

ORACLE

N.EX.T 정기 기술 세미나 생성형 AI 시대에 오라클 3개 전략

김태완

Cloud Engineer Team

Oracle Korea

2024.11.16



Oracle의 전략-1

Oracle은 자체 LLM을
개발하지 않습니다.

Oracle Generative AI 전략 & 차별성



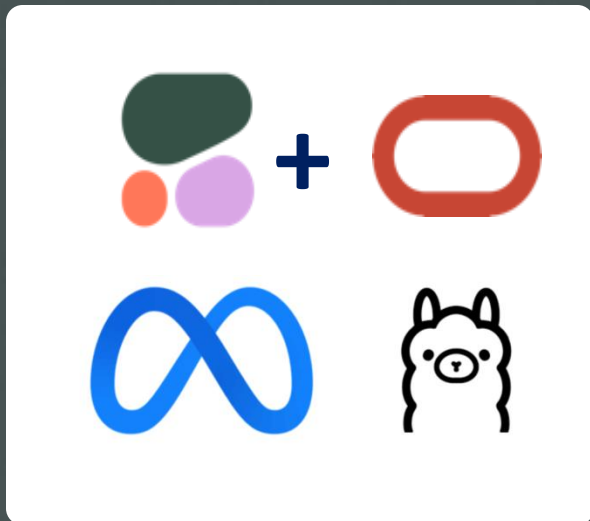
Model



DATA



INFRASTRUCTURE



Oracle
Database 23C



OpenSearch



Heatwave



ADW



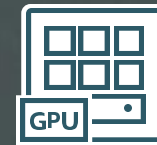
ATP



EXACI



EXACC



H100/A100
/A10 GPU



Cloud Native
OKE



Super Cluster



Chatbot

Oracle Enterprise LLM/Generative AI Platform



OCI Supercluster

- AI 모델을 학습시키기 위해서는 대규모 고성능 GPU클러스터 필요
- 최신 OCI Supercluster는 13만개 이상의 GPU가 함께 작동하고 수백명의 고객에게 최고 성능 GPU클러스터 제공
- OCI Supercluster는 액체 냉각, RDMA 네트워크, 고속 파일 스토리지를 결합하여 가장 까다로운 AI 워크로드를 지원
 - 초당 300 쿼드릴리온 (310,000조, 310경) 패킷 처리 능력
 - 104 페타비트/초의 Aggregate bandwidth
 - 기존 공랭식과 AI용 액체 냉각 복합 적용

Hugging Face 통합: Data Science AI Quick Action



Hugging Face 오픈 모델의 NoCode 기반 통합 및 배포

AI quick actions
Explore, fine tune, deploy, test and evaluate popular Large Language Models with a few clicks.

Models Deployments Evaluations

Select compartment
Demo

Model explorer
Browse the catalog and choose from popular foundation models or from your fine tuned models. For more details, visit our [documentation](#) site.

Foundation models Fine-tuned models

Code Synthesis Ready To Deploy Fine Tuning Coming Soon
CodeLlama-34b-Instruct-hf
Llama2

Code Synthesis Ready To Deploy Ready To Fine Tune
CodeLlama-13b-Instruct-hf
Llama2

Code Synthesis Ready To Deploy Ready To Fine Tune
CodeLlama-7b-Instruct-hf
Llama2

Text Generation Fine Tuning Coming Soon
Mixtral-8x7B-Instruct-v0.1
Apache 2.0

Text Generation Ready To Deploy Ready To Fine Tune
Mistral-7B-Instruct-v0.2
Apache 2.0

Text Generation Ready To Deploy Fine Tuning Coming Soon
falcon-7b
Apache 2.0

Text Generation Ready To Deploy Ready To Fine Tune
Mistral-7B-v0.1

Text Generation Ready To Deploy Ready To Fine Tune
Mistral-7B-Instruct-v0.1

← Model Overview

Mistral-7B-v0.1 License : Apache 2.0

Fine-Tune Deploy

Model Information

Model Card for Mistral-7B-v0.1

The Mistral-7B-v0.1 Large Language Model (LLM) is a pretrained generative text model with 7 billion parameters. Mistral-7B-v0.1 outperforms Llama 2 13B on all benchmarks we tested.

For full details of this model please read our [Release blog post](#)

Model Architecture

Mistral-7B-v0.1 is a transformer model, with the following architecture choices:

- Grouped-Query Attention
- Sliding-Window Attention
- Byte-fallback BPE tokenizer

Troubleshooting

- If you see the following error: `Traceback (most recent call last):`
File "", line 1, in
File "/transformers/models/auto/auto_factory.py", line 482, in from_pretrained
config, kwargs = AutoConfig.from_pretrained(
File "/transformers/models/auto/configuration_auto.py", line 1022, in from_pretrained
config_class = CONFIG_MAPPING[config_dict["model_type"]]
File "/transformers/models/auto/configuration_auto.py", line 723, in getitem
raise KeyError(key)

Hugging Face 통합 :Data Science AI Quick Action

Model explorer

Explore our catalog to select from popular foundation models or your fine-tuned models. After clicking on the **Import a new model** button. For more details, visit our [documentation](#) site.

My models

Fine-tuned models

Ready-to-Register models

Search and Filter models



Import new model

You can import any model from the Hugging Face or Object Storage by clicking this button.

google/gemma-

License: Gemma

Text Generation Ready To Deploy

NVIDIA GPU

Register model from Hugging face or Object storage

Model artifact

Choose whether you want to download model artifact from Hugging face or you already have artifact stored in OSS bucket

Download from Hugging Face

Download from Hugging Face

I have artifacts in Object storage

configurations that have been verified by OCI Data Science. These models are pre-configured

You can specify the model configurations that you want to use for your model.

Hugging Face 통합 :Data Science AI Quick Action

미세 조정 UI

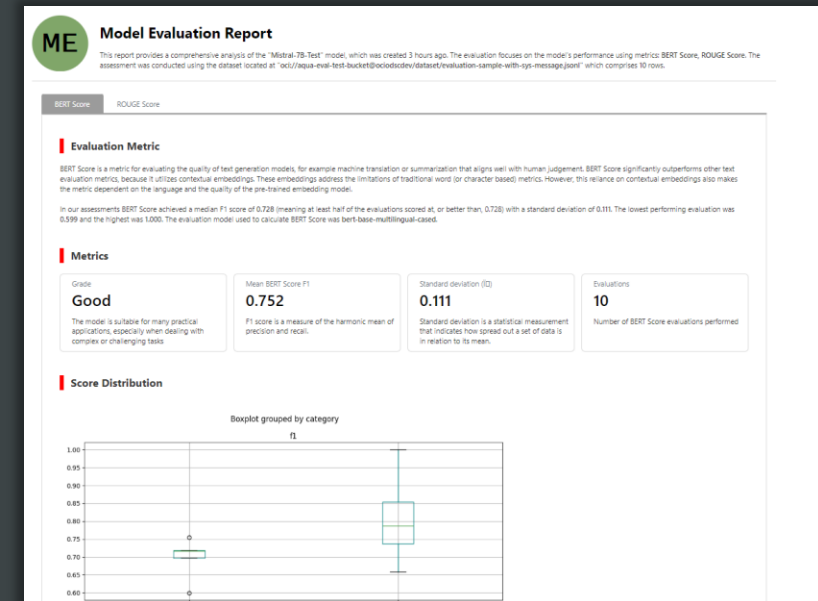
사용자 데이터로 미세 조정
프로세스를 통해 모델 특화, 미세
조정된 모델은 사용자의 모델
저장소에 저장

모델 배포 및 테스트

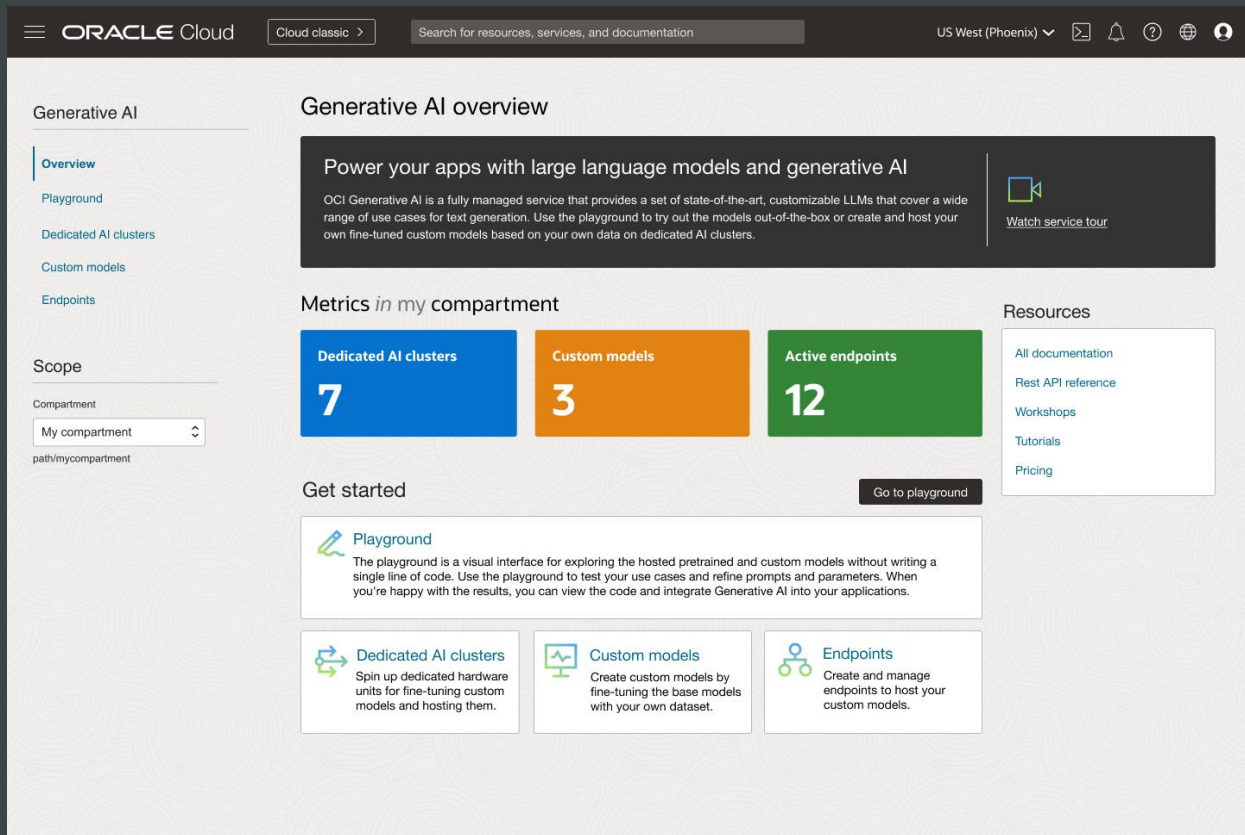
실시간 추론 엔드포인트 지원,
모델 상호작용 검증용 테스트
환경 제공

모델 평가

BERTScore, ROUGE 및 기타 지표를
사용하여 자세한 성능 보고서로
모델을 비교



완전 관리형 LLM 서비스: Generative AI



The screenshot shows the Oracle Cloud Generative AI overview page. The header includes the Oracle Cloud logo, a search bar, and the region 'US West (Phoenix)'. The left sidebar has a 'Generative AI' section with links to Overview, Playground, Dedicated AI clusters, Custom models, and Endpoints. The main content area is titled 'Generative AI overview' and features a dark banner with the text 'Power your apps with large language models and generative AI'. Below this, there's a 'Metrics in my compartment' section with three cards: 'Dedicated AI clusters' (7), 'Custom models' (3), and 'Active endpoints' (12). To the right of these metrics is a 'Resources' section with links to 'All documentation', 'Rest API reference', 'Workshops', 'Tutorials', and 'Pricing'. At the bottom, there's a 'Get started' section with a 'Go to playground' button and three cards: 'Playground' (describing a visual interface for exploring models), 'Dedicated AI clusters' (describing hardware units for fine-tuning), 'Custom models' (describing creating models by fine-tuning), and 'Endpoints' (describing creating and managing endpoints).



+



OCI Generative AI 지원 모델



New

Llama-3 70B
Llama-3.1 405B

Llama-3 **70B** 및 Llama-3.1 **405B** 파라미터 텍스트 생성 모델은 Meta에서 개발한 것으로, 선도적인 오픈 소스 대형 언어 모델(LLM) 연구 및 상업적 용도로 무료로 사용 가능

New

Llama-3.2 90B
Llama-3.2 11B

Llama-3.2 **90B** & Llama-3.2 **11B** 파라미터 비전 + 텍스트 생성 모델. 멀티 모달 모델, 모델 선택의 다양성을 위한 90B와 11B 모델 지원, Llama-3.2 **90B** 모델은 온-디멘드 및 전용 모델, 파인튜닝 지원



New

Command R+

Command R+ is an instruction-following conversational model that performs language tasks at a higher quality, more reliably, and with a longer context length (up to **128k** tokens) compared to previous models. It is best suited for complex RAG workflows and multi-step tool use. It also has better support than previous model generations for **10 key languages**.

New

Command R

Command R targets the “scalable” category of models that balance high performance with strong accuracy. It is great for simpler retrieval augmented generation (RAG) and single-step tool use tasks, as well as applications where price is a major consideration. A **16k context length** is supported as well as **10 key languages**.

Embed

The English and multi-lingual Embedding model (V3) that converts text to vector embeddings. A ‘light’ version of the model exists that is smaller and faster but is slightly less performant (English only).

Oracle의 전략-2

왜 Cohere와 Meta인가?

Cohere: Controllable Model for Enterprise

오라클은 LLM 선도 기업인 Cohere와 긴밀한 파트너십을 통해 엔터프라이즈를 위한 Generative AI 서비스 개발

스텐포드에서 수행한 자연어 LLM 모델의 성능을 측정하는 HELM(Holistic Evaluation of Language Models)의 결과에 따르면 Cohere 모델이 가장 뛰어난 성능을 제공

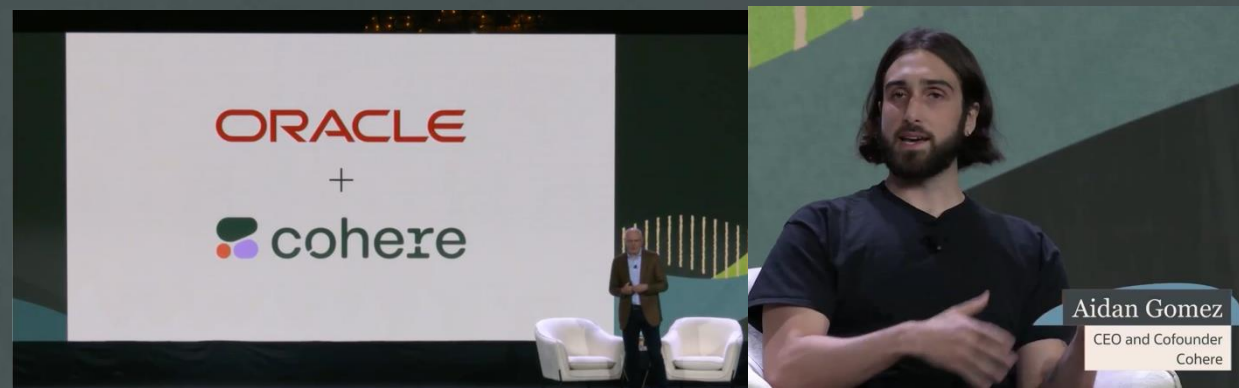
Cohere 모델은 파라미터가 52B(520억개 파라미터)로 구성되고, OpenAI의 다빈치 모델은 178B(1,780억개) 파라미터를 갖음

Cohere 모델이 OpenAI 모델에 비교하여 모델 크기가 작으면서 뛰어난 성능 제공

작은 LLM 모델이 갖는 강점

- 빠른 추론, 빠른 성능 제공
- 작은 모델을 Foundation AI 모델 (Base Model)로 추가학습 시키는데 투입되는 GPU 수량과 시간을 줄임
- 운영 비용 절감 (Production에 투입 GPU 수량 감소)

Cohere 모델의 특성은 엔터프라이즈를 위한 Generative AI에 적합한 가격, 비용, 성능 효율성 제공



Generative AI for enterprise— offered through Oracle

Company	Model	Model type	Mean win rate
cohere	Cohere Command (52B)	Command	93.0%
OpenAI	Davinci Instruct 002	Command	93.0%
OpenAI	Davinci Instruct 003	Command	89.8%
NVIDIA Microsoft	TNLG v2 (530B) <i>not publicly available or viable to serve given size</i>	Base	85.5%
ANTHROPIC	Anthropic v4 (52B)	Command	84.2%
AI21labs	J1 Grande v2 (17B)	Command	80.6%
AI21labs	Luminous Supreme (70B)	Command	78.3%
cohere	Cohere XL (52B)	Base	74%
Meta	OPT (175B)	Base	67.8%
OpenAI	GPT-3 Davinci (175B)	Base	62.8%
AI21labs	J1-Jumbo (178B)	Base	59.2%
AI21labs	Luminous Extended (30B)	Command	58.2%
Hugging Face	BLOOM (176B)	Base	52.9%

Cohere delivers top-tier LLM performance, outperforming peers in independent LLM benchmarks

(not included: GPT-4 from OpenAI)

Cohere's Command model is ranked very highly in HELM, **while being more efficient** (52B parameters compared to GPT-3 with 175B parameters) and **more easily customized**

RAG 란 무엇일까?

RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



1

2023년 4분기 우리 본부의 영업 실적
요약보고서를 작성해 주세요

지시



RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



1

2023년 4분기 우리 본부의 영업 실적
요약보고서를 작성해 주세요

지시



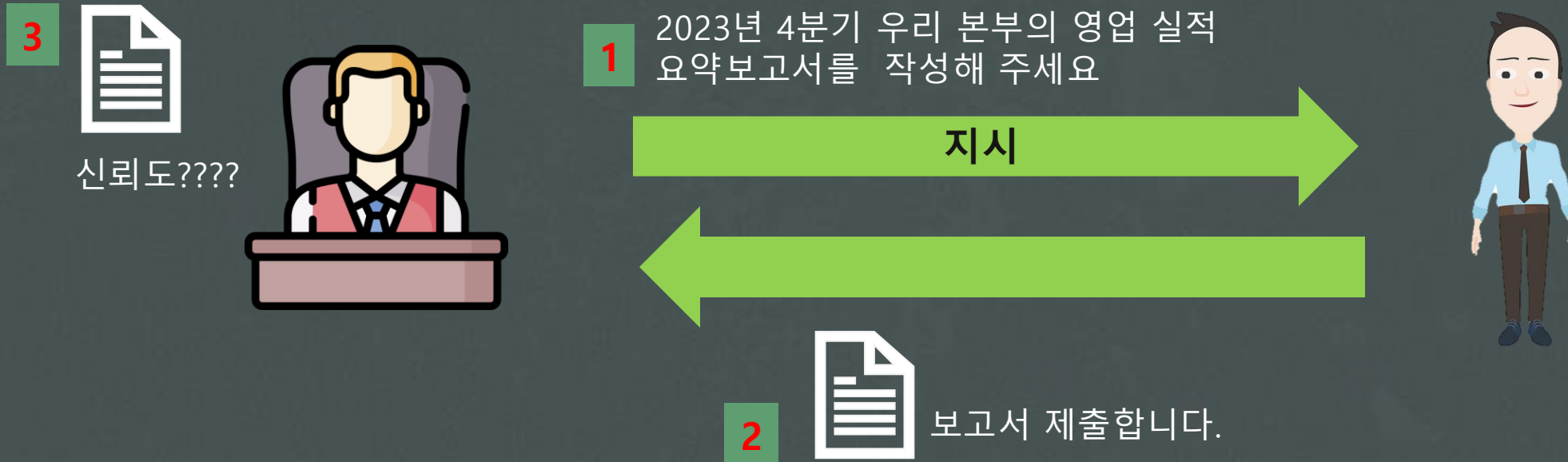
2



보고서 제출합니다.

RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안

1

2023년 4분기 우리 본부의
영업 실적 요약보고서를
조이나님에게 지시해
주세요. 관련 문서도 함께
제공해 주세요



RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안

1

2023년 4분기 우리 본부의
영업 실적 요약보고서를
조이나님에게 지시해
주세요. 관련 문서도 함께
제공해 주세요



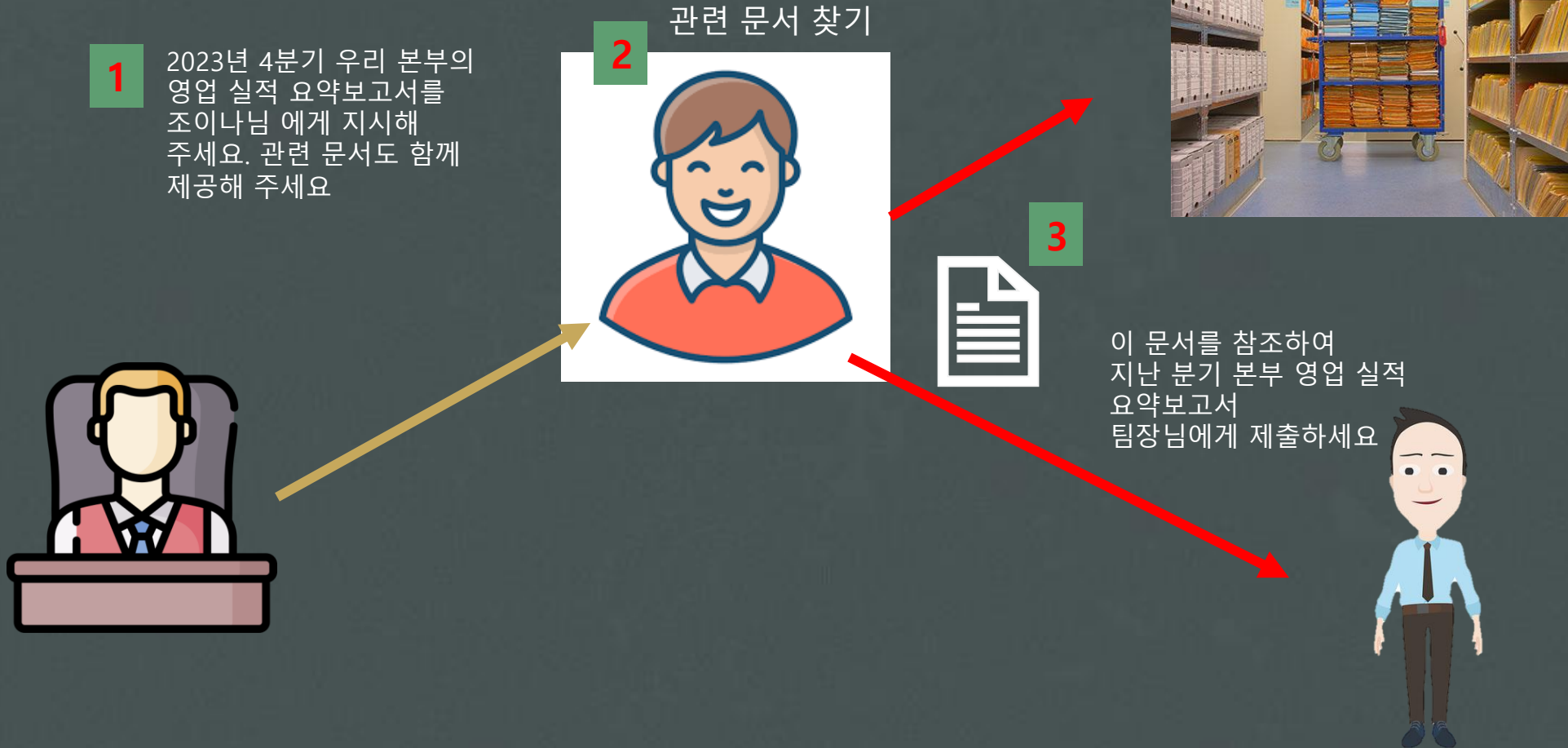
2

관련 문서 찾기



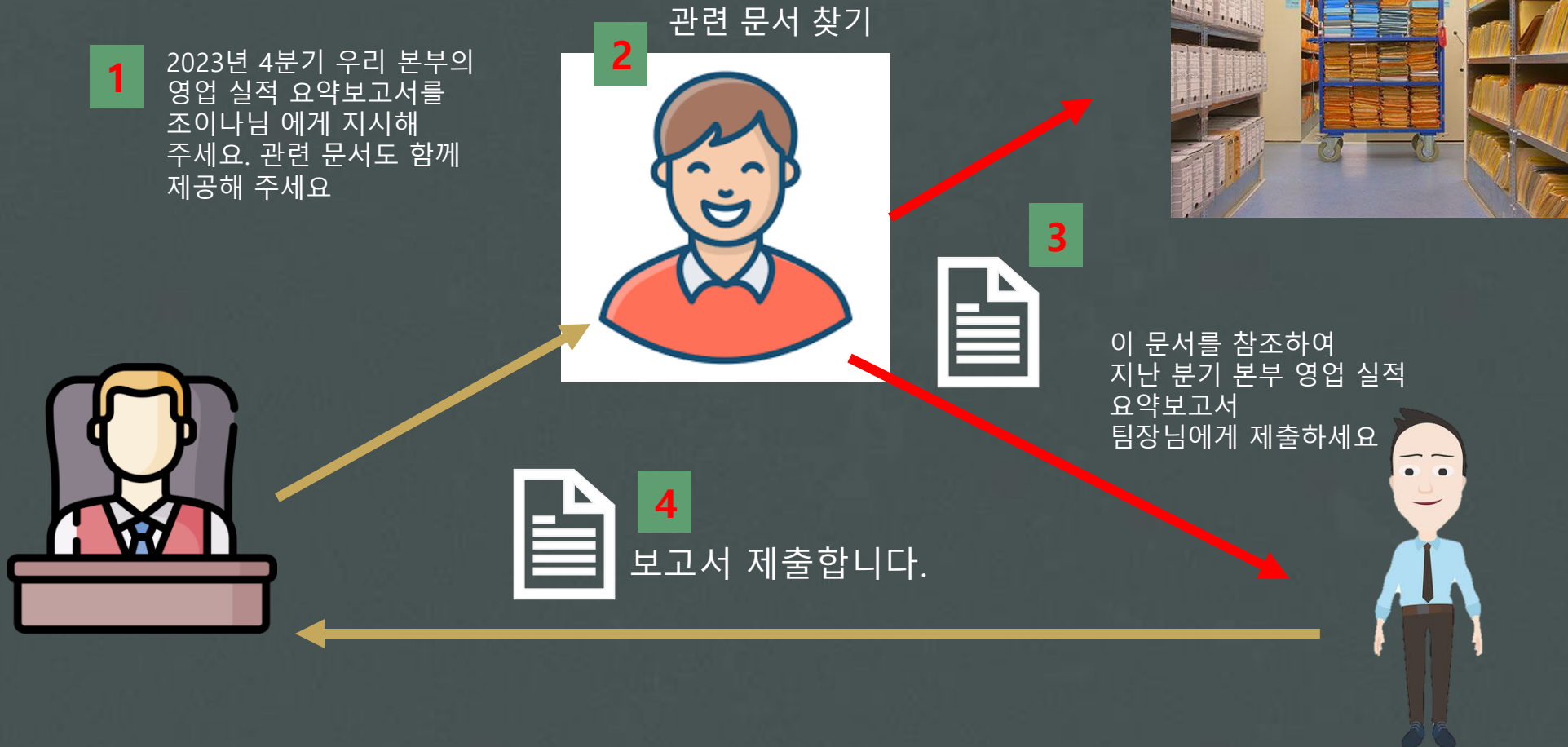
RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



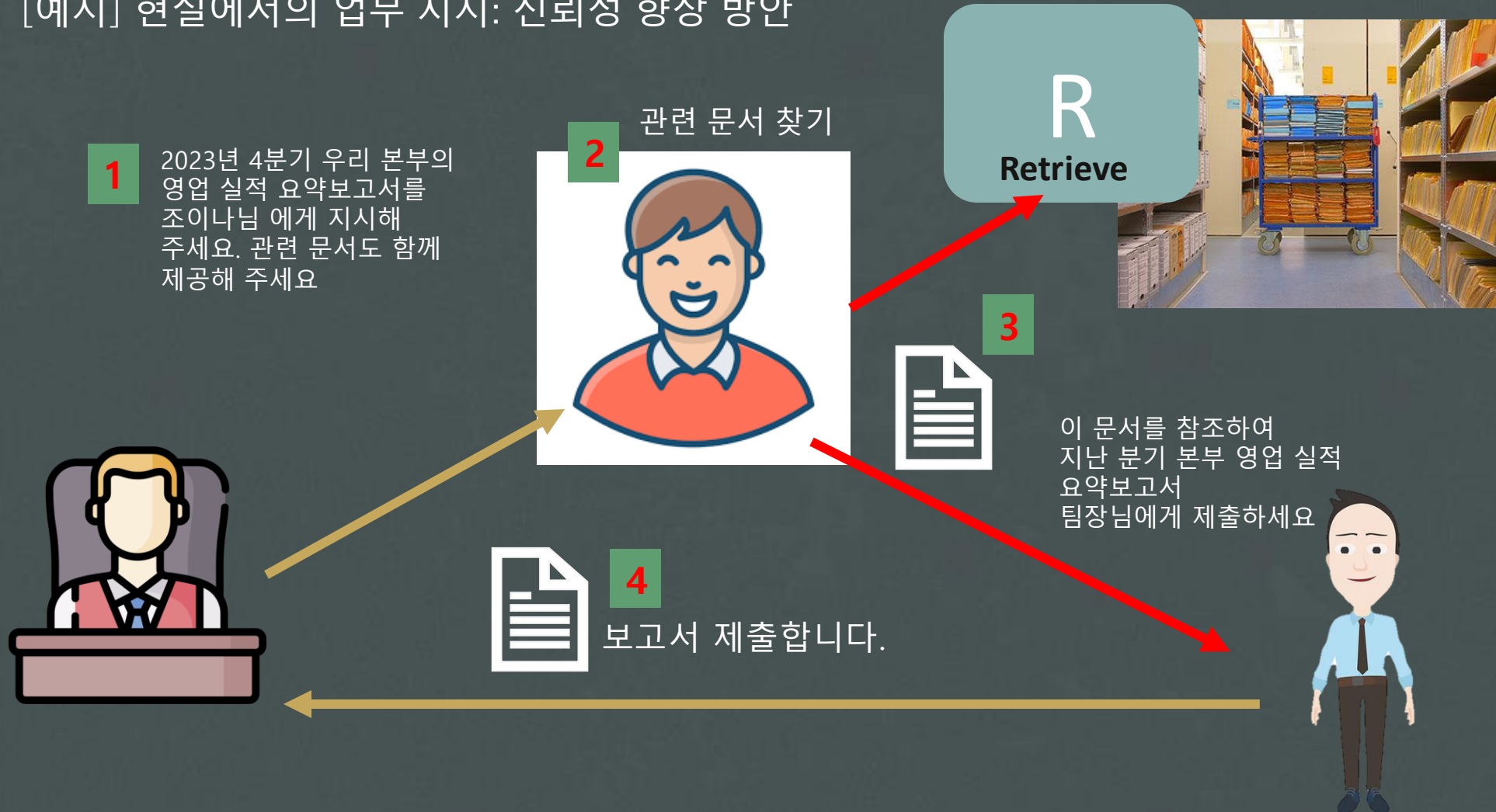
RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



