

ORACLE

Using AI with Oracle GoldenGate

Practical patterns for CDC replication, AI enrichment, and administration



Contents

Executive Summary.....3

Where AI Fits in a GoldenGate Deployment.....4

The GoldenGate AI Service.....5

Embedding Generation Example.....8

Operating the Pipeline.....15

Using AI to Operate GoldenGate 17

GoldenGate Data Fabric and the AI Truth Layer 23

Conclusion.....25

Appendices.....26



Executive Summary

Every AI answer, alert, and agent action is downstream of a committed business event. Oracle GoldenGate already captures those events for numerous use cases, including homogeneous and heterogeneous replication, analytics, data fabric, and high availability, which makes an existing GoldenGate deployment the shortest path to AI that works from current operational truth. This paper shows three practical ways to use AI with the CDC and replication pipelines you already run: supply replicated business data to AI applications, generate vector embeddings as data is replicated, and use AI agents to operate GoldenGate itself.

The centerpiece is the GoldenGate AI Service in GoldenGate 26ai. It connects Replicat to embedding models, and one mapping line, @AISERVICE, turns replicated text into vectors in the same target transaction as the row. The model can come from an AI model registered in a self-managed GoldenGate deployment, from an AI Model connection in fully managed OCI GoldenGate, or from a local ONNX model that runs inside your own deployment. A detailed example of a product catalog walks through the full pipeline, from capture to a semantic search that reads current price and stock.

The second half turns to operations. An AI agent, guided by a Markdown runbook and working through the GoldenGate MCP server and REST APIs, carries out a parallel initial load and resumes CDC, with an administrator authorizing each change. Two further examples cover lag diagnosis and change review. The closing section places these patterns inside Oracle GoldenGate Data Fabric, where GoldenGate serves as the AI Truth Layer for distributed operational data.

Key Takeaways

- Start from the GoldenGate deployment you have. The same Extract that keeps a replica current can keep AI context and vector search current using existing trails or writing to new ones.
- Generate embeddings as data is replicated. The @AISERVICE feature in a Replicat mapping calls a model through the GoldenGate AI Service.
- Choose where the model runs. Use a hosted provider from a self-managed deployment, an AI Model connection in OCI GoldenGate, or a local ONNX model that keeps inference inside your environment.
- Use Parallel Replicat. It maps records and applies independent transactions concurrently.
- Embed history with the same mapping. Run the initial load through GoldenGate at a recorded SCN, then start Replicat after that SCN.
- Put AI agents to work on operations. A Markdown runbook, the GoldenGate MCP server, and REST APIs let an agent plan, carry out authorized steps, and record evidence.
- Evolve into GoldenGate Data Fabric. The patterns here are the building blocks of governed, reusable data products and an AI Truth Layer for operational data.

Audience and Scope

This paper is for data engineers, GoldenGate administrators, and database and AI architects who already work with Extracts, trails, Replicats, and initial instantiations. It explains how existing CDC and replication workflows supply AI context, generate embeddings in real time, and support AI-assisted administration.

Examples use Oracle GoldenGate 26ai Microservices Architecture at release update 23.26.3 or later, and OCI GoldenGate, with Oracle AI Database 26ai as the target; the certification matrix [1] lists supported source and target combinations. The running example is a product catalog: name, category, and description carry the semantic content, while price and stock supply current structured context.

Where AI Fits in a GoldenGate Deployment

AI reaches a GoldenGate deployment at three points, shown in Figure 1. Replicated data becomes current context for AI applications. Replicat generates embeddings while it applies changes. AI agents help administrators run the replication itself. Table 2 maps those three points onto the work GoldenGate already does.

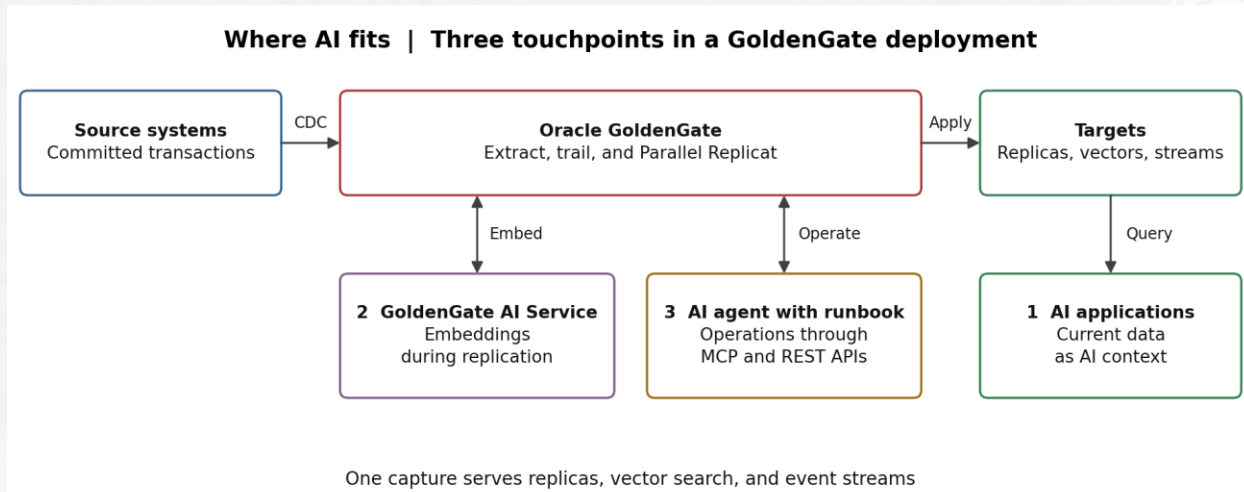


Figure 1. Three AI touchpoints in a GoldenGate deployment.

Table 2. AI touchpoints by GoldenGate use case.

Use case you run today	Replicated data as AI context	Embeddings during replication	AI-assisted operations
Reporting or offload replica	AI applications and agents query the replica, and the production source stays free for transactions.	Add a vector column on the replica and fill it with @AISERVICE.	Lag diagnosis and change review.



Use case you run today	Replicated data as AI context	Embeddings during replication	AI-assisted operations
Keeping analytics platforms current	AI applications and analytical models use current business data delivered to the analytics platform.	Enrich analytical datasets with embeddings during replication, where supported by the target.	Runbook-guided initial loads, data freshness monitoring, and lag diagnosis.
High availability and active-active	Each site serves AI workloads from local, current data.	Replicate existing vectors between sites.	Health monitoring and evidence gathering across deployments.
Feeds to Kafka, search, and lakehouse platforms	Change events supply event-driven AI and stream analytics.	Deliver vectors to search engines and vector-capable stores.	Pipeline status and report review.

Four Reference Patterns for Vector Data

Migrate existing vectors. GoldenGate initial load moves historical embeddings from sources such as PostgreSQL pgvector or MySQL HeatWave into Oracle AI Database, and CDC keeps them synchronized [48]. Keep the embedding model, dimensions, and numeric format the same on both sides.

Replicate or consolidate vectors. Embeddings produced in several regions or services replicate into one retrieval system, or between sites for local, low-latency access.

Stream vectors to AI search engines. GoldenGate delivers vector changes to Elasticsearch [4], [11], to Kafka through GoldenGate for Distributed Applications and Analytics [5], [6], [48], and to other supported targets.

Enrich event streams. GoldenGate Stream Analytics adds filtering, model scoring, and vector operations to GoldenGate-delivered streams [7], [8].

One capture can serve all of these consumers. Each consumer keeps its own Replicat, checkpoint, and lag measurement.

The GoldenGate AI Service

The GoldenGate AI Service is a microservice in the GoldenGate 26ai deployment that connects Replicat to embedding models [9], [10]. It is available from release update 23.26.1 and holds connections to local and remote models. Replicat calls a model while it applies or loads each change and writes the returned vector in the same target transaction as the row. Operational data and its AI-ready representation stay synchronized, with fewer moving parts than a separate embedding workflow after replication [11]. The service generates embeddings today and is the planned home for LLM functions across other GoldenGate microservices [10].



