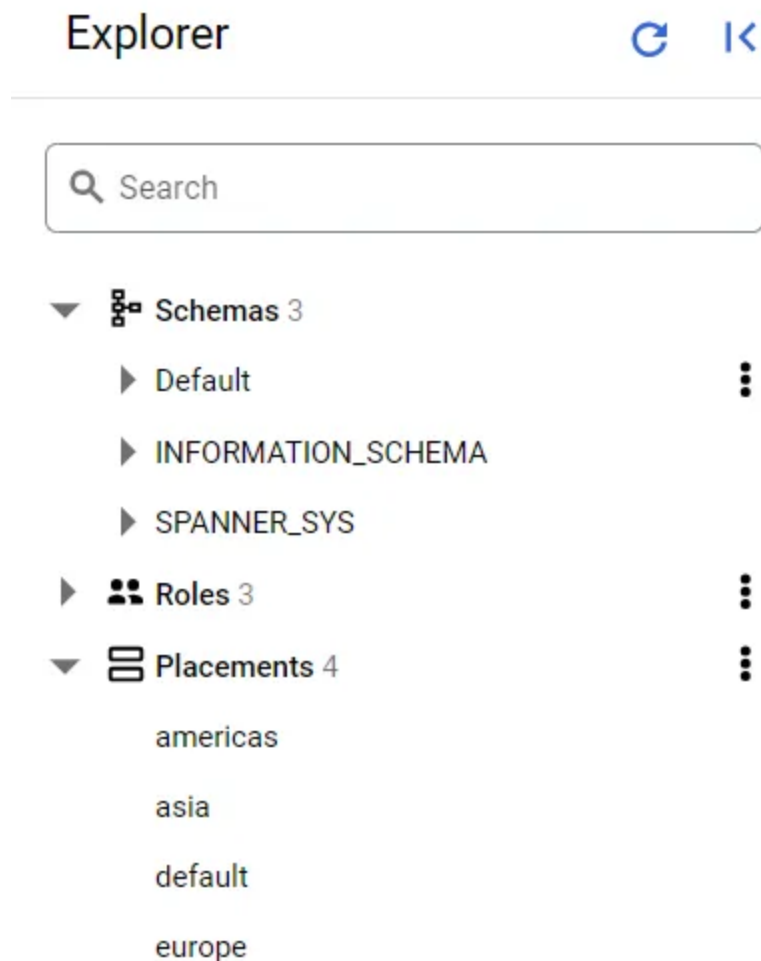
And this command for more details on placements:

```
gcloud spanner databases execute-sql news-international-database  
--instance=news-international-instance  
--sql='SELECT * FROM information_schema.placement_options'
```

The output is:

```
PLACEMENT_NAME: defaults  
OPTION_NAME: instance_partition  
OPTION_TYPE: STRING (MAX)  
OPTION_VALUE: default  
  
PLACEMENT_NAME: americas  
OPTION_NAME: instance_partition  
OPTION_TYPE: STRING (MAX)  
OPTION_VALUE: americas-partition  
  
PLACEMENT_NAME: asia  
OPTION_NAME: instance_partition  
OPTION_TYPE: STRING (MAX)  
OPTION_VALUE: asia-partition  
  
PLACEMENT_NAME: europe  
OPTION_NAME: instance_partition  
OPTION_TYPE: STRING (MAX)  
OPTION_VALUE: europe-partition
```

In the cloud console placements are shown in the Explorer as follows:



List of placements in the Explorer

## Table creation

The following table serves an example of storing news. A news article is broken up in chunks (using a chunking algorithm). For each chunk a row is inserted that contains the chunk itself, its corresponding embedding (for search) and a reference to the entire article.

In addition to an identifier serving as key, the location is stored which refers to placement values created earlier. This ties the row to the corresponding partition.

```
CREATE TABLE newschunk (  
  id INT64 NOT NULL,  
  news_chunk STRING(MAX) NOT NULL,  
  embedding ARRAY<FLOAT64>,  
  news_document STRING(MAX) NOT NULL,  
  location STRING(MAX) NOT NULL PLACEMENT KEY  
) PRIMARY KEY(id)
```