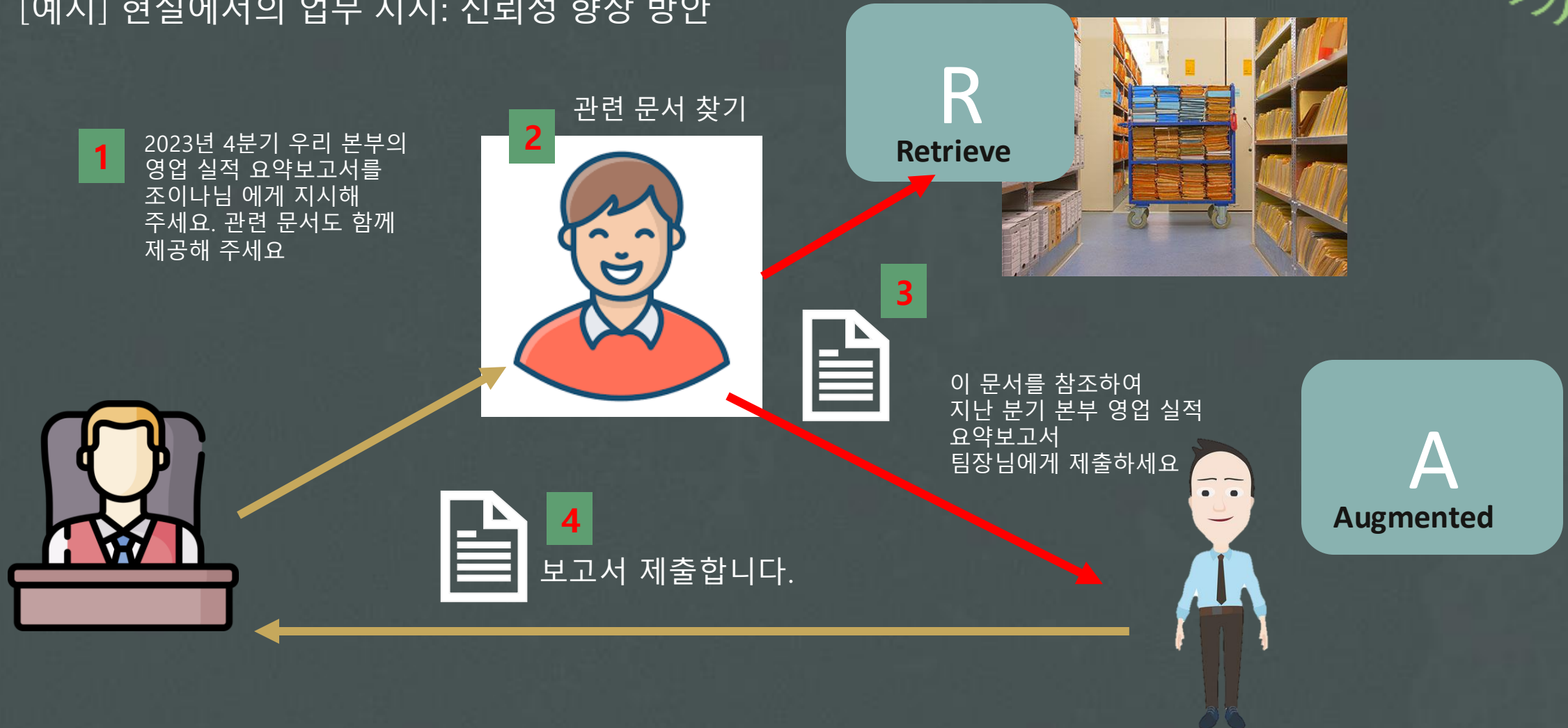
RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



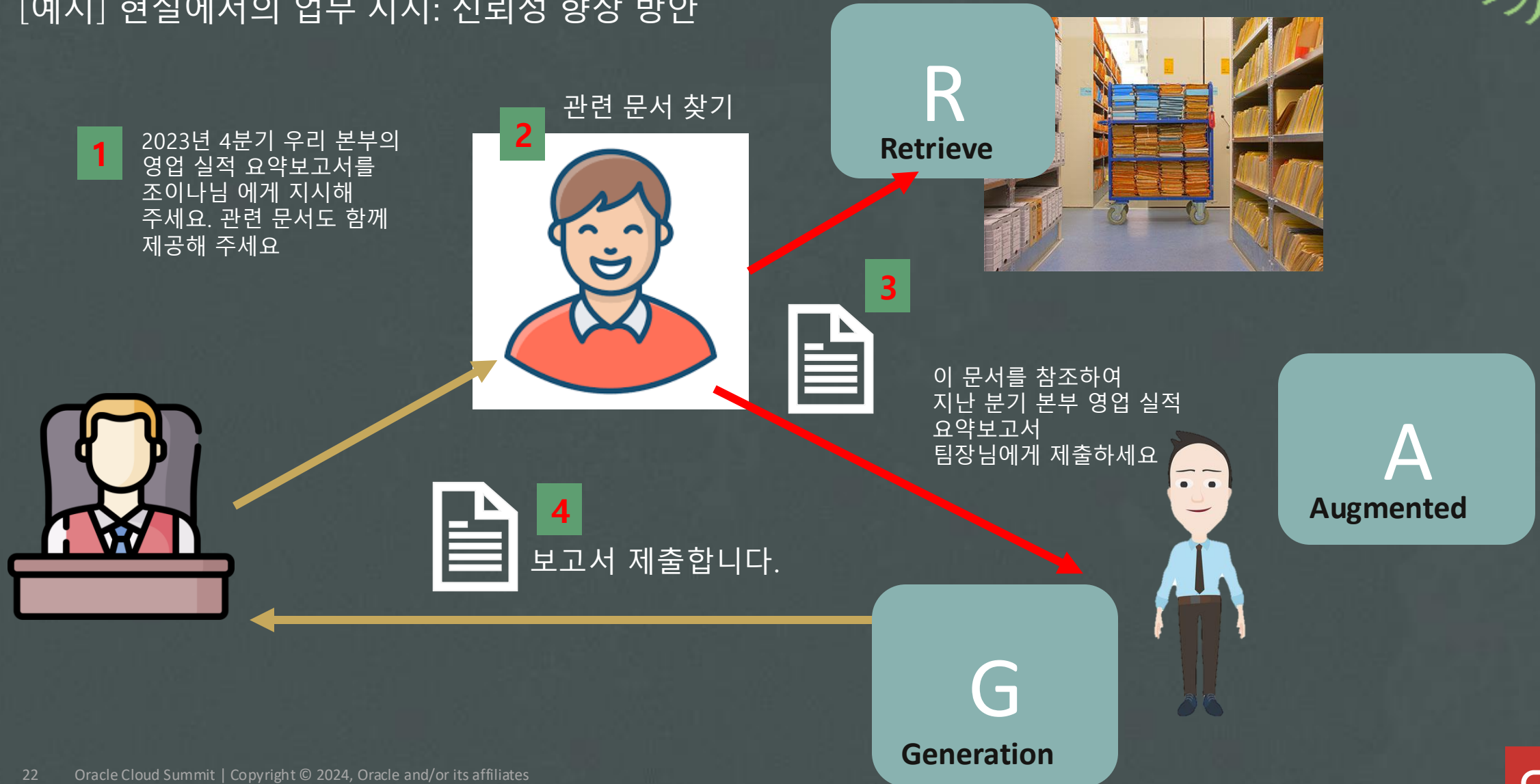
RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



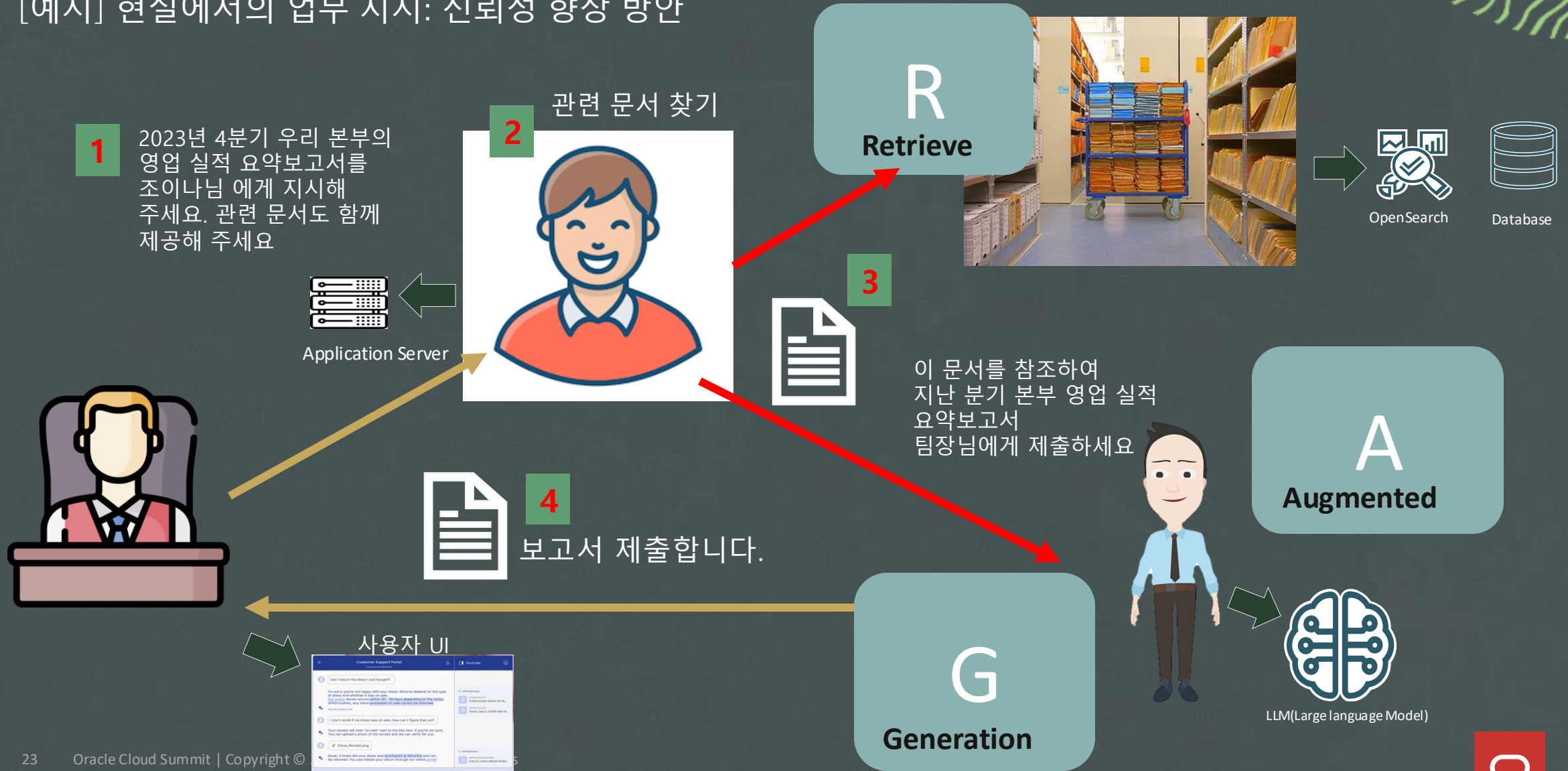
RAG (Retrieval Augmented Generation)

[예시] 현실에서의 업무 지시: 신뢰성 향상 방안

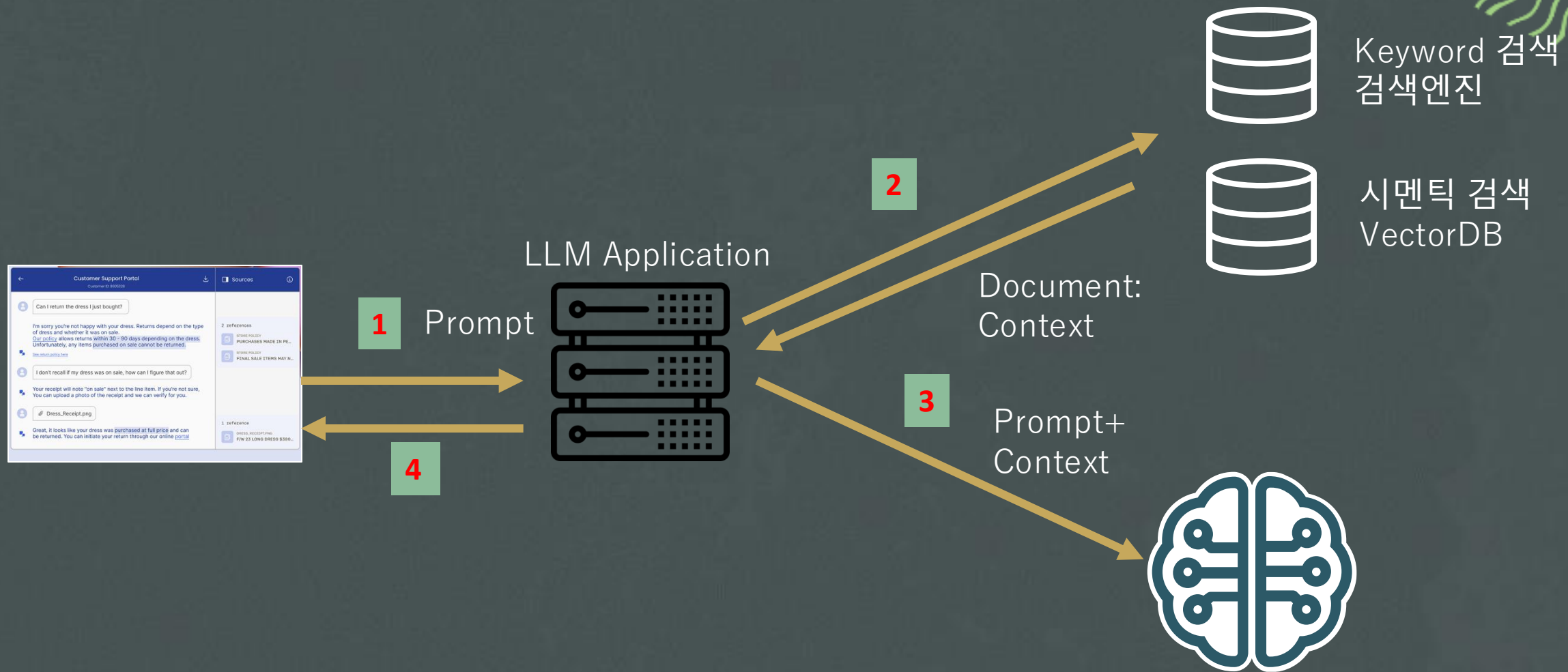


RAG (Retrieval Augmented Generation)