One Mapping Line

Embedding generation is one line in a Replicat mapping [12], [13]. The line names the operation, the model, and the source text column:

```
MAP GG_AI.PRODUCT_CATALOG, TARGET GG_AI.PRODUCT_CATALOG,  
COLMAP (USEDEFAULTS,  
  description_vector = @AISERVICE(  
    embed, 'text-embedding-3-small', product_description));
```

The line is the same wherever the model runs. What changes is how the model is made available to the AI Service, and GoldenGate offers three ways to do that. Table 3 summarizes them.

Three Ways to Supply the Model

Table 3. Supplying a model to the AI Service.

Variation	Where inference runs	Setup	Choose it when
Self-managed deployment, hosted provider	Provider's service	Proxy, CA certificate, provider, and model in Service Manager	You run GoldenGate yourself and want a hosted model.
OCI GoldenGate, AI Model connection	Provider's service	Create an AI Model connection and assign it to the deployment	You use the managed service.
Self-managed deployment, local ONNX model	Inside the GoldenGate deployment	Copy the model and tokenizer into the models directory	Data must stay local, or the site is air-gapped.

Self-Managed Deployment with a Hosted Provider

In a self-managed deployment, an administrator registers the provider once in Service Manager, and every Replicat in the deployment can then call its models [10]. GoldenGate works with OCI Generative AI, OpenAI, Cohere, Gemini, Hugging Face, and custom endpoints that follow the OpenAI API. Setup takes five short steps in Service Manager and the deployment:

1. Configuration: set HTTP_PROXY and NO_PROXY when the provider sits outside the corporate network.
2. Certificates: add the provider's CA certificate, retrieved with `openssl s_client -showcerts -connect <server_name>:443`.
3. AI: define the provider with its base URL and API key, then name the embedding model. OCI Generative AI also takes the tenancy, compartment, and user OCIDs and the key fingerprint.
4. Services: ensure the AI Service is listed as running.



5. Deployment, AI Models: ensure the model listed and ready for Replicat.

OCI GoldenGate with an AI Model Connection

OCI GoldenGate delivers the same capability as a fully managed service [11], [14], however providing the AI Model Connection follows a different path. In OCI GoldenGate an administrator creates an AI Model connection, which pairs a provider with a specific embedding model, and assigns it to the deployment [15]. AI Model connections support OCI Generative AI, OpenAI, Google Gemini, and Voyage AI. Once the connection is assigned, the AI Service generates embeddings during replication with the same @AISERVICE line. The OCI quickstart walks through the full flow [16].

Self-Managed Deployment with a Local ONNX Model

From release update 23.26.3, the AI Service can run an ONNX embedding model inside the GoldenGate deployment [17]. Copy the model file and its tokenizer.json into the Service Manager models directory:

```
$OGG_HOMEfor the ServiceManager/  
  var/  
    lib/  
      models/  
        my-model.onnx  
        tokenizer.json
```

Service Manager discovers the model automatically, with no provider entry, endpoint, or API key to manage. Inference stays inside your environment, which suits regulated data, air-gapped sites, and pipelines that call for low inference latency.

Targets and Vector Columns

The AI Service works with supported targets that accept vector data. OCI GoldenGate names Oracle AI Database 26ai, Oracle Autonomous AI Database, PostgreSQL, Snowflake, and Elasticsearch [11]. On Oracle AI Database, declare the target column with the native VECTOR type [18], using the dimensions and numeric format of the chosen model, for example VECTOR(1536, FLOAT32) for the default output of text-embedding-3-small. Give each model its own column and record the model name with the data so that every consumer queries a column with the model that produced it.

Other Ways to Deliver Vectors

Two further options complement the AI Service. When the source already stores vectors, GoldenGate replicates the VECTOR column along with the row, as the reference patterns describe [19]. When embedding logic should run inside the target database, the @DBFUNCTION mapping, available since

GoldenGate 23.4.1, calls a database function during apply, and a scheduled database job can embed rows after apply. See Appendix for both database-side options.

Register a model once, add one mapping line, and every replicated change arrives with its embedding.

Embedding Generation Example

This walkthrough extends an existing GoldenGate deployment for an Oracle-to-Oracle topology, however the functionality will work for any target endpoint that allows vector data to be stored in it. GoldenGate replicates a product catalog, and a Parallel Replicat calls the AI Service to generate a vector from each product description as the change is applied. A semantic search on the target then combines the vector with current price and stock in a single query. Seven steps take you from an empty schema to that query.

The walkthrough uses Parallel Replicat because it maps records and applies independent transactions concurrently [20], [21]. Figure 2 shows the pipeline.

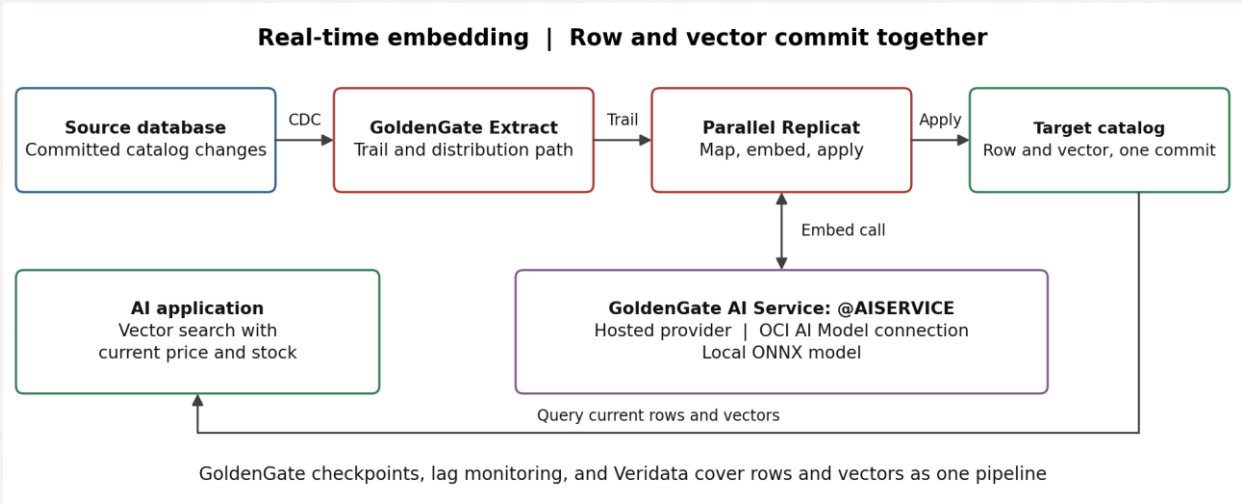


Figure 2. Real-time embedding during Parallel Replicat apply.

Prerequisites

Table 4 lists what to have in place before Step 1.

Table 4. Walkthrough prerequisites.

Prerequisite	Required state	Owner
Deployments	Oracle GoldenGate 26ai Microservices deployments for source and target at release update 23.26.3 or later, or an OCI GoldenGate deployment, with Administration, Distribution, and Receiver services running.	GoldenGate administrator

Prerequisite	Required state	Owner
Databases	Source Oracle Database prepared for GoldenGate capture; Oracle AI Database 26ai target. A GoldenGate administrator user and credential alias on each. Refer to the GoldenGate documentation to ensure your database is ready for Extract.	Database administrator
Schema	GG_AI schema on both databases with appropriate supplemental logging.	Database administrator
Embedding model	One of the three options in Step 2.	GoldenGate administrator

Step 1: Create the Catalog Table

Create the table as the GG_AI user on both databases. The target adds one vector column sized for the model you choose in Step 2. Seed two products on the source.

```
-- Source and target
CREATE TABLE product_catalog (
  product_id      NUMBER PRIMARY KEY,
  product_name    VARCHAR2(200) NOT NULL,
  category        VARCHAR2(100) NOT NULL,
  product_description VARCHAR2(1000) NOT NULL,
  price           NUMBER(10,2) NOT NULL,
  stock_qty       NUMBER NOT NULL
);

-- Target only
ALTER TABLE product_catalog
  ADD (description_vector VECTOR(1536, FLOAT32));

-- Source only
INSERT INTO product_catalog VALUES
  (1001, 'EcoClean Dish Soap', 'Cleaning',
   'Plant-based dish soap for everyday washing.', 5.99, 30);
INSERT INTO product_catalog VALUES
  (1002, 'FloorShine', 'Cleaning',
   'Hard-floor cleaner with a fresh citrus scent.', 8.49, 20);
COMMIT;
```

Step 2: Make a Model Available

Pick the option that matches your deployment. Each one ends with a model name that Replicat can call.