Create it with this command:

```
gcloud spanner databases ddl update news-international-database  
--instance=news-international-instance  
--ddl="CREATE TABLE newschunk(  
id INT64 NOT NULL,  
news_chunk STRING(MAX) NOT NULL,  
embedding ARRAY<FLOAT64>,  
news_document STRING(MAX) NOT NULL,  
location STRING(MAX) NOT NULL PLACEMENT KEY  
) PRIMARY KEY(id);"
```

Run a query to confirm the table's existence:

```
gcloud spanner databases execute-sql news-international-database  
--instance=news-international-instance  
--sql='SELECT * FROM newschunk'
```

I provide the following just in case you need it during development:

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="DROP TABLE newschunk"
```

## Embeddings retrieval and insertion

Like in AlloyDB AI (<https://medium.com/google-cloud/core-rag-architecture-with-alloydb-ai-7c7388f33ff1>) it is possible to retrieve embeddings from within database queries like in the following. First a model has to be created with a specified Vertex AI location (for example, us-central1 — not us-central-1) as it is in the leader region of nam-eur-asia3 (see the documentation <https://cloud.google.com/spanner/docs/ml-tutorial-embeddings>).

You might have to allow access to the Vertex AI models from Cloud Spanner first:

```
gcloud beta services identity create
--service=spanner.googleapis.com
--project=<project-id>
```

And you might have to enable the Vertex AI API:

```
https://console.developers.google.com/apis/api/aiplatform.googleapis.com/overview
```

The model specification looks like this with `us-central1` as location and version `003` of the model `textembedding-gecko`:

```
CREATE MODEL textembeddinggecko
INPUT(content STRING(MAX))
OUTPUT(
  embeddings
  STRUCT<
    statistics STRUCT<truncated BOOL, token_count FLOAT64>,
    values ARRAY<FLOAT64>>
  )
REMOTE OPTIONS (
  endpoint = '//aiplatform.googleapis.com/projects/<project-id>/locations/us-centr
);
```

Create the model with the following command:

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="CREATE MODEL textembeddinggecko
INPUT(content STRING(MAX))
OUTPUT(
  embeddings
  STRUCT<
    statistics STRUCT<truncated BOOL, token_count FLOAT64>,
    values ARRAY<FLOAT64>>
  )
REMOTE OPTIONS (
  endpoint = '//aiplatform.googleapis.com/projects/<project-id>/locations/us-cer
);"
```

The following command I provide in case you need it during your development:

```
gcloud spanner databases ddl update news-international-database
--instance=news-international-instance
--ddl="DROP MODEL textembeddinggecko"
```

The following command queries the model with an input text string ( `In yesterday's event ...` ), retrieves the embedding for that input text string with the model and stores it in the table. Note that this row is inserted into the `americas` partition:

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='INSERT INTO
newschunk (id, news_chunk, embedding, news_document, location)
SELECT 1, "In yesterday's event ...", embeddings.values,
"http://yes...", "americas"
FROM ML.PREDICT(
MODEL textembeddinggecko,
(SELECT "In yesterday's event ..." AS content)
);'
```

It might be that the command fails with the following error message. In that case, repeat it until it succeeds: `ERROR: (gcloud.spanner.databases.execute-sql) ABORTED: Transaction was aborted.`

To check for the inserted row execute the following command:

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='SELECT * FROM newschunk'
```

Two additional texts are inserted, one for each of the remaining partitions ( europe , asia ) like this:

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='INSERT INTO
newschunk (id, news_chunk, embedding, news_document, location)
SELECT 2, "In tomorrow's event ...", embeddings.values,
"http://tom...", "europe"
FROM ML.PREDICT(
MODEL textembeddinggecko,
(SELECT "In tomorrow's event ..." AS content)
);'
```

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='INSERT INTO
newschunk (id, news_chunk, embedding, news_document, location)
SELECT 3, "In today's event ...", embeddings.values,
"http://tod...", "asia"
FROM ML.PREDICT(
MODEL textembeddinggecko,
(SELECT "In today's event ..." AS content)
);'
```

To check that one row for each partition was created, run:

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='SELECT id, news_chunk, news_document, location
FROM newschunk;'
```

The output is

```
id: 1
news_chunk: In yesterday's event ...
news_document: http://yes...
location: americas

id: 2
news_chunk: In tomorrow's event ...
news_document: http://tom...
location: europe

id: 3
news_chunk: In today's event ...
news_document: http://tod...
location: asia
```

## Geo-location and global similarity query

Executing a similarity search requires an input embedding that is used by the similarity search to compare it with the stored embeddings. The Cloud Spanner documentation shows examples here: <https://cloud.google.com/spanner/docs/find-k-nearest-neighbors>. As you can see, an embedding is provided as an `ARRAY` (vector) constant in these examples.

In an application, however, the application would have to compute the input embedding based on for example user input. If the user wants to search for documents that contain similar content to what the user enters, the similarity search would take the user's input and from that creates the input embedding by retrieving the embedding using a text embedding model.

For the purposes of this blog, I compute an embedding based on a fixed string within the retrieval query. This is to show the principle and to be able to execute a query without having to build an application in this blog.