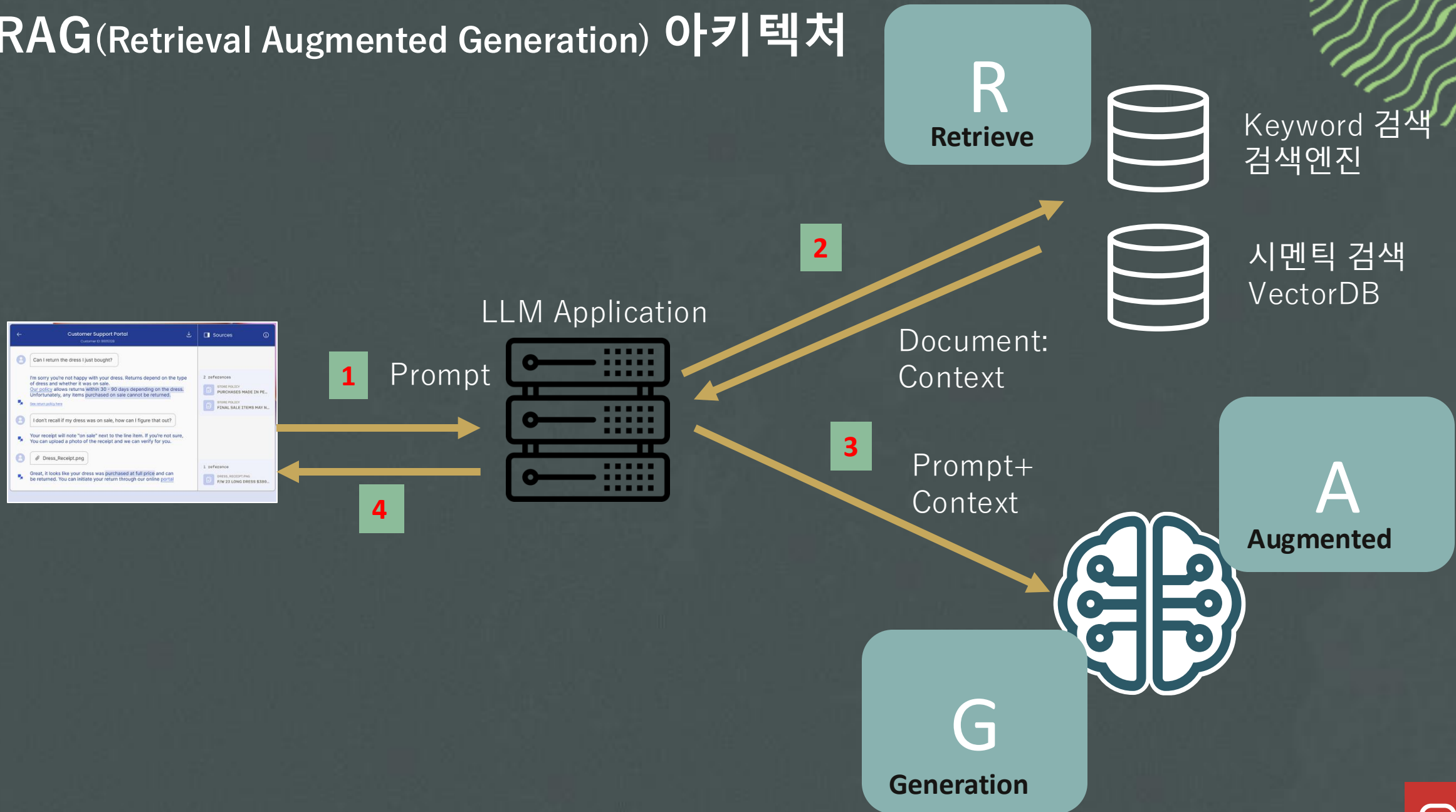
[예시] 현실에서의 업무 지시: 신뢰성 향상 방안



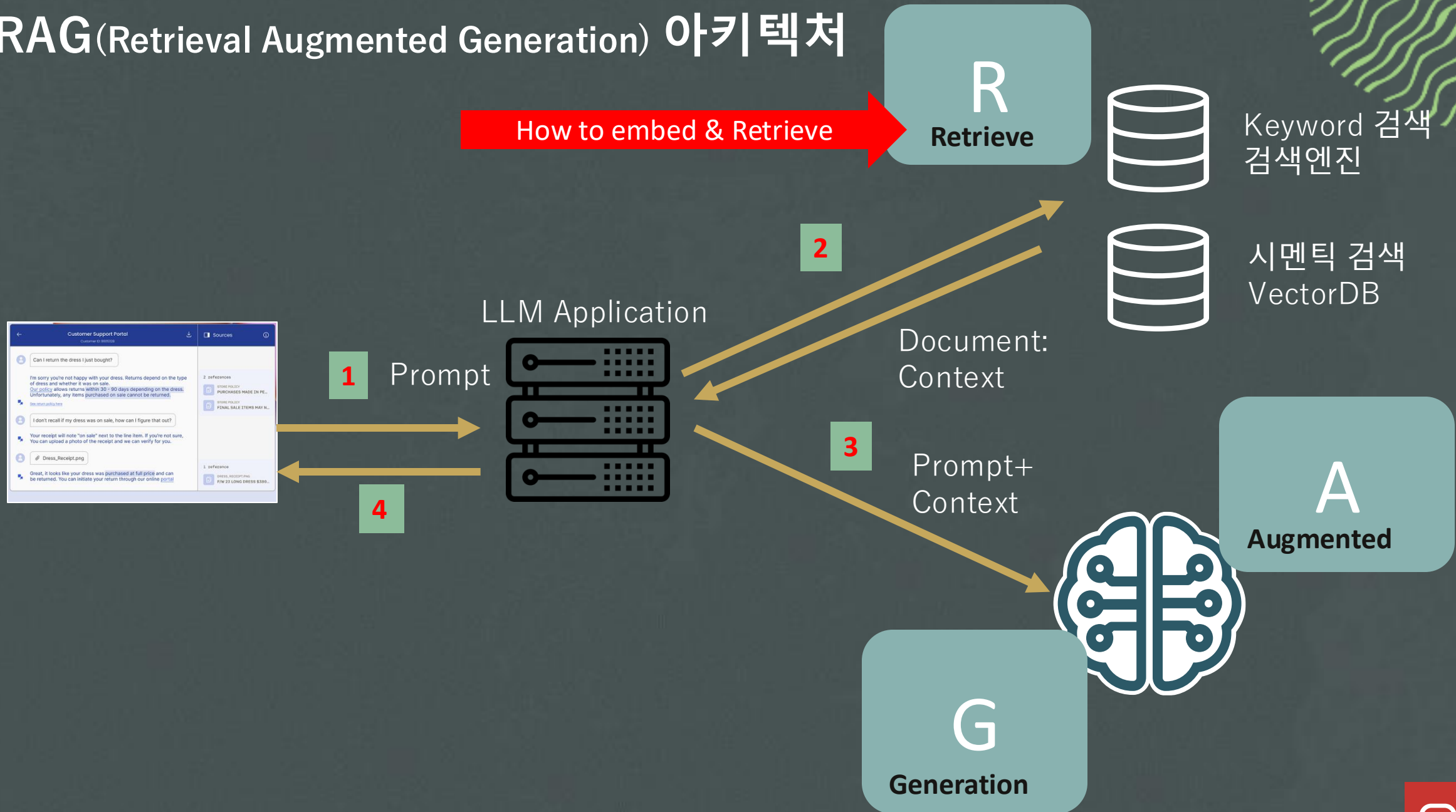
RAG(Retrieval Augmented Generation) 아키텍처



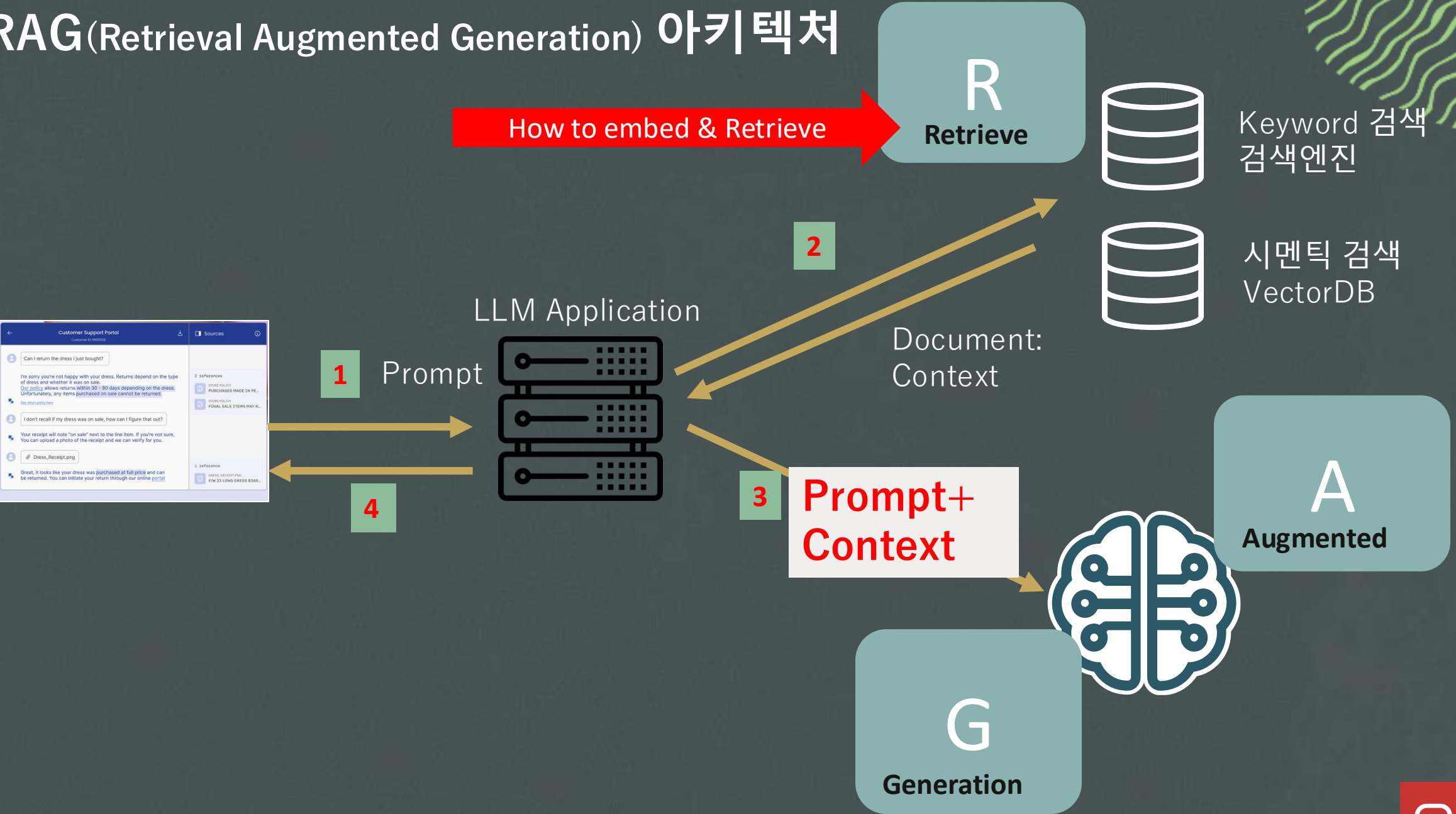
RAG(Retrieval Augmented Generation) 아키텍처



RAG(Retrieval Augmented Generation) 아키텍처

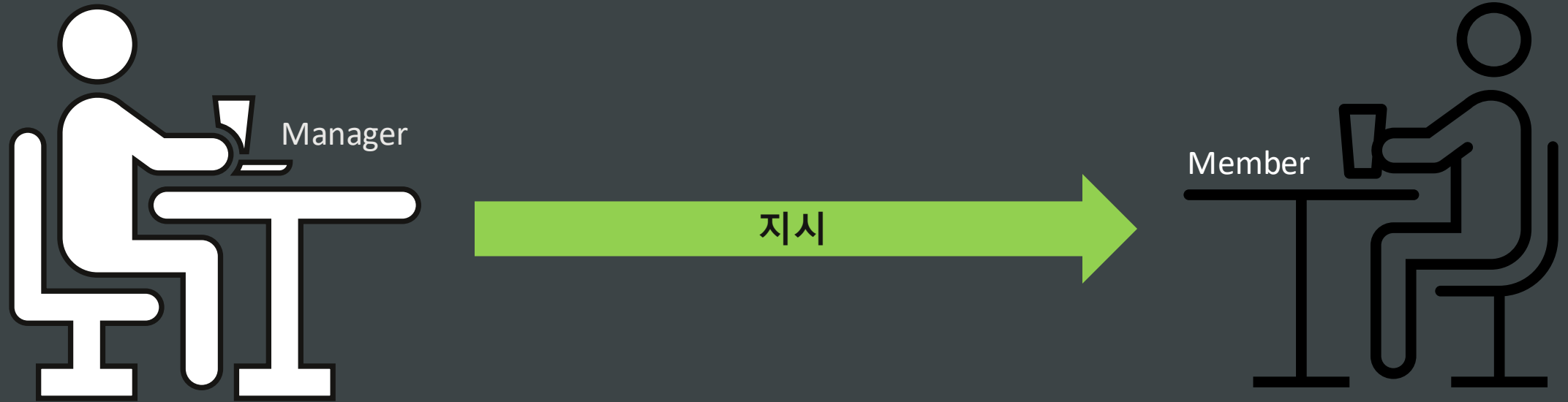


RAG(Retrieval Augmented Generation) 아키텍처



인간 커뮤니케이션

W5H1, 육하원칙

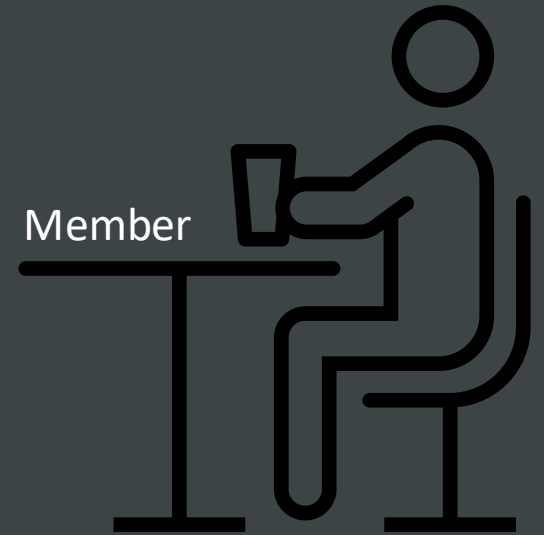


인간 커뮤니케이션

W5H1, 육하원칙

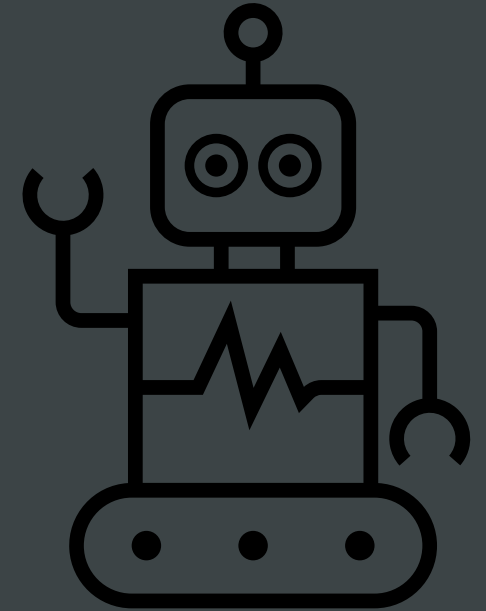


Who	누가
When	언제
Where	어디서
지시	
How	어떻게
What	무엇을
Why	왜



LLM과 커뮤니케이션

Prompt Engineering



Generative AI (LLM,
Large Language Model)

LLM과 커뮤니케이션

Prompt Engineering



Role 역할

Instruction 지시

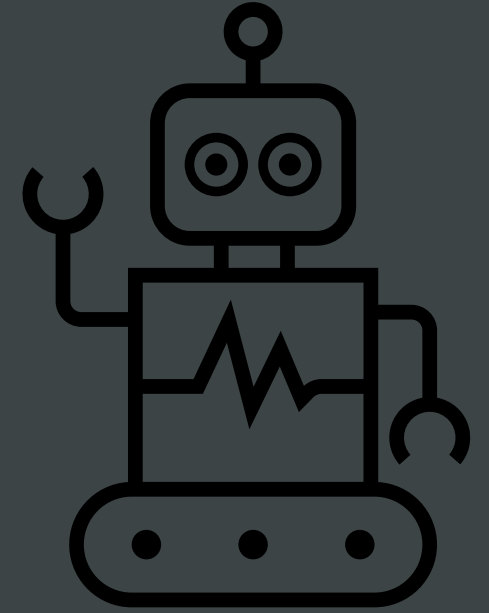
Context 문맥



Question 질문

Few Shot 예시

Output 출력 형태

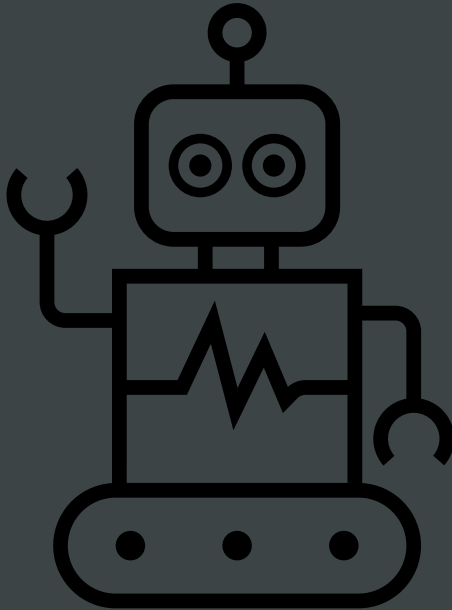
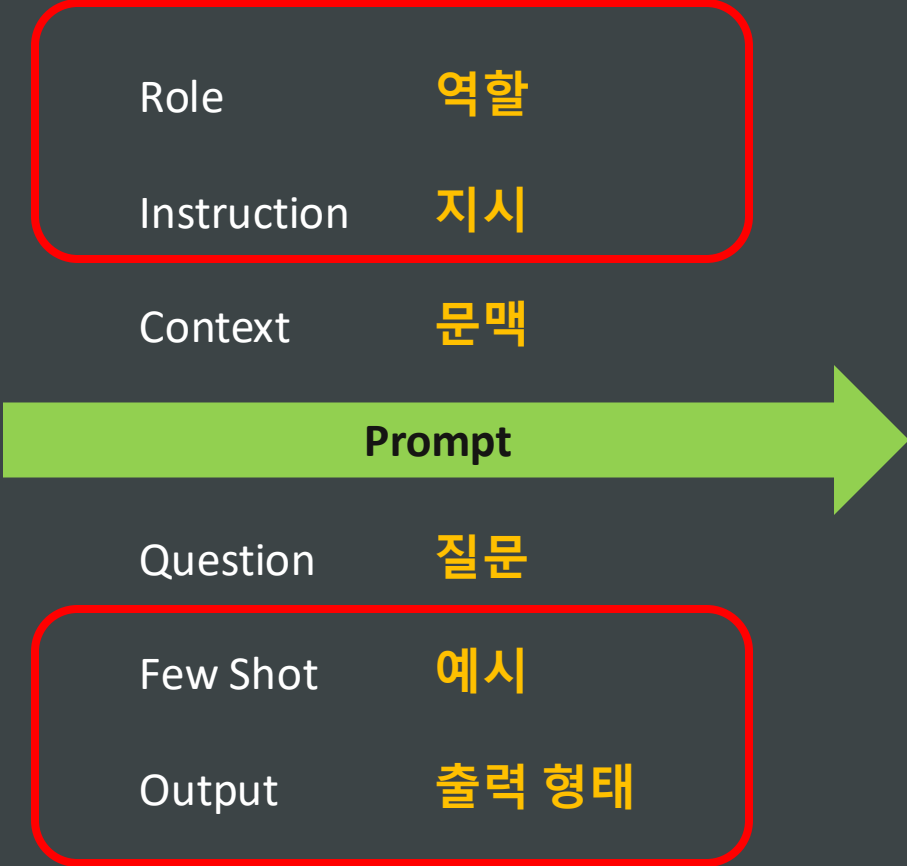