- Hosted provider, self-managed: in Service Manager, set the proxy, add the provider certificate, and define the provider and the model text-embedding-3-small, as the previous section describes [10].
- OCI GoldenGate: create an AI Model connection for your provider and embedding model, and assign it to the deployment [15], [16].
- Local ONNX model, self-managed: copy the model file and tokenizer.json into ServiceManager/var/lib/models and let Service Manager discover it [17]. Size the vector column in Step 1 for that model's output.

The model now appears in the deployment's list of AI models.

Step 3: Capture Catalog Changes

In Admin Client, connect to the source deployment. Log all columns of the catalog so that every update record carries the description that the model reads. Then create the Extract, its trail, and the distribution path to the target deployment [22]. Host names, ports, and credential aliases are examples.

```
-- Source deployment
DBLOGIN USERIDALIAS gg_source DOMAIN OracleGoldenGate
ADD TRANDATA GG_AI.PRODUCT_CATALOG, ALLCOLS
ADD HEARTBEATABLE

ADD EXTRACT EPROD INTEGRATED TRANLOG BEGIN NOW
REGISTER EXTRACT EPROD DATABASE
ADD EXTTRAIL ea, EXTRACT EPROD

-- EPROD.prm
EXTRACT EPROD
USERIDALIAS gg_source DOMAIN OracleGoldenGate
EXTTRAIL ea
LOGALLSUPCOLS
TABLE GG_AI.PRODUCT_CATALOG;

-- Enter ADD DISTPATH as one line
ADD DISTPATH DPROD
  SOURCE trail://source-host:9002/services/v2/sources?trail=ea
  TARGET wss://target-host:9103/services/v2/targets?trail=da

START EXTRACT EPROD
START DISTPATH DPROD
```

Step 4: Instantiate the Target with the Same Mapping

With EPROD running, record the current SCN on the source. Run a GoldenGate initial load of the catalog as of that SCN [2], [3], [23], [33], and apply it with a Replicat that carries the MAP statement from Step 5. Historical rows and live changes are then embedded by the same model and the same line of configuration [10]. For very large tables, a database-native bulk load such as Oracle Data Pump [24], [25], followed by a bulk embedding pass, is the alternative.

```
-- Source: record the snapshot SCN as <S>
SELECT current_scn FROM v$database;

-- ELOAD.prm: initial-load Extract, reading as of the snapshot
EXTRACT ELOAD
USERIDALIAS gg_source DOMAIN OracleGoldenGate
EXTFILE il
TABLE GG_AI.PRODUCT_CATALOG, SQLPREDICATE 'AS OF SCN <S>';

ADD EXTRACT ELOAD, SOURCEISTABLE
START EXTRACT ELOAD
```

Deliver the load file to the target deployment over a distribution path and apply it before starting RPROD. Capture started before the snapshot, so every change committed after the SCN is already in the trail.

Create a separate distribution path for the initial load. This example reads the source il files and writes target trail li, keeping the initial-load data separate from the CDC trail da.

Enter the ADD DISTPATH command on one line:

```
ADD DISTPATH DLOAD SOURCE trail://source-host:9002/services/v2/sources?trail=il TARGET
wss://target-host:9103/services/v2/targets?trail=li
START DISTPATH DLOAD
```

Use the same connection security and authentication configuration as the existing distribution path. Verify that all initial-load data has reached the target, then stop the initial-load distribution path:

```
STOP DISTPATH DLOAD
```

Complete extraction and delivery before launching RLOAD.

On the target host, create rload.prm in the target deployment's parameter directory. Replace <target-trail-directory> with the directory containing the delivered li trail files.

```
SPECIALRUN
END RUNTIME
USERIDALIAS gg_target DOMAIN OracleGoldenGate
EXTFILE <target-trail-directory>/li

MAP GG_AI.PRODUCT_CATALOG, TARGET GG_AI.PRODUCT_CATALOG,
COLMAP (
```

```

USEDEFAULTS,
description_vector = @AIService(
  embed, 'text-embedding-3-small', product_description)
);

```

Use the same model name and mapping that RPROD will use in Step 5. If Step 2 selected a different model, substitute its registered name and ensure the target vector column has the corresponding dimensions.

From a shell initialized with the target deployment's GoldenGate and database environment, execute:

```

"$OGG_HOME/bin/replicat" \
  PARAMFILE "$OGG_CONF_HOME/ogg/rload.prm" \
  REPORTFILE "$OGG_VAR_HOME/lib/report/RLOAD.rpt"

```

Here, OGG_CONF_HOME and OGG_VAR_HOME must identify the target deployment. Its credential store and AI model configuration must be available to the process.

SPECIALRUN performs a one-time load without checkpoints. Historical rows receive their embeddings as RLOAD inserts them into the target.

Or if using OCI GoldenGate, which does not have access to the operating system shell, you can use the following method:

Connect AdminClient to the target deployment. Log in to the target database and create a checkpoint table dedicated to the initial load. Run ADD CHECKPOINTTABLE only if this table does not already exist.

```

DBLOGIN USERIDALIAS gg_target DOMAIN OracleGoldenGate
ADD CHECKPOINTTABLE ggadmin.ggs_load_checkpoint

```

Create RLOAD as a classic, non-integrated Replicat reading the initial-load trail 1i delivered in item 3:

```

ADD REPLICAT RLOAD, EXTTRAIL 1i, CHECKPOINTTABLE ggadmin.ggs_load_checkpoint
EDIT PARAMS RLOAD

```

Save the following parameter file:

```

REPLICAT RLOAD
USERIDALIAS gg_target DOMAIN OracleGoldenGate

MAP GG_AI.PRODUCT_CATALOG, TARGET GG_AI.PRODUCT_CATALOG,
COLMAP (
  USEDEFAULTS,
  description_vector = @AIService(
    embed, 'text-embedding-3-small', product_description)
);

```

Use the same registered model and mapping that RPROD will use in Step 5. If Step 2 selected a different model, substitute its registered name and ensure the target vector column has the corresponding dimensions.

This configuration obtains its input trail from ADD REPLICAT. Omit SPECIALRUN, END RUNTIME, and EXTFILE from the parameter file.

Once the initial load has started and there are trail files on the target deployment, you can start the RLOAD replicat.

```
START REPLICAT RLOAD
```

Monitor its status and progress:

```
SEND REPLICAT RLOAD, STATUS
SEND REPLICAT RLOAD, LOGEND
```

Repeat the checks until LOGEND confirms that all available records have been processed. Because delivery has already completed, this marks the end of the initial-load input. RLOAD generates embeddings as it inserts the historical rows.