```
User input: "In some day's event ..."
```

```
Corresponding input embedding: ML.PREDICT(  
  MODEL textembeddinggecko,  
    (SELECT "In some day's event ..." AS content)  
)
```

## Global similarity search

The following query searches in all partitions of the database. It is a global query searching news from all geo-locations by similarity:

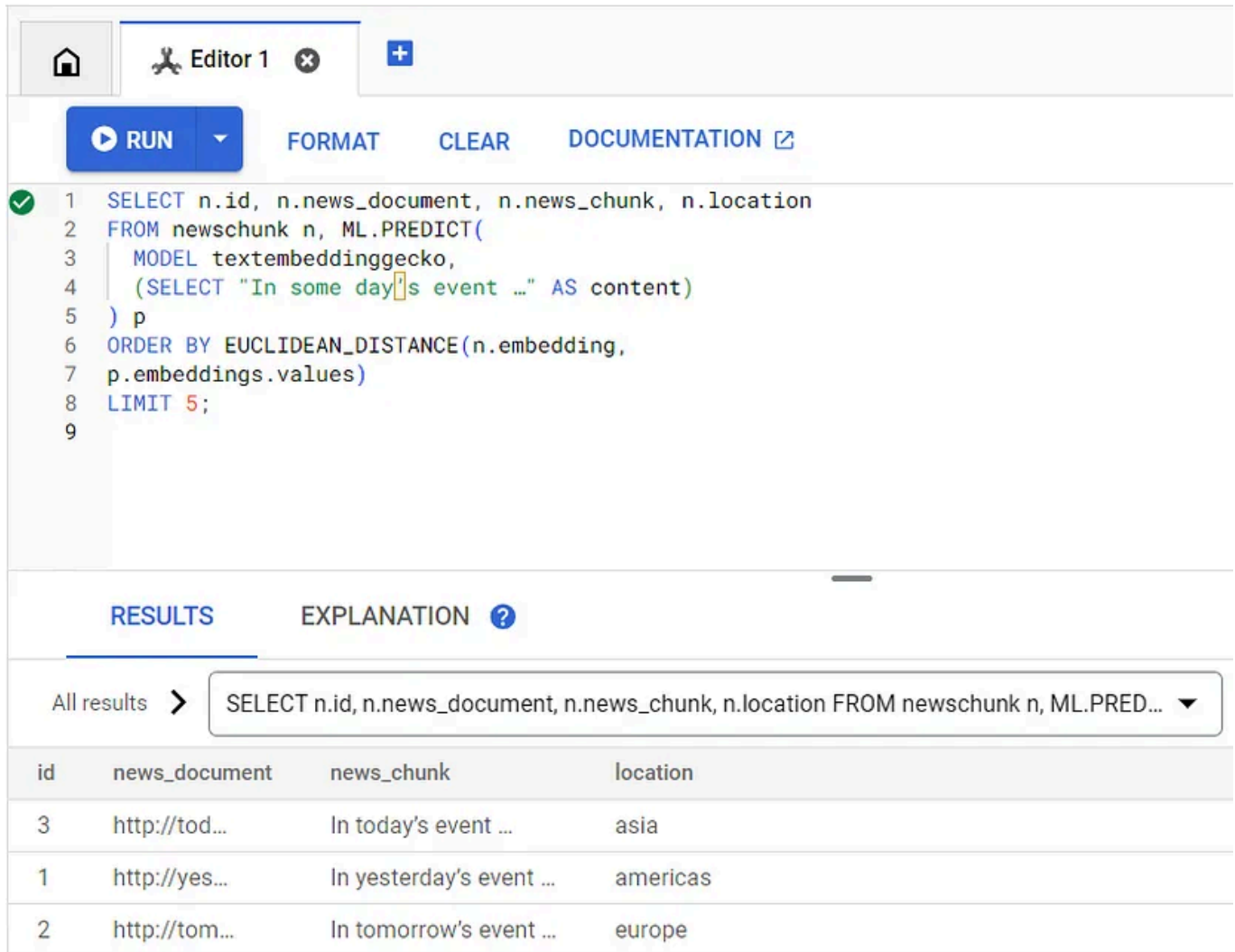
```
gcloud spanner databases execute-sql news-international-database  
--instance=news-international-instance  
--sql='SELECT n.id, n.news_document, n.news_chunk, n.location  
  FROM newschunk n, ML.PREDICT(  
    MODEL textembeddinggecko,  
      (SELECT "In some day's event ..." AS content)  
    ) p  
  ORDER BY EUCLIDEAN_DISTANCE(n.embedding,  
    p.embeddings.values)  
  LIMIT 5;'
```

## The result is