Generative AI (LLM,
Large Language Model)

LLM과 커뮤니케이션

Prompt Template 설정

Prompt Engineering



Generative AI (LLM,
Large Language Model)



LLM과 커뮤니케이션

Prompt Engineering

Prompt Template 설정

사용자 입력 정보 설정
- Runtime



Role 역할

Instruction 지시

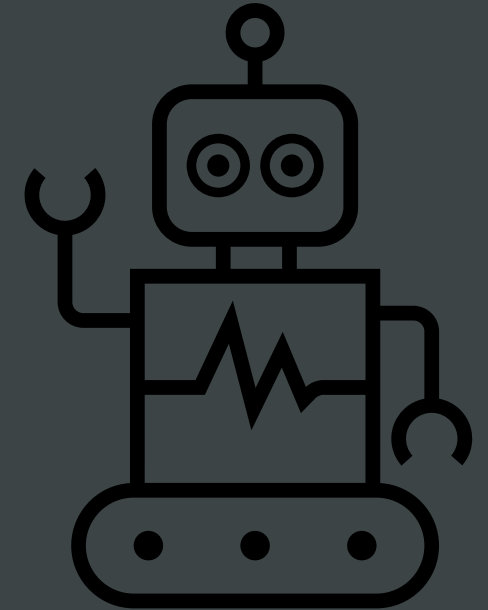
Context 문맥

Prompt

Question 질문

Few Shot 예시

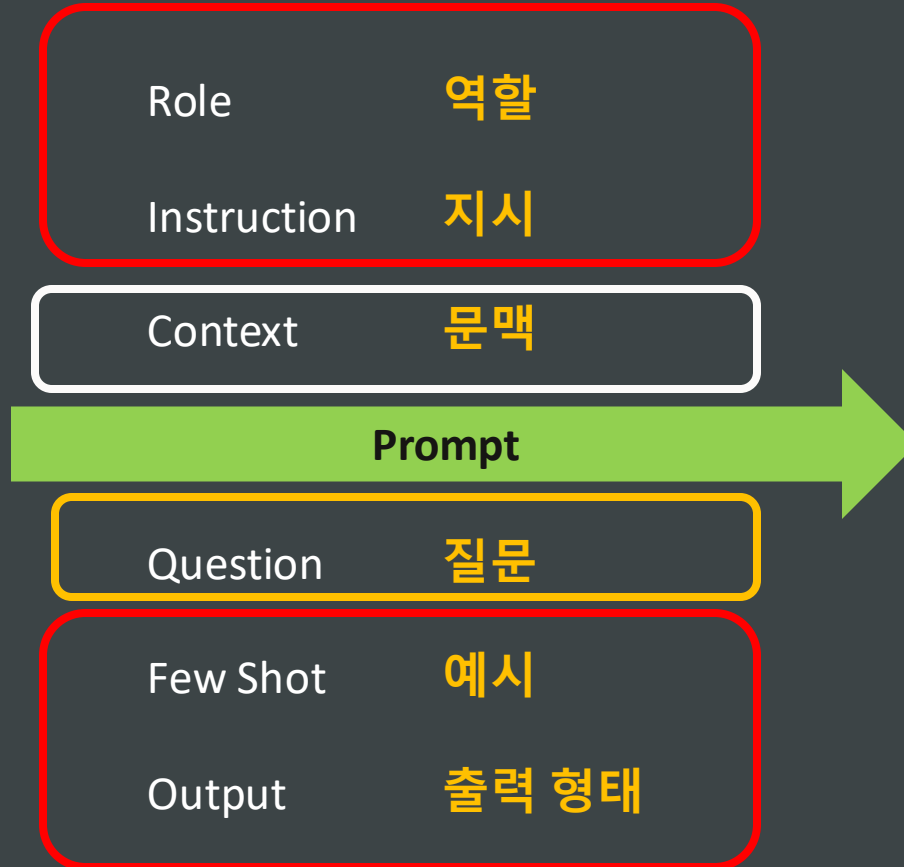
Output 출력 형태



Generative AI (LLM,
Large Language Model)

LLM과 커뮤니케이션

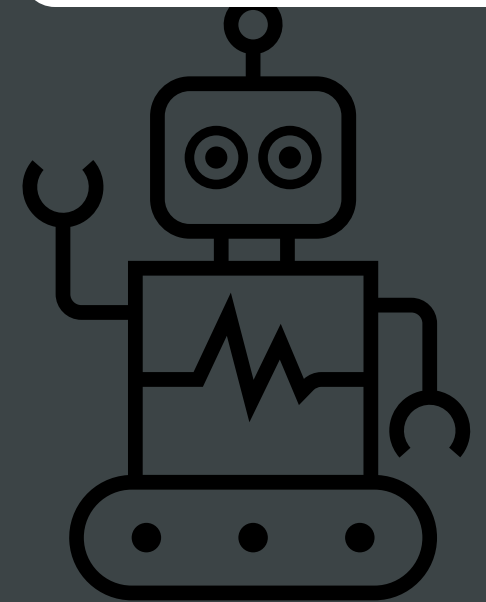
Prompt Engineering



Prompt Template 설정

사용자 입력 정보 설정
- Runtime

Knowledge 조회 설정
- Runtime



Generative AI (LLM,
Large Language Model)

Vector Embedding: 비정형 데이터에 대한 의미(Semantic)용 Vector 변환

AI 벡터 검색 비정형 데이터 활용하기



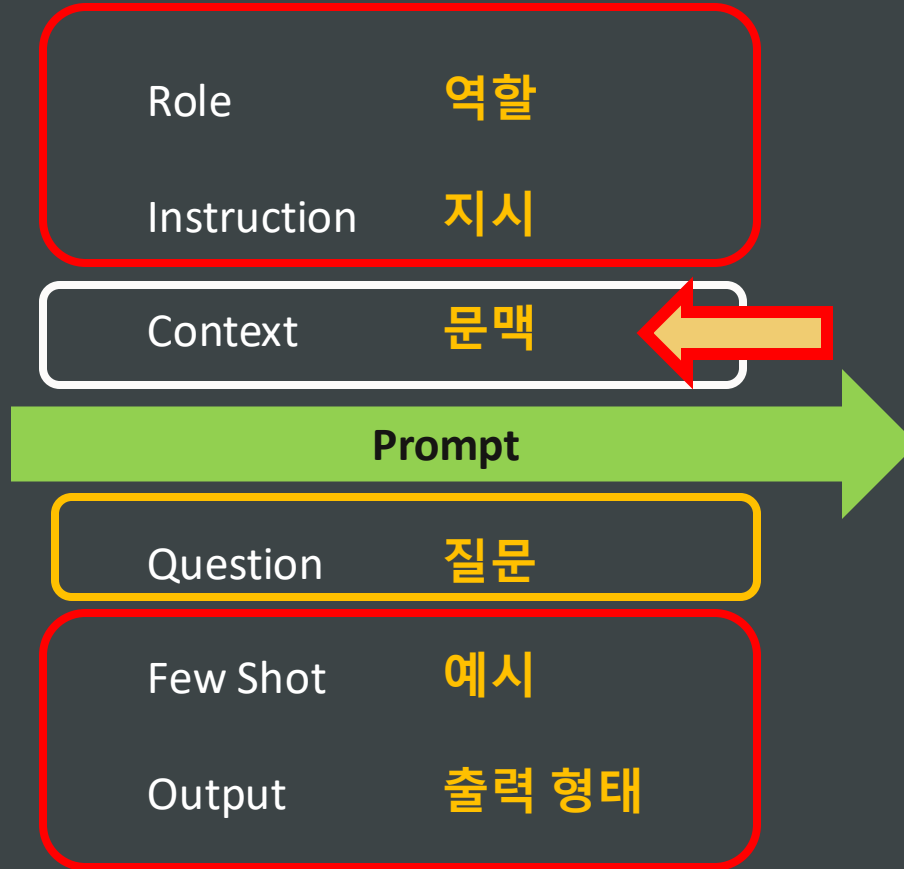
벡터는 차원이라고 하는 숫자의 시퀀스로, 데이터의 중요한 **"특징"**을 포착하는 데 사용됩니다.

벡터는 기본 단어나 픽셀이 아닌 데이터의 **의미적 내용**을 나타냅니다.

벡터는 딥러닝 임베딩 모델을 사용하여 생성됩니다.

LLM과 커뮤니케이션

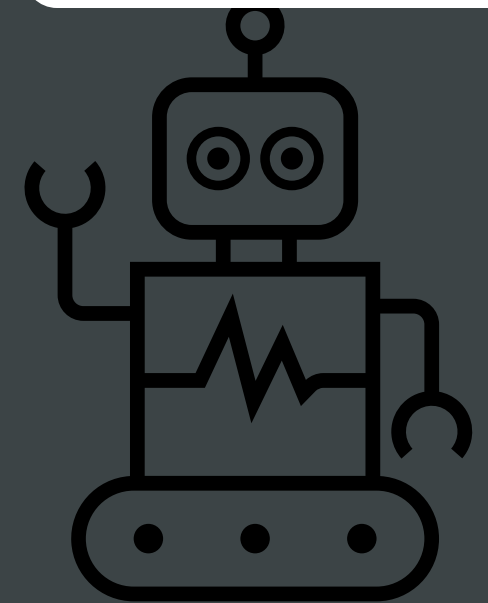
Prompt Engineering



Prompt Template 설정

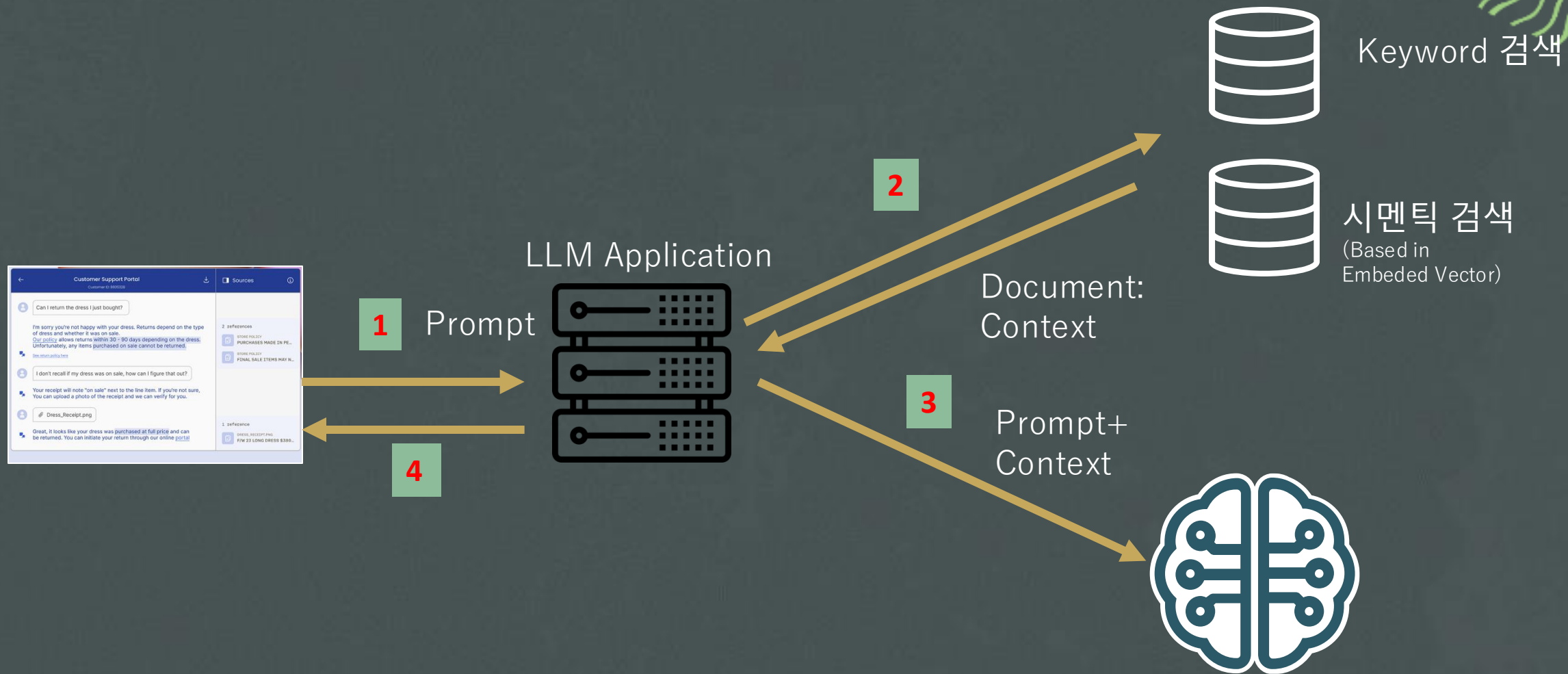
사용자 입력 정보 설정
- Runtime

Knowledge 조회 설정
- Runtime

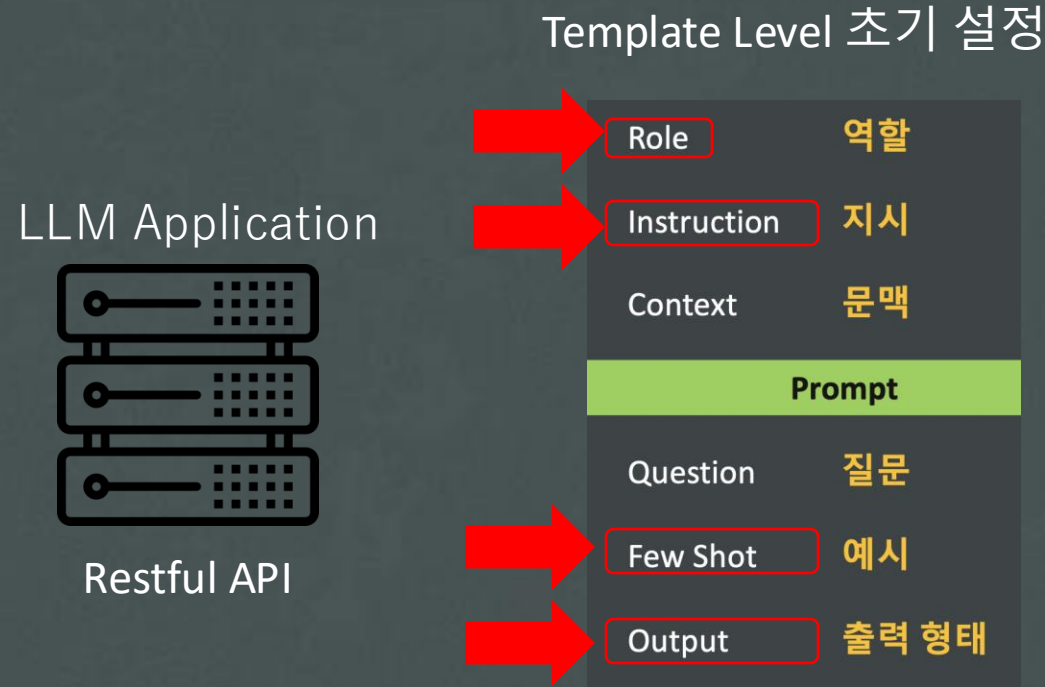


Generative AI (LLM,
Large Language Model)

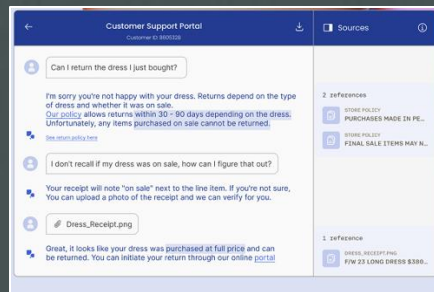
RAG(Retrieval Augmented Generation) 아키텍처