Stop RLOAD cleanly and review its report:

```
STOP REPLICAT RLOAD
VIEW REPORT RLOAD
```

Step 5: Apply Changes with Embeddings

Connect Admin Client to the target deployment. Create a checkpoint table and a Parallel Replicat that reads the delivered trail [22]. The mapping adds one line to an ordinary catalog MAP: the @AIService call that fills DESCRIPTION_VECTOR.

```
-- Target deployment
DBLOGIN USERIDALIAS gg_target DOMAIN OracleGoldenGate
ADD CHECKPOINTTABLE ggadmin.ggs_checkpoint
ADD HEARTBEATABLE
ADD REPLICAT RPROD, PARALLEL, EXTTRAIL da,
    CHECKPOINTTABLE ggadmin.ggs_checkpoint

-- RPROD.prm
REPLICAT RPROD
USERIDALIAS gg_target DOMAIN OracleGoldenGate
MAP GG_AI.PRODUCT_CATALOG, TARGET GG_AI.PRODUCT_CATALOG,
COLMAP (USEDEFAULTS,
    description_vector = @AIService(
        embed, 'text-embedding-3-small', product_description));

-- Start after the snapshot SCN recorded in Step 4
START REPLICAT RPROD, AFTERCSN <S>
```

AFTERCSN begins apply with the first transaction committed after the snapshot SCN, so the loaded rows and the change stream meet exactly at that point [26]. Later restarts use the saved checkpoint: START REPLICAT RPROD.

Step 6: Change the Catalog and Watch the Vectors Follow

Run four changes on the source, one at a time, and query the target after each.

```
-- Source
INSERT INTO product_catalog VALUES
  (1003, 'BabySafe Dish Soap', 'Cleaning',
   'Gentle fragrance-free soap for baby bottles.', 6.99, 12);
COMMIT;

UPDATE product_catalog
  SET product_description =
    'Fragrance-free dish soap for baby bottles and sensitive skin.'
  WHERE product_id = 1003;
COMMIT;

UPDATE product_catalog SET stock_qty = 7 WHERE product_id = 1003;
COMMIT;

DELETE FROM product_catalog WHERE product_id = 1003;
COMMIT;

-- Target, after each change
SELECT product_id, product_description, stock_qty,
       VECTOR_DIMENSION_COUNT(description_vector) AS dims
FROM product_catalog WHERE product_id = 1003;
```

You should see:

- Insert: row 1003 arrives with a 1,536-dimension vector; VECTOR_DIMENSION_COUNT confirms the dimension count [27].
- Description update: the new description and a new vector arrive in the same commit.
- Stock update: stock_qty is 7, and the vector continues to match the description.
- Delete: row 1003 and its vector are removed together.

Step 7: Search with Current Context

The application embeds the search phrase with the same model and passes the result as a bind variable. The query ranks products by cosine distance [28] and filters on live stock.

```

SELECT p.product_id, p.product_name, p.price, p.stock_qty,
       VECTOR_DISTANCE(p.description_vector, :query_vector, COSINE)
       AS distance
FROM product_catalog p
WHERE p.stock_qty > 0
ORDER BY distance
FETCH FIRST 3 ROWS ONLY;

```

Run it before the delete in Step 6 with the phrase “fragrance-free soap for sensitive skin”. You should see product 1003 at the top of the result with its current price and stock. Semantic meaning and structured facts come from one row, replicated and embedded by one pipeline.

Operating the Pipeline

Embedding during replication keeps operations simple: one pipeline to monitor, one checkpoint to recover from, and one place to tune, with the practices GoldenGate administrators already use.

Monitor Lag and Freshness

Admin Client reports the state of each process, and the heartbeat table created in Steps 3 and 5 measures end-to-end lag from source commit to target apply [22]. Table 5 lists the measures that matter.

```

INFO REPLICAT RPROD
SEND REPLICAT RPROD, GETLAG
STATS REPLICAT RPROD, TOTAL

-- Target database: end-to-end lag from the heartbeat
SELECT * FROM ggadmin.gg_lag;

```

Table 5. What to measure.

Measure	How	Use it to
Apply lag	SEND REPLICAT RPROD, GETLAG and the heartbeat lag view.	Set an alert threshold well inside trail retention.
Apply throughput	STATS REPLICAT RPROD, TOTAL sampled over time.	Compare change rate before and after enabling embedding.
Model latency	Provider latency and error rate for a hosted model, or host CPU for a local ONNX model, read beside Replicat lag.	Decide whether to add appliers or move the model closer.
Search freshness	Commit a tagged test product on the source and time its appearance in the Step 7 query.	Report the number the AI application cares about.

Real Time or Batch

Embed in real time when freshness drives the AI result: search, recommendations, and agents that act on current state. Use the initial load, or a scheduled batch, for historical backfill, model reprocessing, and data that changes slowly. Both share one model and one vector column, so a table can start with a bulk load and stay current through replication.

Recover with Checkpoints

Replicat records its position in the checkpoint table as it applies transactions. After any interruption it resumes from that position, so each change is applied, and embedded, in order. Table 6 covers the common conditions.

Table 6. Recovery by condition.

Condition	What GoldenGate does	What you do
Replicat or host restart	Resumes from the checkpoint and reads the retained trail.	START REPLICAT RPROD and watch lag return to normal.
Model or provider unavailable	Replicat waits or stops at the change that calls the model, according to its error handling. Capture and the trail continue.	Restore the provider, credentials, or model file, then restart. The backlog drains in order.
Changed embedding input	Capture and apply continue for the existing columns.	Update the mapping, then re-embed affected rows with an initial load of that table.

Scale Apply

Parallel Replicat reads and maps trail records with mapper threads and applies independent transactions with applier threads [21]. MAP_PARALLELISM sets the number of mappers. APPLY_PARALLELISM, or the MIN_APPLY_PARALLELISM and MAX_APPLY_PARALLELISM pair, sets the appliers. Tune mapper and applier parallelism against measured embedding latency and target apply throughput; provider quotas or local-model CPU can limit overall throughput.

The following three practices keep throughput high [29]. Place the model endpoint close to the deployment, in the same region or network. Plan provider quotas for peak change rates. Reuse embeddings for identical input where the application allows. Where price and stock change far more often than product text, the scheduled job (see Appendix) embeds only when the text changes.

Change the Model

A new model produces a new representation, so give it a new column and switch when it is ready:

1. Register the new model in the AI Service and add a vector column sized for it.
2. Add a second @AIService line to the mapping so new changes fill both columns, then embed existing rows for the new column with an initial load.
3. Point the application's query model and query column to the new pair in one release.
4. Remove the old column and mapping line after the agreed fallback window.

Three Measures of a Healthy Pipeline

Three measures together show the pipeline is doing its job. Compare source and target rows with Oracle GoldenGate Veridata for replication fidelity. Count rows with a vector for embedding coverage. Run a small set of labeled search phrases and review the ranked products, with their current price and stock, for retrieval quality.

Secure the Data Path

Table 7 lists the controls for the embedding path.

Table 7. Access and data controls.

Control	Practice
Embedding input	Approve which columns are embedded. A local ONNX model keeps the text inside the GoldenGate deployment; a hosted model sends it to the approved provider and region.
Credentials	Provider keys live in the AI Service configuration or the OCI AI Model connection.
Retrieval access	Apply row-level policy for tenants and users.
Transport and storage	Encrypt trails and network paths [30], [31], and enable encryption on the target database.
Deletes	A source delete removes the target row and its vector in one transaction.

Using AI to Operate GoldenGate

The AI wave that depends on current data can also help run the pipelines that deliver it. GoldenGate Microservices Architecture is administered through REST APIs [32], which makes it a natural fit for AI agents: an agent can inspect processes, read reports, propose a plan, and carry out authorized steps. This section describes what is possible today, a pattern for doing it well, and three exercises. Each exercise keeps the administrator in the approving role.

What Is Possible Today

An AI agent reaches a GoldenGate environment through three interfaces [33]:

- The GoldenGate MCP server exposes tools for inspecting and administering replication components, including process status, lag, reports, and statistics [34], [35]. MCP, the Model Context Protocol, is the interface through which an AI tool discovers and invokes operations on external systems.
- The Oracle Database MCP server, delivered through SQLcl, runs database queries and SQL operations.
- The GoldenGate REST APIs run Microservices operations directly, including options beyond the current MCP tool set [32]. A status request is a single call: `GET /services/v2/replicats/RPROD/info/status`.