```
id: 3  
news_document: http://tod..  
news_chunk: In today's event ...  
location: asia  
  
id: 1  
news_document: http://yes..  
news_chunk: In yesterday's event ...  
location: americas
```

```
id: 2
news_document: http://tom...
news_chunk: In tomorrow's event ...
location: europe
```

The same query run in the Explorer is as follows:



The screenshot shows a database Explorer interface with a query editor and a results table. The query editor contains a SQL query that uses a machine learning model to predict the location of news documents based on their content. The results table shows the top 5 results, ordered by Euclidean distance.

```
1 SELECT n.id, n.news_document, n.news_chunk, n.location
2 FROM newschunk n, ML.PREDICT(
3   MODEL textembeddinggecko,
4   (SELECT "In some day's event ..." AS content)
5 ) p
6 ORDER BY EUCLIDEAN_DISTANCE(n.embedding,
7   p.embeddings.values)
8 LIMIT 5;
9
```

RESULTS      EXPLANATION ?

All results > SELECT n.id, n.news\_document, n.news\_chunk, n.location FROM newschunk n, ML.PRED...

id	news_document	news_chunk	location
3	http://tod...	In today's event ...	asia
1	http://yes...	In yesterday's event ...	americas
2	http://tom...	In tomorrow's event ...	europe

Similarity query in Explorer

## Geo-local similarity search

A corresponding similarity query that only searches in news in the `americas` partition (geo-local search) limits the scope of the query by the predicate `WHERE location = "americas"` (using the placement key):

```
gcloud spanner databases execute-sql news-international-database
--instance=news-international-instance
--sql='SELECT n.id, n.news_document, n.news_chunk, n.location
FROM newschunk n, ML.PREDICT(
  MODEL textembeddinggecko,
  (SELECT "In some day's event ..." AS content)
) p
WHERE location = "americas"
ORDER BY EUCLIDEAN_DISTANCE(n.embedding,
  p.embeddings.values)
LIMIT 5;'
```

The output is

```
id: 1
news_document: http://yes...
news_chunk: In yesterday's event ...
location: americas
```

This concludes the creation of the Cloud Spanner instance, database, partitions, placements, model, table and test data queried by geo-local and global similarity search.

## Architecture improvement possibilities

### Country-specific news

The partitions are by a larger geographic area encompassing many countries each. It is possible to have placements not only for geolocations like Europe or Asia, but in addition for individual countries if the new source is to be associated with a specific country. This then supports country specific similarity searches as well.

Of course, in this case the similarity queries become more complex, especially if news of several countries are searched.

### **AI embedding model endpoint for each partition**

Specifying a model ( `CREATE MODEL` ) requires specifying a model access endpoint location (above it is using the endpoint in `us-central1` ). There are many regions that have model access endpoints: <https://cloud.google.com/vertex-ai/docs/general/locations>.

Therefore it is possible to create different models, one close or in the same regions as the partitions. If the queries were to run close or in the region of the partition the call to the model would be as close as it can be.

## **Conclusion**

The newly implemented partition feature of Cloud Spanner supports geo-location similarity queries as the base of a global RAG architecture.

A partitioned database supports geo-local similarity queries scoped to a partition, and in the same partitioned database global queries across all partitions.

This is a very expressive approach for distributed AI systems based on the RAG architecture approach of storing and querying embeddings represented as vectors on a global basis.

Cloud Spanner

Distributed Ai

Vector Database

Vector Search

Embedding Model