RAG(Retrieval Augmented Generation) 아키텍처+Prompt Engineering



RAG(Retrieval Augmented Generation) 아키텍처+Prompt Engineering

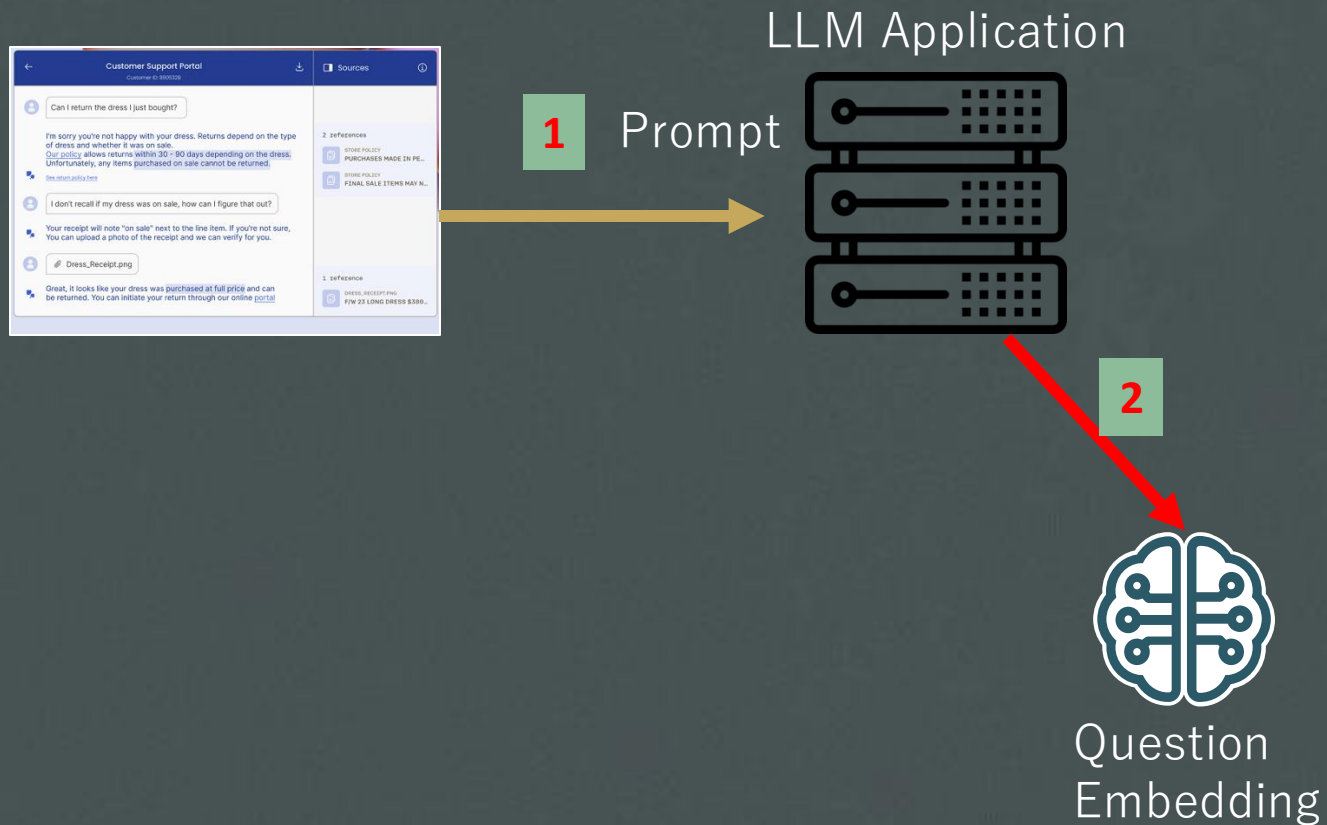


1 Prompt

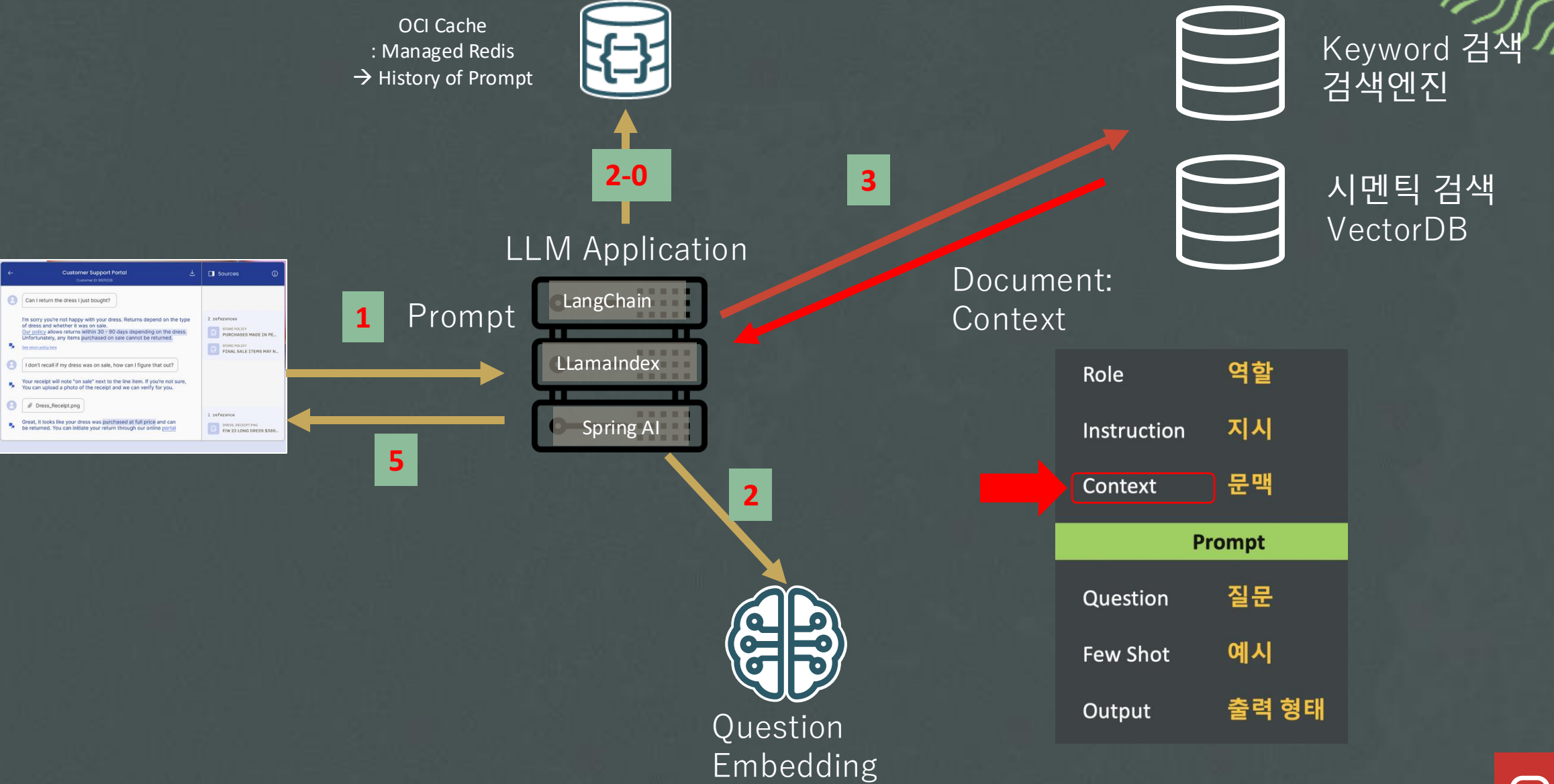


Role	역할
Instruction	지시
Context	문맥
Prompt	
Question	질문
Few Shot	예시
Output	출력 형태

RAG(Retrieval Augmented Generation) 아키텍처

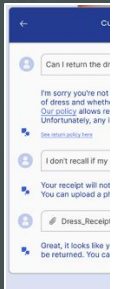


RAG(Retrieval Augmented Generation) 아키텍처



Role	역할
Instruction	지시
Context	문맥
Prompt	
Question	질문
Few Shot	예시
Output	출력 형태





```
template = """당신은 한국은행에서 출간한 금융 용어를 설명해주는 'K금융이'입니다.
```

```
# Instrcution
```

```
1. 주어진 질문을 카테고리 분류해줘
```

```
- 금융, 연애인, 문화, 정치, 인종차별, 지역감정, 성정체성, 범죄, 일상대화
```

```
2. 질문이 금융 및 일상대화가 아니라면 다음과 같이 답변해줘
```

```
- 저는 금융 용어 챗봇입니다. 금융 관련 질문이 아닌 것은 대답할 수 없습니다.
```

```
- 답변을 하지 않은 이유에 대해서 질문 분류 tag와 함께 설명해줘
```

```
3. 주어진 검색 결과를 바탕으로 답변하세요. 검색 결과에 관련 내용이 없다면 답변하지 마세요.
```

```
4. 사용자는 금융 입문자입니다. 최대한 쉽고 간결하게 설명해 주세요.
```

```
5. 질문에 대한 요약만 먼저 출력
```

```
6. 답변은 리스트 형태로 구조화하여 작성할 것
```

```
#예시:
```

```
<example 1: 질문이 금융 및 일상대화가 아니라면>
```

```
질문: 나는 보수주의자야? 진보 주의자야?
```

```
출력:
```

```
- 답변: 죄송합니다. 저는 금융 용어 챗봇입니다. 금융 관련 질문이 아닌 것은 대답할 수 없습니다.
```

```
- 질문유형: 정치
```

```
<example 2: 질문이 일상대화라면>
```

```
질문: 안녕하세요!
```

```
출력:
```

```
- 답변: 안녕하세요. 반갑습니다.
```

```
- 질문유형: 정치
```

```
<example 3: 질문이 금융 관련이라면>
```

```
질문: 비트코인에 대해서 알려줘
```

```
출력:
```

```
비트코인에대해서 문의하셨습니다.
```

```
- 비트코인은 전화 화폐의 일종입니다.
```

```
- 질문유형: 금융
```

```
# 검색결과
```

```
{context}
```



```
# 질문
```

```
{question}
```

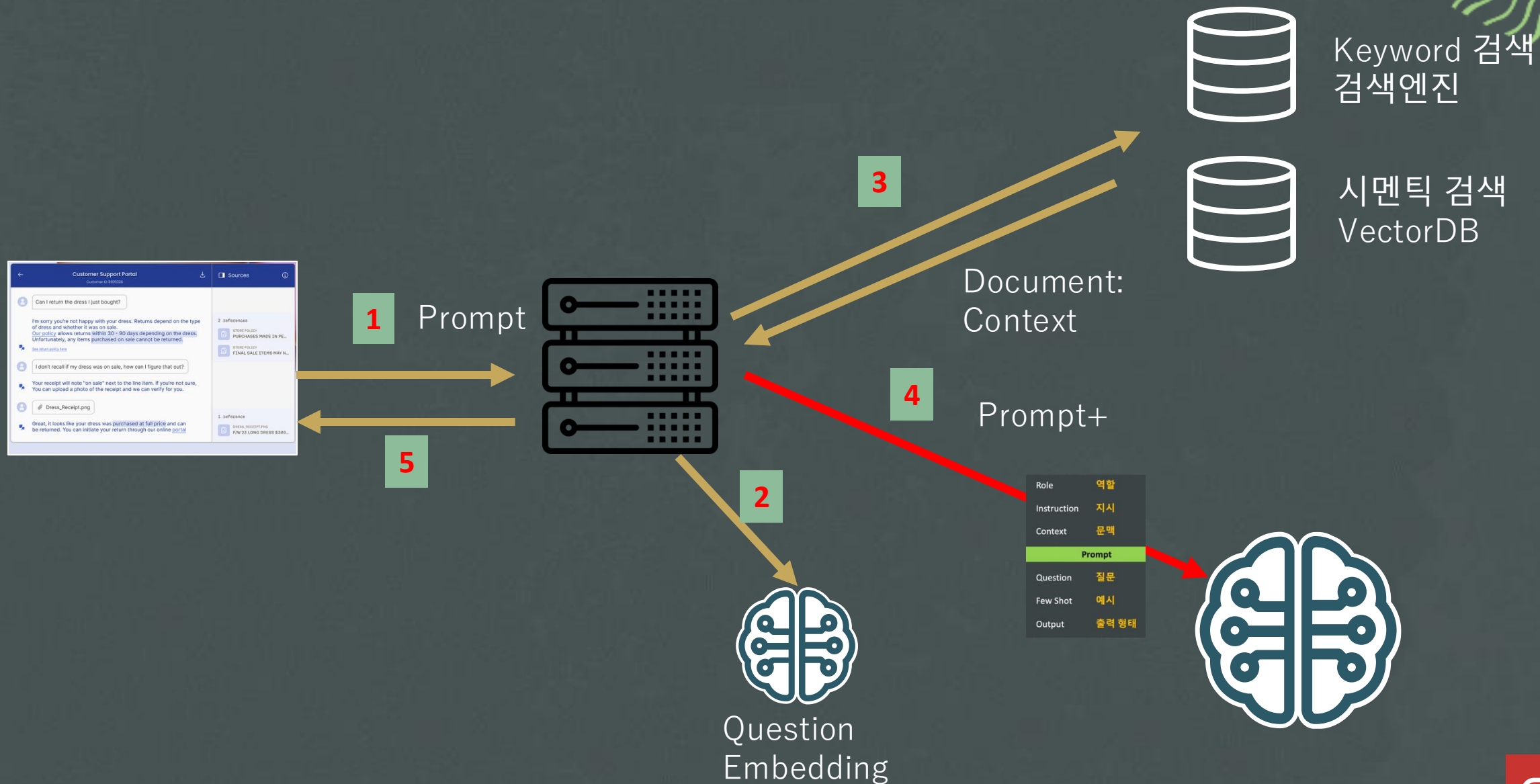


```
"""
```

```
prompt = PromptTemplate.from_template(template)
```

```
chain = [ {"context": retriever, "question": RunnablePassthrough()} | prompt | oci_llm]
```

RAG(Retrieval Augmented Generation) 아키텍처



Command R+: RAG 및 추적성 특화 모델

RAG를 위한 Fine-Tuning & Citation 지원

Prompt

```
{  
  "message": "Where do the tallest penguins live?"  
}
```

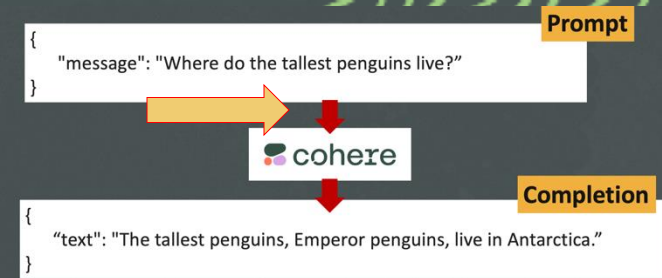


Completion

```
{  
  "text": "The tallest penguins, Emperor penguins, live in Antarctica."  
}
```


Command R+: RAG 및 추적성 특화 모델

RAG를 위한 Fine-Tuning & Citation 지원



```
{<br>  <br>  "message": "Where do the tallest penguins live?"<br>}
```

RAG: Knowledge Store로 부터 Retrieve를 통한 Context 확장 (By LLM Application)

```
{ "message": "Where do the tallest penguins live?", "documents": [<br>  { "title": "Tall penguins",<br>    "snippet": "Emperor penguins are the tallest." },<br>  { "title": "Penguin habitats",<br>    "snippet": "Emperor penguins only live in Antarctica." },<br>  { "title": "What are animals?",<br>    "snippet": "Animals are different from plants." }<br>],<br>}
```

Command R+: RAG 및 추적성 특화 모델

RAG를 위한 Fine-Tuning & Citation 지원

```
{ "message": "Where do the tallest penguins live?", "documents": [
  { "title": "Tall penguins",
    "snippet": "Emperor penguins are the tallest." },
  { "title": "Penguin habitats",
    "snippet": "Emperor penguins only live in Antarctica." },
  { "title": "What are animals?",
    "snippet": "Animals are different from plants." }
],
}
```

**Prompt
With Context**



+ meta Info

```
{
  "text": "The tallest penguins, Emperor penguins, live in Antarctica."
}
```