With those interfaces an agent takes on work that fills an administrator's day:

- Health and lag: read process status, lag, and statistics across deployments and summarize the exceptions.
- Error triage: summarize report and discard files and propose the next step.
- Configuration review: compare parameter files and proposed changes with the documented options.
- Procedures: follow a runbook for instantiation, re-instantiation, or migration, with authorization at each change.
- Records: produce run reports and runbook-style summaries.

Oracle's published lab used Codex as the agent, and other MCP-capable clients such as Claude or GitHub Copilot can be evaluated in the same way [33]. The GoldenGate MCP server is an open-source project for prototyping, development, and testing [35]. Teams that prefer to keep an agent away from live systems can give it exported status, lag, and report files and receive the same analysis offline.

The Runbook Pattern

The most effective way to direct an agent is a Markdown runbook. The runbook captures operational knowledge in a form that people can review and an agent can follow. It defines how a procedure runs:

what to check, the order of operations, the conditions for proceeding, and when to stop [33]. A complete runbook holds six things:

- Objective and scope: the environments and tables included, and the components to preserve.
- Prerequisites: connectivity, privileges, capture coverage, and trail retention.
- Operational sequence: the ordered steps of the procedure.
- Criteria for proceeding: the evidence that completes each step.
- Failure handling: when to hold processes and call for review.
- Execution records: parameters, counts, checkpoints, and final status.

A skeleton is enough to start, and it grows with each run:

```
# Runbook: catalog re-instantiation
## Objective and scope
## Prerequisites
## Sequence          (numbered steps, one interface named per step)
## Criteria for proceeding
## Failure handling
## Execution record  (SCN, counts, checkpoints, final status)
```

Table 8 lists who does what, and Figure 3 shows how the parts connect.

Table 8. Roles in an agent-driven procedure.

Component	Role
Markdown runbook	Documents the strategy, steps, boundaries, and validation criteria.
AI agent	Reads the runbook, inspects the environment, prepares a plan, coordinates authorized actions, and analyzes responses.
Oracle Database MCP server (SQLcl)	Runs database queries and SQL operations.
GoldenGate MCP server	Inspects and administers replication components.
GoldenGate REST APIs	Run Microservices operations, including options outside the MCP tool set.
Administrator	Authorizes the plan and each operational change, and reviews the evidence.



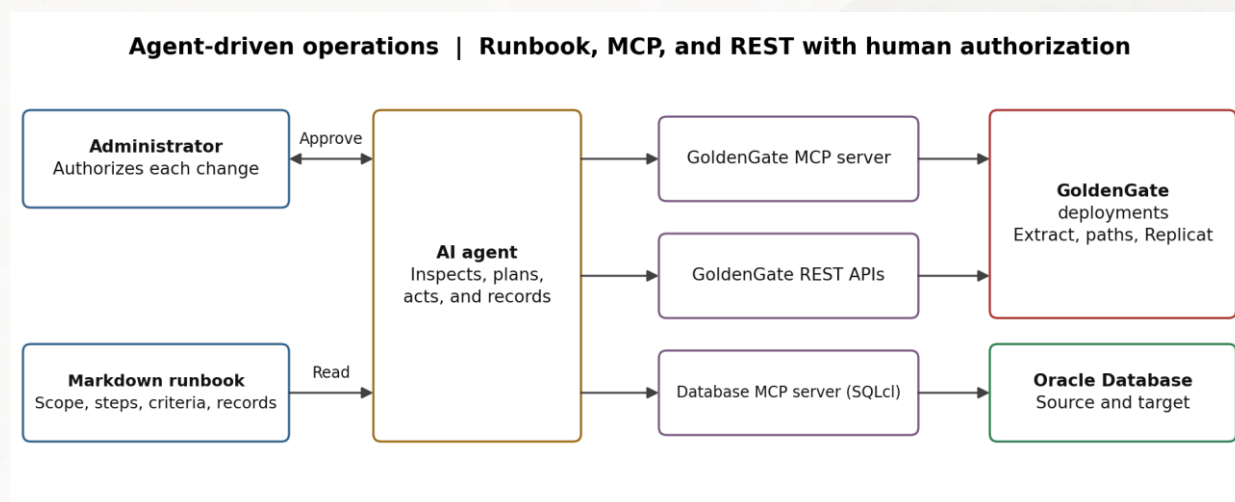


Figure 3. Agent-driven operations with a runbook, MCP, and REST.

Each run starts from explicit, reviewable instructions, and the agent determines the values for that run from the live environment: table list, SCN, and process names. Lessons from a run, such as an extra check or an interface detail, go back into the runbook, so the procedure improves each time it is used. The runbook also sets the boundary of the agent's work: everything outside its scope stays untouched.

Exercise 1: Runbook-Guided Initial Load and CDC Resumption

Oracle GoldenGate subject matter experts have automated one of the more demanding GoldenGate procedures this way: a parallel initial load followed by resumption of CDC from a consistent point, Oracle to Oracle, on GoldenGate 23.26.3 with an Autonomous Database target [33]. It is the same handoff as Steps 4 and 5 of the walkthrough, a load at a recorded SCN followed by **START REPLICAT** with **AFTERCSCN**, carried out by an agent. The agent worked through nine steps:

1. Inspect the environment: review connections, processes, tables, sizes, and mappings.
2. Prepare replication: stop the CDC Replicat, preserve its parameters and checkpoint, and keep Extract capturing.
3. Set the reference point: review capture coverage and open transactions, select an SCN, and record source counts.
4. Extract in parallel: create four **SOURCEISTABLE** Extracts with non-overlapping ranges, all reading at the same SCN [3], [33].
5. Prepare the target: confirm that every extraction file is complete, then truncate the tables in scope.
6. Apply in parallel: run four Parallel Replicats, each with four appliers and **BATCHSQL**.
7. Validate: compare counts by table and range, and review errors, discards, and checkpoints.



8. Resume CDC: start the Replicat with AFTERCSN and monitor the backlog as it drains.
9. Complete: remove that run's temporary processes and files, and produce the final report.

In the documented run, the workflow loaded about 5.1 million records, validated the counts, resumed CDC, and removed its temporary components. One practical finding went straight into the runbook: in the version tested, creating the initial-load Extract and resuming with AFTERCSN were done through direct REST calls, because those options sat outside the MCP tools in use. MCP and REST work together, and the runbook records which interface each step uses.

For a replication team, the value is in the coordination. Re-instantiation touches the database, capture, several temporary processes, and apply, and each handoff has a condition to meet before the next step begins. The agent tracks those conditions, gathers the evidence, and keeps the record, while the administrator decides. A procedure that once depended on one expert's memory becomes a reviewed document that any qualified administrator can run.

A request to the agent follows this pattern:

“Follow the re-instantiation runbook. Inspect the environment and the prerequisites first, then show me your plan for the parallel initial load. Keep CDC capture running and leave everything outside the scope untouched. Wait for my approval before any operational change. When you finish, report the validation evidence and the final replication status.”

The approach extends to Db2, SQL Server, MySQL, and other supported platforms, with the runbook adapted to each database's consistency and positioning mechanism [33].

Exercise 2: Diagnose Apply Lag

Lag diagnosis is a read-only task and a good first assignment for an agent. Give it an evidence set for one matched time window:

- Extract and Replicat status and lag, from Admin Client or the MCP tools.
- Report files and change statistics for EPROD and RPROD.
- Parameter files for both processes.
- Model latency from the provider, or CPU data from the GoldenGate host for a local ONNX model.
- A baseline from before embedding was enabled.

Illustrative prompt: “Review these timestamped EPROD and RPROD exports. Read only. Separate facts from hypotheses; assess capture, transport, target apply, and embedding. Explain what evidence is missing. Propose one bounded diagnostic change with its owner, stop limit, rollback, and validation.”

Table 9 lists three common observations and what each one indicates.

Table 9. Lag-diagnosis observations.

Observation to establish	Interpretation and next check
RPROD lag rises; provider latency or throttling rises	The hosted model is the likely limit. Confirm in the Replicat report and the provider console, then review quotas and endpoint placement.
RPROD lag rises; GoldenGate host CPU is saturated (local ONNX model)	Local inference is the likely limit. Compare apply throughput with the baseline, then test more appliers or more host capacity.
RPROD lag rises; model latency is flat	Look upstream to capture lag, distribution path status, and change mix. Embedding is keeping pace.

A useful response contains timestamped facts and their sources, ranked hypotheses, missing evidence, one proposed test, and the result that would confirm or refute it. A bounded test is to raise MAX_APPLY_PARALLELISM by a fixed step for an agreed interval, with the administrator setting the limit at which it stops. Compare change rate, lag, and search freshness before, during, and after, and keep the change when the numbers improve inside the resource budget. The same evidence set serves a person and an agent alike, so the exercise also leaves the team with a reusable checklist for the next lag investigation.

Exercise 3: Review a Proposed Change

Give the agent the Replicat type, parameter file, resource baseline, and two proposals: “add THREADRANGE to the catalog MAP” and “switch to a different embedding model.” Ask it to identify applicable controls, dependencies, risks, and acceptance tests. A good review points out that THREADRANGE belongs to Coordinated Replicat [36], [37], that Parallel Replicat scales through its parallelism parameters [21], and that a different model produces a different representation and therefore calls for a new vector column.

Illustrative prompt: “Review the proposed changes against this catalog topology. Identify the settings that apply to this Replicat type. Recommend at most one change for a controlled test, with the evidence needed, the approval owner, the rollback, and the measures to compare before and after.”

The administrator compares the recommendation with the deployed release, records a diff and an approved baseline, and tests one change at a time. Accept the change when apply lag and search freshness improve and the Step 6 results still hold. The observed outcome is kept with the recommendation as the acceptance evidence. Change review is where an agent's breadth helps most: it reads the whole parameter file and the reference documentation together and returns the few points that deserve an expert's attention.

Operating Safely

- Give the agent a dedicated identity with least privilege, using GoldenGate's role-based access [38]. Start with read access.
- Expose an approved list of tools, and pin and review the MCP server revision you run.
- Require authorization for the plan and for each operational change.
- Keep secrets in the credential store, and treat report and configuration text as evidence to analyze.
- Record the runbook version, tool versions, timestamps, prompts, recommendations, and human decisions with each run.

With these practices in place, administrators keep control of every decision while the agent takes on the inspection, coordination, and record keeping that consume most of the time in a complex procedure.

Where to Start

Adoption works best in three stages, each building trust for the next:

1. Analyze offline. Give the agent exported status, lag, and report files from a real incident and compare its analysis with what your team concluded.
2. Connect read-only. Let the agent query live status, lag, and reports through MCP or REST with a read-only identity, and use it for daily health reviews and lag diagnosis.
3. Automate a procedure. Write the runbook for one procedure your team repeats, such as a re-instantiation. Run it with the agent in a test environment, and move it to production once the evidence from several runs is consistent.

Each stage produces records that the next one builds on. The runbook becomes a shared asset: the team's operating knowledge in a form that both people and agents can use, and a natural companion to the GoldenGate procedures your team already documents.

GoldenGate Data Fabric and the AI Truth Layer

The catalog pipeline is a small instance of a larger idea. AI systems are only as good as the operational context they receive, and that context is trustworthy when it comes from committed transactions, arrives while it is still current, and can be traced to its source. Oracle GoldenGate Data



Fabric applies that idea across the enterprise. It captures committed change once and continuously turns it into governed, reusable data products for operational, analytical, and AI consumers, wherever they run. That role is the AI Truth Layer for distributed operational data.

Every technique in this paper is a building block of that layer, as Table 10 shows:

Table 10. From this paper to GoldenGate Data Fabric.

Shown in this paper	Extended by GoldenGate Data Fabric
One capture feeds the replicated catalog and its vectors.	Capture once, activate many: the same committed change maintains standby copies, application streams, lakehouse tables, and vector products.
An initial load at a recorded SCN, then continuous apply.	ZeroETL pipelines coordinate the initial copy with the handoff to continuous change [39].
One @AISERVICE line that calls a model through the AI Service.	Pipelines call customer-configured model endpoints under policy controls, so vector search reflects recently committed transactions.
A hand-written Replicat mapping.	Recipes and the Workbench package proven replication designs as reusable templates, with AI-assisted pipeline design under human review.
Lag, checkpoints, and Veridata as evidence that rows and vectors are current.	The Dataset: a durable product record that carries freshness, quality, lineage, and run evidence with the data.
An agent that follows a runbook and waits for authorization.	AI-assisted development at three levels of control, from natural-language prompting to a pro-code API, with people keeping approval of meaning, access, and release.

The three Data Fabric outcomes map directly onto the walkthrough. Operate: the replica stays synchronized through planned and unplanned change. Activate: current rows and vectors reach the AI application as soon as they commit. Optimize: the choice of model, of where inference runs, and of vector target stays with the customer.

Start with one pipeline and one AI consumer. Each product you add reuses the capture you already run.



Conclusion

AI is only as current as the data behind it, and GoldenGate already moves that data. Register a model with the GoldenGate AI Service, add one @AISERVICE line to a Replicat mapping, and each committed change arrives on the target with its embedding in the same transaction.

The model can run with a hosted provider, through an AI Model connection in OCI GoldenGate, or as a local ONNX model inside your own deployment. Parallel Replicat provides the scale, and checkpoints, heartbeat lag, and Veridata provide the evidence that rows and vectors are current. AI agents, guided by runbooks and working through MCP and REST, then take on the procedures and diagnostics of running replication, with every change authorized by a person.

These patterns are the working parts of an AI Truth Layer. GoldenGate Data Fabric carries them from one pipeline to the whole estate, so that every AI answer and agent action starts from committed, current, traceable data.

Appendices

Database-side Options

Two options place embedding logic in the target database. Both use a model loaded into Oracle AI Database [40] and a short function that wraps VECTOR_EMBEDDING [41].

```
CREATE OR REPLACE FUNCTION create_product_vector(  
  p_name IN VARCHAR2, p_category IN VARCHAR2, p_description IN VARCHAR2  
) RETURN VECTOR IS  
  v VECTOR(384, FLOAT32);  
BEGIN  
  SELECT VECTOR_EMBEDDING(ALL_MINILM_L12_V2 USING  
    p_name || '. ' || p_category || '. ' || p_description AS data)  
    INTO v FROM dual;  
  RETURN v;  
END;  
/
```

Embedding During Apply with @DBFUNCTION

Available since GoldenGate 23.4.1, @DBFUNCTION maps a target column to a database function that runs as Replicat applies the row [42], [43], [44]:

```
product_vector = @DBFUNCTION(  
  GG_AI.CREATE_PRODUCT_VECTOR(  
    :PRODUCT_NAME, :CATEGORY, :PRODUCT_DESCRIPTION))
```

Scheduled Enrichment After Apply

A database job embeds only the products whose text changed, which suits catalogs where price and stock updates dominate. A SHA-256 hash of the text [45] identifies what each vector represents. Figure 4 shows the flow.

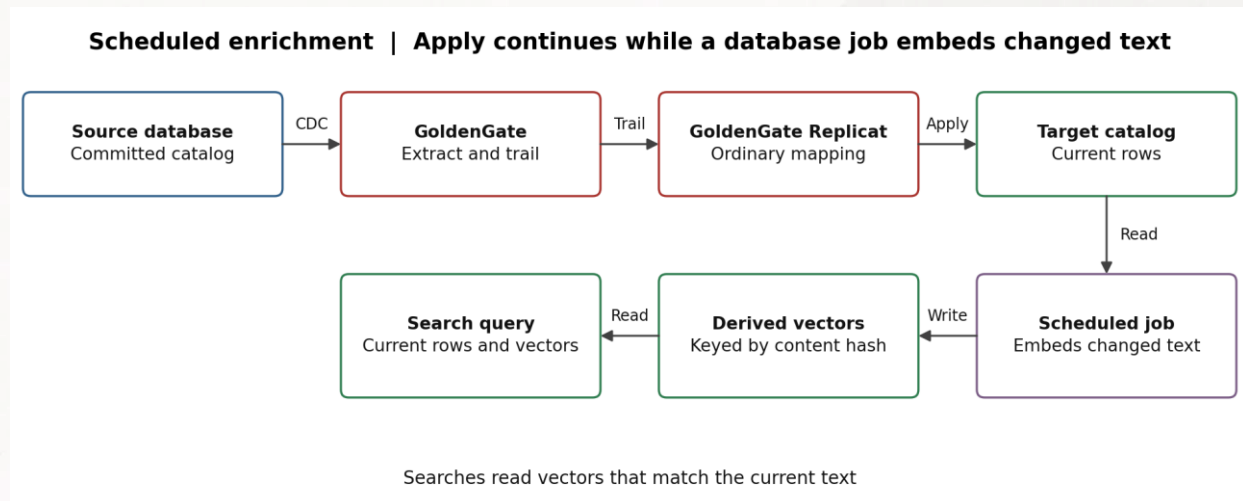


Figure 4. Scheduled enrichment after target commit.

```
CREATE TABLE product_embeddings (
  product_id NUMBER PRIMARY KEY, content_hash RAW(32) NOT NULL,
  embedding VECTOR(384, FLOAT32) NOT NULL);
```

```
CREATE OR REPLACE VIEW product_content AS
SELECT p.*, STANDARD_HASH(product_name || '|' || category || '|' ||
  product_description, 'SHA256') AS content_hash
FROM product_catalog p;
```

```
CREATE OR REPLACE PROCEDURE refresh_product_embeddings AS
BEGIN
  MERGE INTO product_embeddings e
  USING (SELECT c.product_id, c.content_hash,
    create_product_vector(c.product_name, c.category,
      c.product_description) AS v
    FROM product_content c
    LEFT JOIN product_embeddings x ON x.product_id = c.product_id
    WHERE x.product_id IS NULL
      OR x.content_hash <> c.content_hash) s
  ON (e.product_id = s.product_id)
  WHEN MATCHED THEN UPDATE SET
    e.content_hash = s.content_hash, e.embedding = s.v
  WHEN NOT MATCHED THEN INSERT (product_id, content_hash, embedding)
  VALUES (s.product_id, s.content_hash, s.v);
  DELETE FROM product_embeddings e WHERE NOT EXISTS
    (SELECT 1 FROM product_catalog p WHERE p.product_id = e.product_id);
  COMMIT;
END;
/
```

Schedule the procedure with DBMS_SCHEDULER [46], [47]. Applications join PRODUCT_CONTENT to PRODUCT_EMBEDDINGS on product and content hash, so that searches read vectors that match the current text.

Glossary

GoldenGate AI Service. The microservice in a GoldenGate 26ai deployment that connects Replicat to embedding models, local or remote.

@AIService. The column-mapping function that calls a model through the AI Service and returns a vector for the target column.

AI Model connection. The OCI GoldenGate connection type that pairs an AI provider with a specific embedding model and is assigned to a deployment.

Local ONNX model. An embedding model in ONNX format, with its tokenizer.json, that the AI Service runs inside the GoldenGate deployment.

Embedding. A numerical vector that represents the meaning of text. Vectors are comparable when the same model and input format produced them.

Parallel Replicat. A Replicat type that maps and applies independent transactions concurrently while keeping dependent transactions in order [21]. The walkthrough uses its nonintegrated mode.

Checkpoint and snapshot SCN. A checkpoint is the saved restart position of a process. The snapshot SCN is the source position of the initial load; START REPLICAT with AFTERCSN begins apply immediately after it.

Heartbeat table. GoldenGate tables and views that measure end-to-end lag from source commit to target apply.

MCP. Model Context Protocol, the interface through which an AI agent discovers and uses tools. Oracle publishes an open-source GoldenGate MCP server for development and testing [34].

Runbook. A Markdown document that defines a procedure for an agent and its reviewers: scope, prerequisites, sequence, criteria, failure handling, and records.

@DBFUNCTION. A column-mapping function, available since GoldenGate 23ai, that calls a function in the target database during apply. See Appendix.

AI Truth Layer. The role of GoldenGate Data Fabric for AI: answers and actions grounded in committed, current, traceable operational data.

Dataset and Recipe. In GoldenGate Data Fabric, a Dataset is the durable record of a governed data product, and a Recipe is a reusable replication design.

Implementation FAQ

How do I generate vector embeddings with Oracle GoldenGate?

Register an embedding model with the GoldenGate AI Service and add an @AIService line to the Replicat mapping. Replicat writes the row and its vector in one transaction.

Can GoldenGate create embeddings and keep my data in my own environment?

Yes. From release update 23.26.3, the AI Service runs a local ONNX model inside the GoldenGate deployment, so the text stays in your environment.

Does OCI GoldenGate support embedding generation?

Yes. Create an AI Model connection for OCI Generative AI, OpenAI, Google Gemini, or Voyage AI and assign it to the deployment.

How are historical rows embedded?

Run the initial load through GoldenGate at a recorded SCN with the same @AISERVICE mapping, then start Replicat after that SCN.

Can an AI agent administer GoldenGate?

An agent can inspect, plan, and carry out authorized steps through the GoldenGate MCP server and REST APIs, guided by a Markdown runbook. An administrator authorizes each change.

How does this relate to GoldenGate Data Fabric?

The pipeline in this paper is one governed, current data product built by hand. GoldenGate Data Fabric provides the design, governance, and reuse to build many of them, with GoldenGate as the AI Truth Layer for operational data.

Citations and References

- [1] Oracle, "Oracle GoldenGate 26ai Certification Matrix," Oracle, accessed October 17, 2026. <https://www.oracle.com/a/ocom/docs/ogg-26ai-cert-matrix.xlsx>
- [2] Oracle, "SOURCEISTABLE," Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/reference/sourceistable.html>
- [3] Nick Wagner, "Scaling GoldenGate SOURCEISTABLE Initial Load Extracts," Oracle Data Integration Blog, April 21, 2026. <https://blogs.oracle.com/dataintegration/scaling-goldengate-sourceistable-initial-load-extracts>
- [4] Oracle, "Add a Replicat for Elasticsearch Server," Oracle Cloud Infrastructure GoldenGate Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/cloud/paas/goldengate-service/ocigg/replicate/add-a-replicat-for-elasticsearch-server.html>
- [5] Madhu Kumar S, "GoldenGate REST API's Call to replicate the data to Oracle Cloud Infrastructure (OCI) Streaming and Apache Kafka," Oracle Data Integration Blog, August 9, 2023. <https://blogs.oracle.com/dataintegration/replicating-the-data-to-apache-kafka-using-goldengate-for-big-data-restapis-call>
- [6] Oracle, "Oracle GoldenGate for Distributed Applications and Analytics," Oracle Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/goldengate-daa.html>
- [7] Oracle, "GoldenGate Stream Analytics 26ai," Oracle GoldenGate Stream Analytics 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/stream-analytics/26/index.html>
- [8] Alex Kotopoulos, "GoldenGate Stream Analytics 26ai – Bringing GenAI, Vectors and AutoML to Streaming Pipelines," Oracle Data Integration Blog, October 14, 2025. <https://blogs.oracle.com/dataintegration/gg-stream-analytics-26ai-announcement>
- [9] Oracle, "Manage Vector Embedding with the AI Service," Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/ai-service.html>
- [10] Volker Kuhr, "Revolutionizing Data Replication: AI-Powered Vector Embeddings in Oracle GoldenGate 26ai," Oracle Data Integration Blog, January 29, 2026. <https://blogs.oracle.com/dataintegration/revolutionizing-data-replication-ai-powered-vector-embeddings-in-oracle-goldengate-26ai>

- [11] Julien Testut, “New OCI GoldenGate enhancements introduce AI-ready data pipelines and expanded connectivity,” Oracle Data Integration Blog, July 29, 2026. <https://blogs.oracle.com/dataintegration/new-oci-goldengate-enhancements-introduce-ai-ready-data-pipelines-and-expanded-connectivity>
- [12] Oracle, “@AIService,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/reference/aiservice.html>
- [13] Alex Lima, “Announcing Oracle GoldenGate 26ai smarter automation, broader compatibility, and simplified operations,” Oracle Data Integration Blog, January 29, 2026. <https://blogs.oracle.com/dataintegration/announcing-oracle-goldengate-26ai-smarter-automation-broader-compatibility-and-simplified-operations>
- [14] Oracle, “About AI with OCI GoldenGate,” Oracle Cloud Infrastructure GoldenGate Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/cloud/paas/goldengate-service/ocigg/ai-with-oci-goldengate.html>
- [15] Oracle, “Create an AI Model connection,” Oracle Cloud Infrastructure GoldenGate Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/cloud/paas/goldengate-service/ocigg/connections/create-ai-model-connection.html>
- [16] Oracle, “Use AI Models in OCI GoldenGate to Create Vector Embeddings,” Oracle Cloud Infrastructure GoldenGate Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/cloud/paas/goldengate-service/ocigg/quickstarts/use-ai-models-to-create-vector-embeddings.html>
- [17] Alex Lima, “Run AI Models Locally in Oracle GoldenGate with ONNX,” Oracle Data Integration Blog, August 4, 2026. <https://blogs.oracle.com/dataintegration/run-ai-models-locally-in-oracle-goldengate-with-onnx>
- [18] Oracle, “Vector Data Type,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/oracle/oracle-database/26/nfcoa/vector-data-type.html>
- [19] Oracle, “Vector Data Modeling,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/ai-vector-data-modeling.html>
- [20] Oracle, “Types of Replicat,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/replicat-types-replicat.html>
- [21] Oracle, “About Parallel Replicat,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/replicat-parallel-replicat.html>
- [22] Oracle, “Set Up a Unidirectional Data Replication Environment for Business Reporting,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/ggsol/undir-add-data-replication-business-reporting.html>
- [23] Volker Kuhr, “GoldenGate Instantiation methods,” Oracle Data Integration Blog, May 5, 2025. <https://blogs.oracle.com/dataintegration/goldengate-instantiation-methods>
- [24] Oracle, “Parameters Available in Data Pump Export Command-Line Mode,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/oracle/oracle-database/26/sutil/parameters-available-command-line-mode-export-oracle-data-pump.html>
- [25] Oracle, “Parameters Available in Oracle Data Pump Import Command-Line Mode,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/oracle/oracle-database/26/sutil/parameters-available-command-line-mode-import-oracle-data-pump.html>
- [26] Oracle, “START REPLICAT,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/gclir/start-replicat.html>
- [27] Oracle, “VECTOR_DIMENSION_COUNT,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. https://docs.oracle.com/en/database/oracle/oracle-database/26/sqlrf/vector_dimension_count.html
- [28] Oracle, “VECTOR_DISTANCE,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. https://docs.oracle.com/en/database/oracle/oracle-database/26/sqlrf/vector_distance.html
- [29] Oracle, “External AI Embedding with @AIService,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/external-ai-embedding-with-aiservice-function.html>
- [30] Oracle, “Secure Communication Using TLS and mTLS Support (Data in Transit),” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/secure-data-transit.html>
- [31] Oracle, “Trail File Encryption (Data at Rest),” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/secure-data-rest.html>
- [32] Oracle, “Retrieve Status,” Oracle GoldenGate 26ai REST API, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/oggra/op-services-version-replicats-replicat-info-status-get.html>



- [33] Marcio Carbonera, “Automating Complex Oracle GoldenGate Procedures with AI, MCP, and REST APIs,” Oracle Data Integration Blog, September 21, 2026. <https://blogs.oracle.com/dataintegration/automating-complex-oracle-goldengate-procedures-with-ai-mcp-and-rest-apis>
- [34] Oracle, “Oracle GoldenGate MCP Server,” GitHub repository oracle/mcp, accessed October 17, 2026. <https://github.com/oracle/mcp/blob/main/src/oracle-goldengate-mcp-server/README.md>
- [35] Julien Testut, “Enabling Agent-driven Operations in Oracle GoldenGate with MCP: An Open Source MCP Server,” Oracle Data Integration Blog, May 6, 2026. <https://blogs.oracle.com/dataintegration/enabling-agent-driven-operations-in-oracle-goldengate-with-mcp-an-open-source-mcp-server>
- [36] Oracle, “TABLE | MAP: THREADRANGE,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/reference/table-map.html>
- [37] Oracle, “About Coordinated Replicat,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/replicat-coordinated-replicat.html>
- [38] Oracle, “Oracle GoldenGate Microservices Roles,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/reference-goldengate-microservices-role-id.html>
- [39] Oracle, “Mirror data with ZeroETL mirror pipelines,” Oracle Cloud Infrastructure GoldenGate Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/cloud/paas/goldengate-service/ocigg/quickstarts/mirror-data-with-zeroetl-mirror-pipelines.html>
- [40] Oracle, “SQL Quick Start Using a Vector Embedding Model Uploaded into the Database,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/oracle/oracle-database/26/vecse/sql-quick-start-using-vector-embedding-model-uploaded-database.html>
- [41] Oracle, “VECTOR_EMBEDDING,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. https://docs.oracle.com/en/database/oracle/oracle-database/26/sqlrf/vector_embedding.html
- [42] Oracle, “@DBFUNCTION,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/reference/dbfunction.html>
- [43] Oracle, “Native Database Embedding with @DBFUNCTION,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/native-db-embedding-with-dbfunction.html>
- [44] Alex Porcescu, “Real-Time Embedding with Oracle GoldenGate: A Step-by-Step Guide,” Oracle Data Integration Blog, December 22, 2025. <https://blogs.oracle.com/dataintegration/real-time-embedding-with-oracle-goldengate-a-step-by-step-guide>
- [45] Oracle, “STANDARD_HASH,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. https://docs.oracle.com/en/database/oracle/oracle-database/26/sqlrf/STANDARD_HASH.html
- [46] Oracle, “DBMS_SCHEDULER,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. https://docs.oracle.com/en/database/oracle/oracle-database/26/arpls/DBMS_SCHEDULER.html
- [47] Oracle, “Scheduling Jobs with Oracle Scheduler,” Oracle AI Database 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/oracle/oracle-database/26/admin/scheduling-jobs-with-oracle-scheduler.html>
- [48] Oracle, “AI Service Use Cases,” Oracle GoldenGate 26ai Documentation, accessed October 17, 2026. <https://docs.oracle.com/en/database/goldengate/core/26/coredoc/ai-usecases.html>

ORACLE CORPORATION

Worldwide Headquarters

Nashville, TN 37203 USA

Worldwide Inquiries

TELE + 1.650.506.7000 + 1.800.ORACLE1

FAX + 1.650.506.7200

oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit [oracle.com](https://www.oracle.com). Outside North America, find your local office at [oracle.com/contact](https://www.oracle.com/contact).



blogs.oracle.com/oracle



facebook.com/oracle



x.com/oracle

Copyright © 2026, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

October 2026

Author: Oracle Product Development



Oracle is committed to developing practices and products that help protect the environment