Command R+: RAG 및 추적성 특화 모델

RAG를 위한 Fine-Tuning & Citation 지원

Prompt

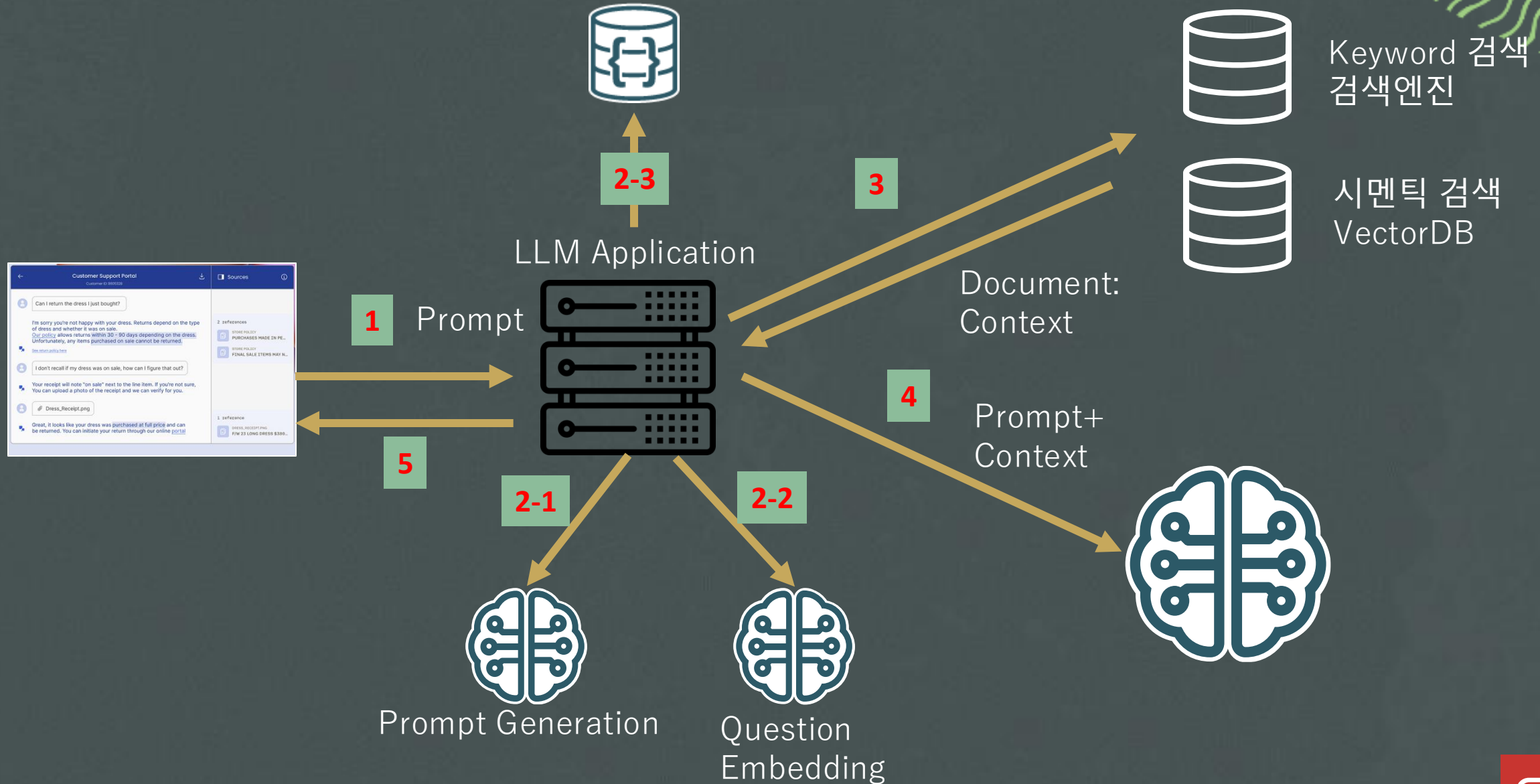
```
{ "message": "Where do the tallest penguins live?", "documents": [
  { "title": "Tall penguins",
    "snippet": "Emperor penguins are the tallest." },
  { "title": "Penguin habitats",
    "snippet": "Emperor penguins only live in Antarctica." },
  { "title": "What are animals?",
    "snippet": "Animals are different from plants." }
],
}
```



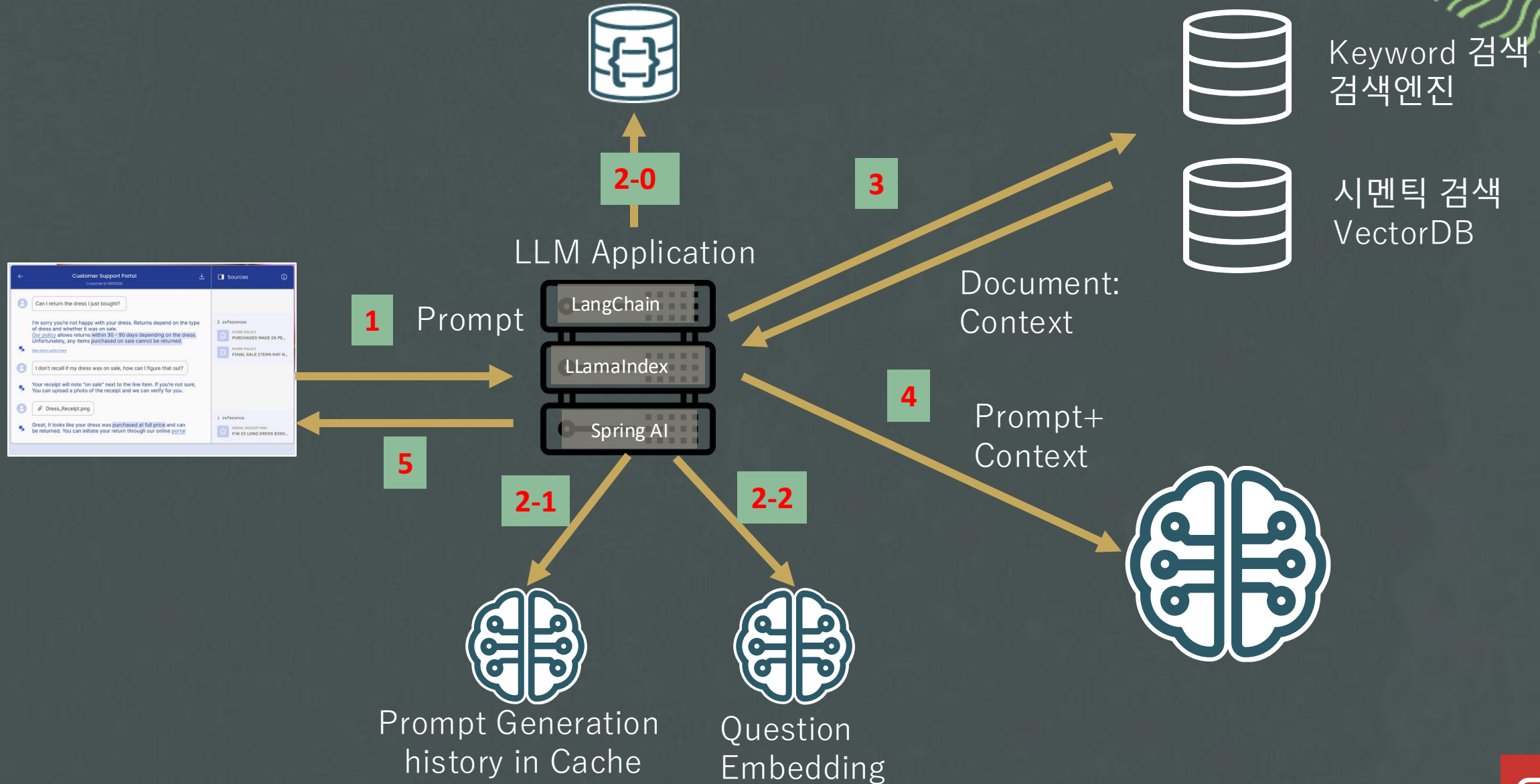
Completion

```
{
  "text": "<doc0>The tallest</doc0> <prompt>penguins</prompt>,
<doc0>Emperor penguins<doc0>, live in <doc1>Antarctica</doc1>."
}
```

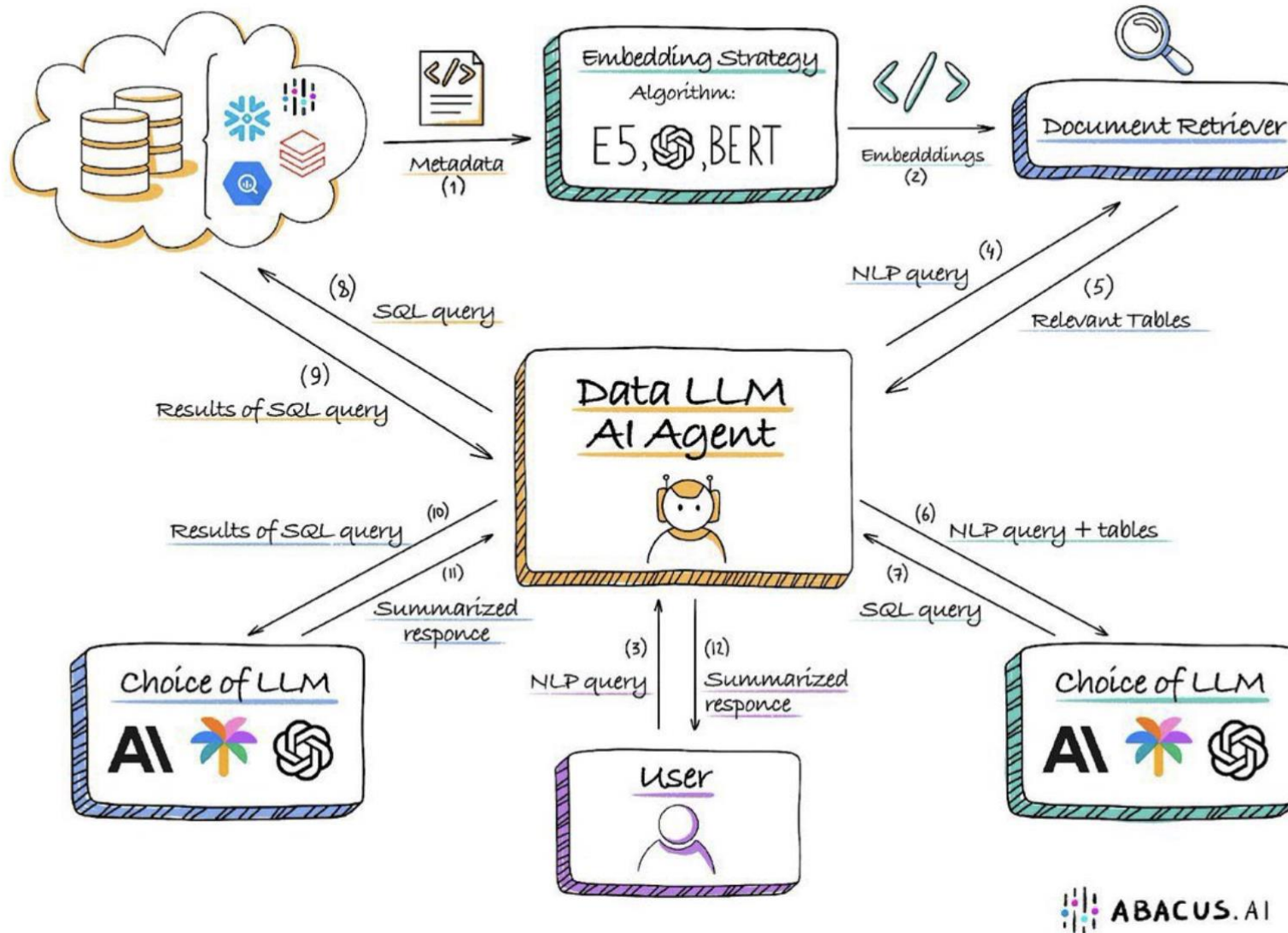
RAG(Retrieval Augmented Generation) 아키텍처



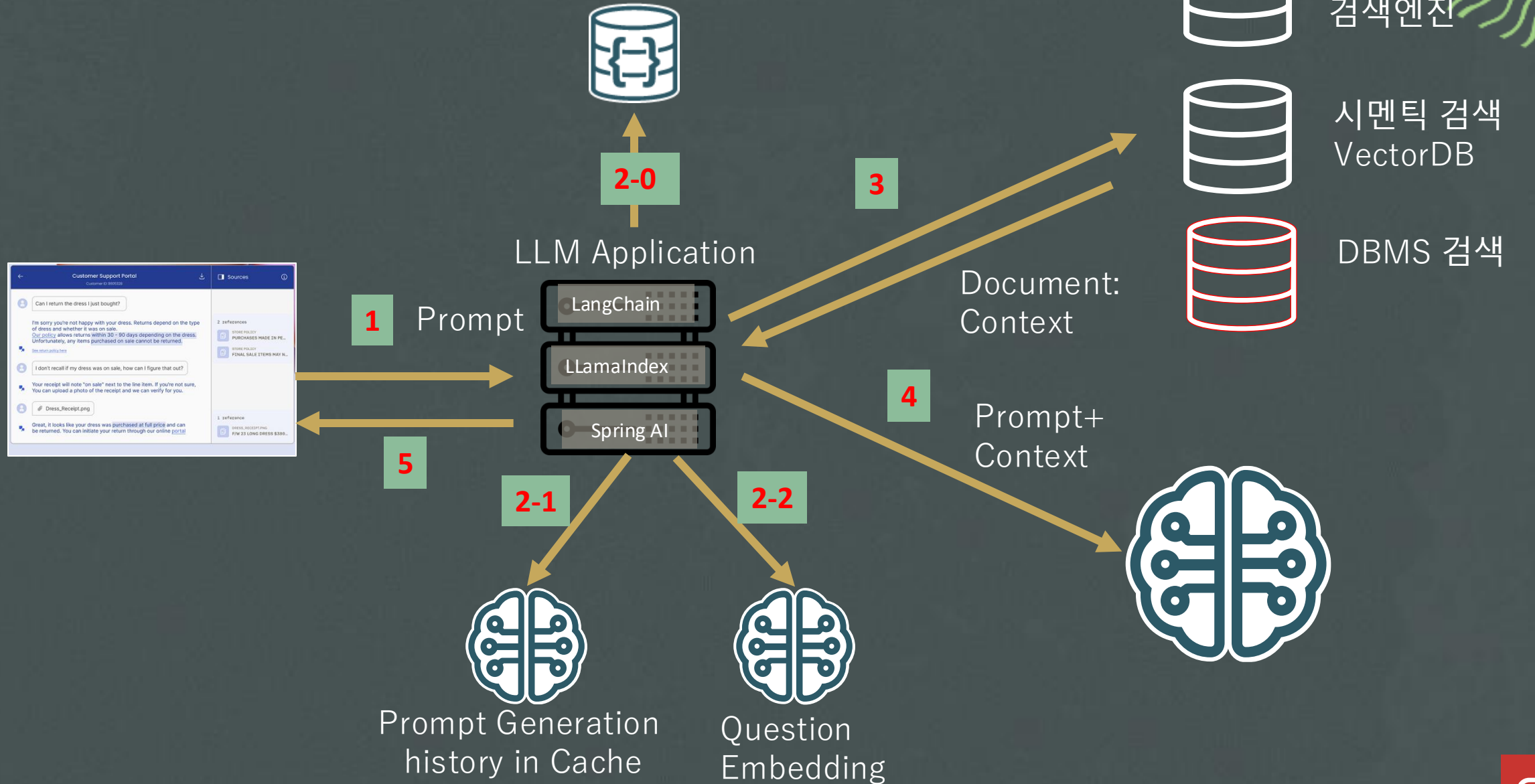
RAG(Retrieval Augmented Generation) 아키텍처



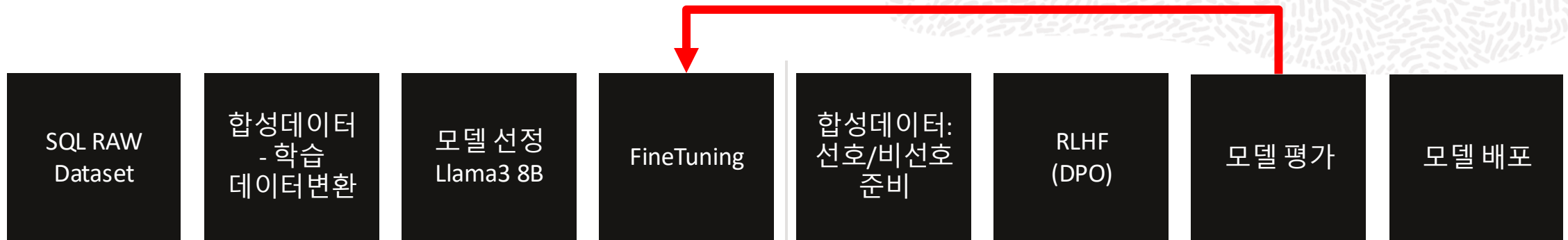
DataLLM - Get Insights From Your Data



RAG(Retrieval Augmented Generation) 아키텍처:



NL2SQL, Text to SQL



NL2SQL, Text to SQL



Hugging Face is way more fun with friends and colleagues! 🥳 [Join an organization](#) Dismiss this message

Datasets: Shritama/nl2sql like 1

[Dataset card](#) [Viewer](#) [Files](#) [Community](#) 1

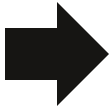
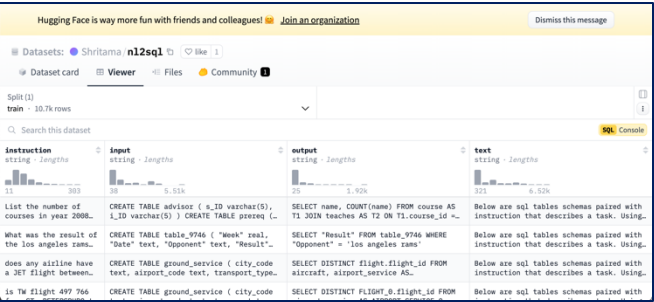
Split (1)
train · 10.7k rows

Search this dataset SQL Console

instruction string · lengths	input string · lengths	output string · lengths	text string · lengths
 11 303	 38 5.51k	 25 1.92k	 321 6.52k
List the number of courses in year 2008...	CREATE TABLE advisor (s_ID varchar(5), i_ID varchar(5)) CREATE TABLE prereq (...	SELECT name, COUNT(name) FROM course AS T1 JOIN teaches AS T2 ON T1.course_id =...	Below are sql tables schemas paired with instruction that describes a task. Using...
What was the result of the los angeles rams...	CREATE TABLE table_9746 ("Week" real, "Date" text, "Opponent" text, "Result"...	SELECT "Result" FROM table_9746 WHERE "Opponent" = 'los angeles rams'	Below are sql tables schemas paired with instruction that describes a task. Using...
does any airline have a JET flight between...	CREATE TABLE ground_service (city_code text, airport_code text, transport_type...	SELECT DISTINCT flight.flight_id FROM aircraft, airport_service AS...	Below are sql tables schemas paired with instruction that describes a task. Using...
is TW flight 497 766	CREATE TABLE ground_service (city_code	SELECT DISTINCT FLIGHT_0.flight_id FROM	Below are sql tables schemas paired with



NL2SQL, Text to SQL



Instruction => 한글
ANSI SQL => 대상 DB SQL
DDL => 대상 DB DDL

Instruction, SQL, DDL



NL2SQL, Text to SQL



Hugging Face Search models, datasets, u: Models Datasets Spaces Posts Docs Pricing

Hugging Face is way more fun with friends and colleagues! Join an organization Dismiss this message

Tasks Libraries Datasets Languages Licenses Other

Filter Tasks by name

Multimodal

- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering
- Video-Text-to-Text
- Any-to-Any

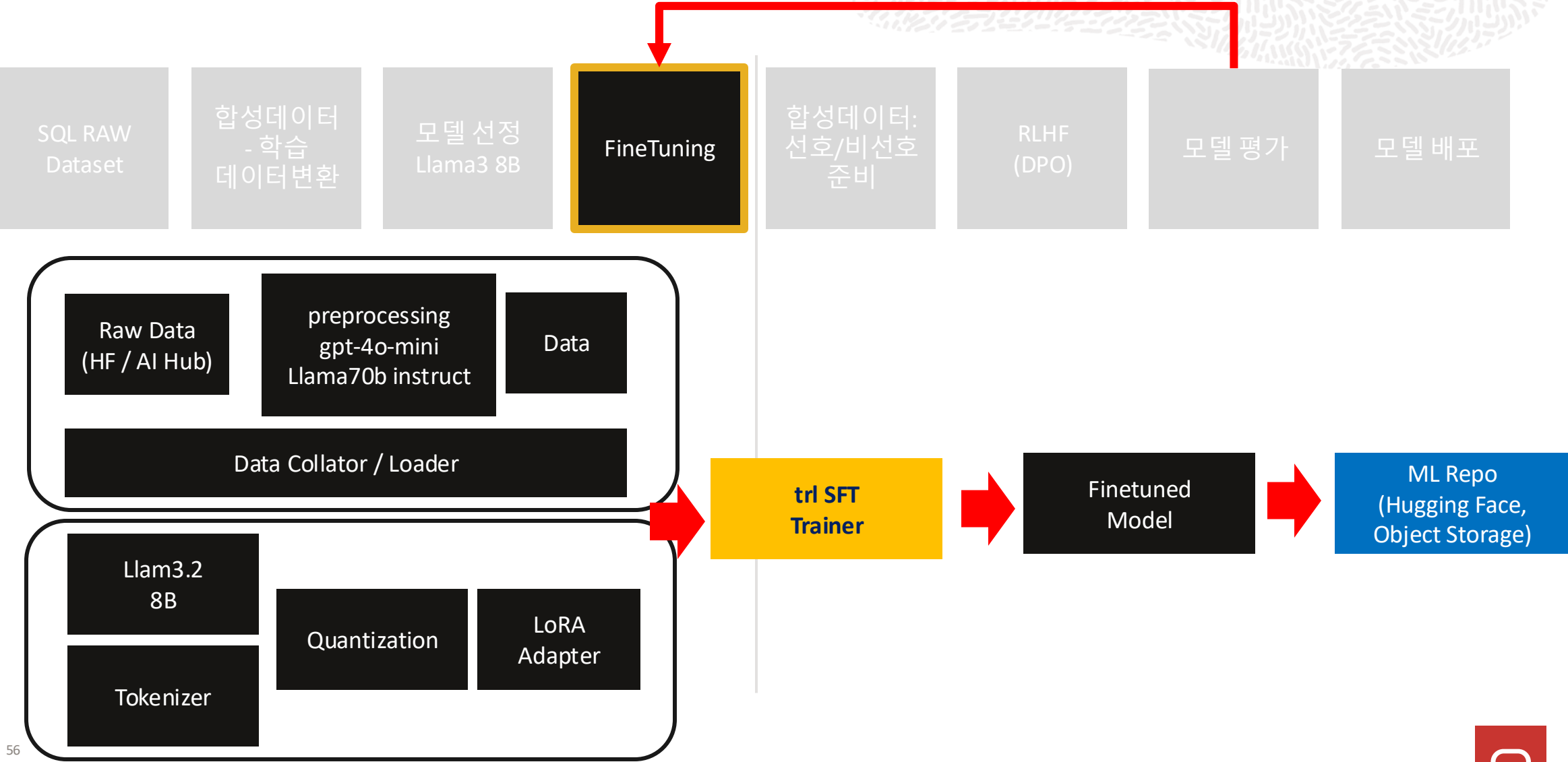
Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation

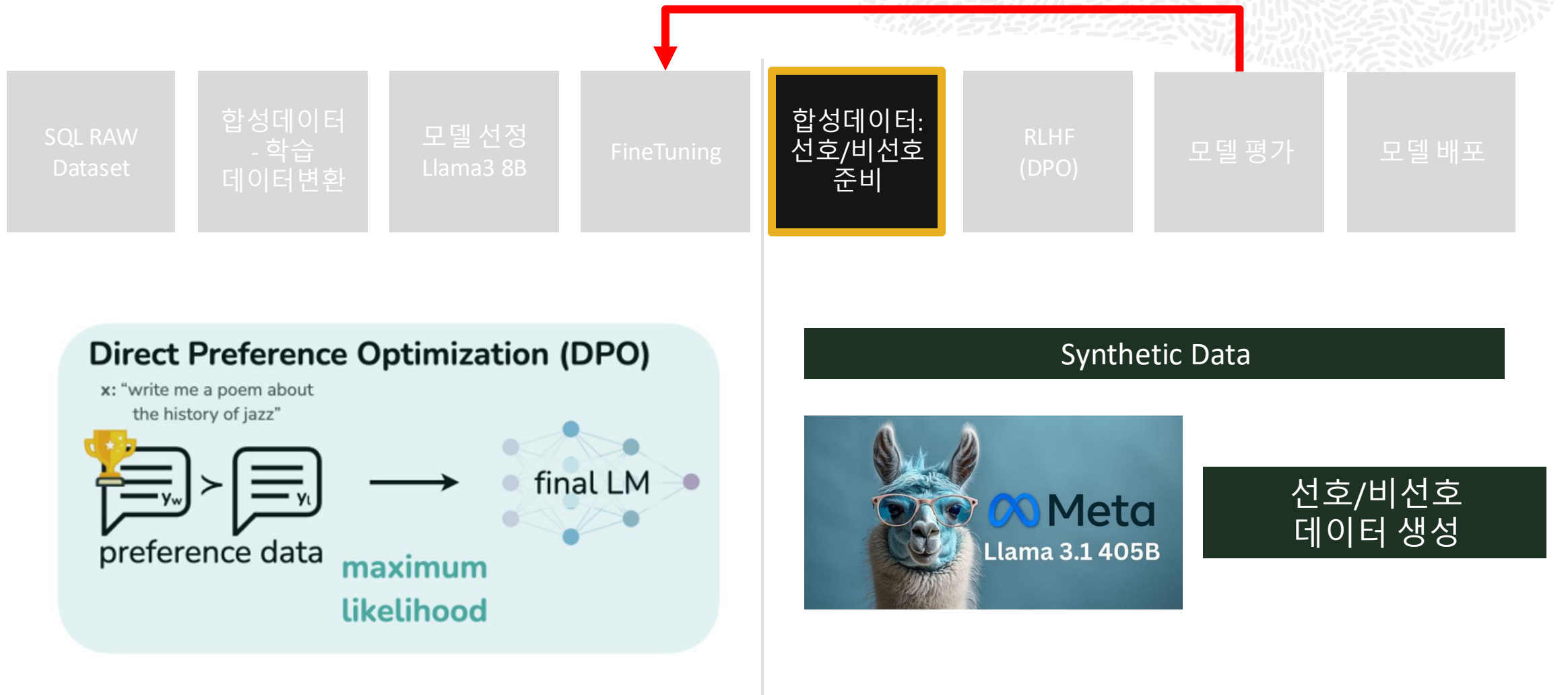
Models 20,467 llama3 Full-text search Sort: Trending

- meta-llama/Llama-3.2-1B**
Text Generation • Updated 14 days ago • 270k • 495
- meta-llama/Llama-3.2-11B-Vision-Instruct**
Image-Text-to-Text • Updated 15 days ago • 776k • 649
- meta-llama/Llama-3.1-8B-Instruct**
Text Generation • Updated 19 days ago • 2.67M • 2.78k
- meta-llama/Llama-3.2-3B-Instruct**
Text Generation • Updated 19 days ago • 437k • 335

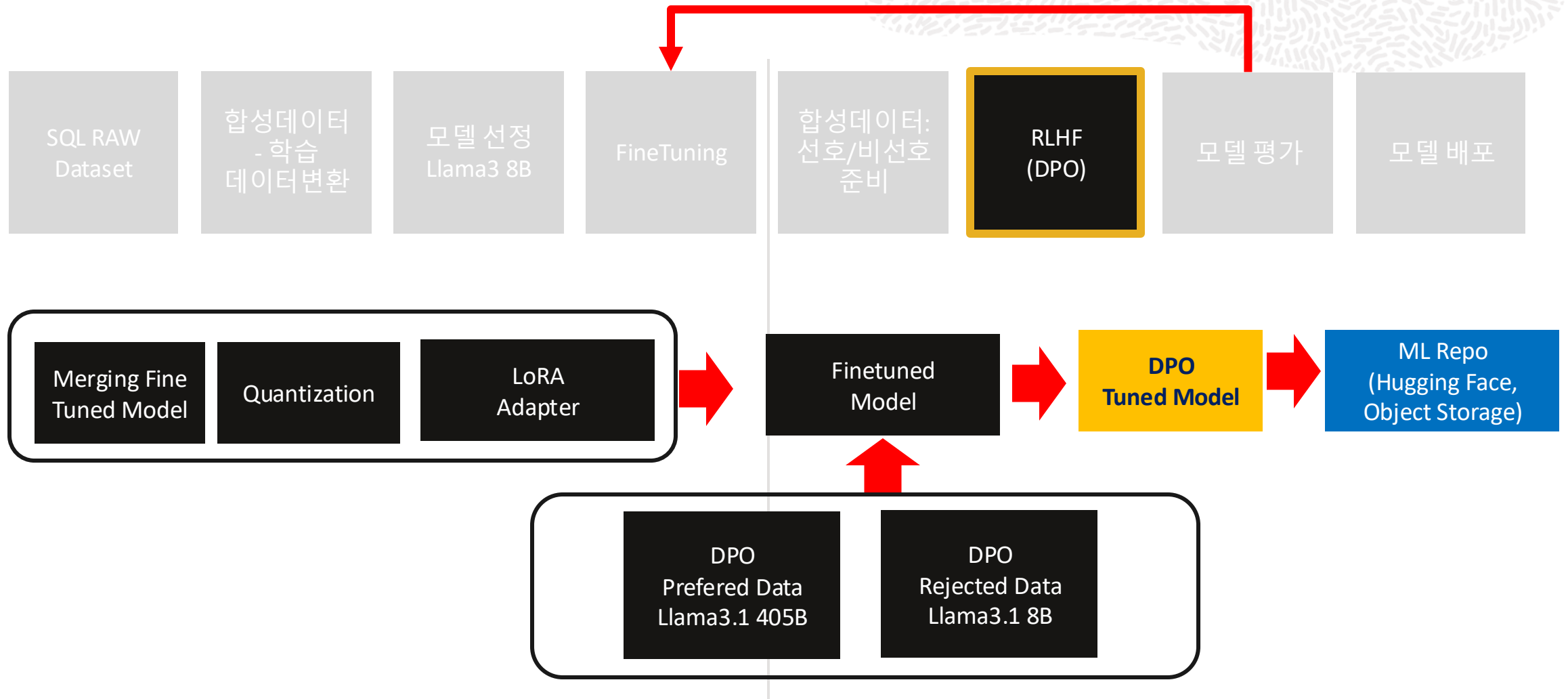
NL2SQL, Text to SQL



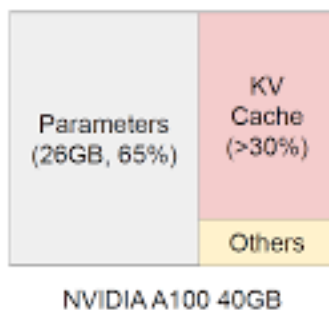
NL2SQL, Text to SQL



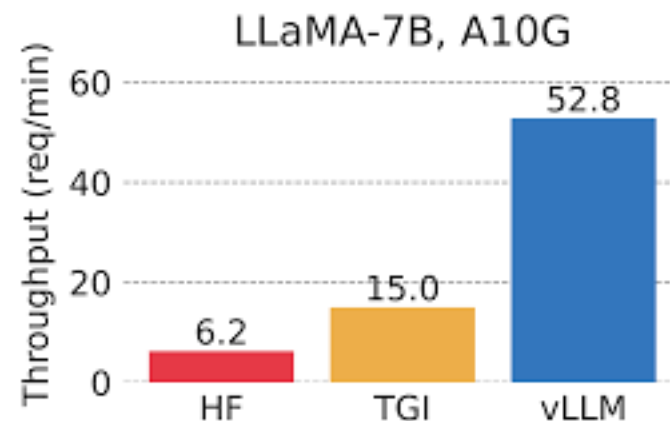
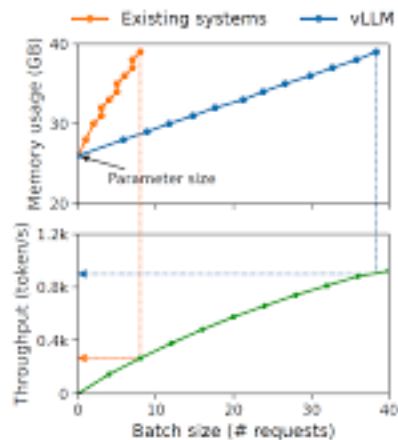
NL2SQL, Text to SQL



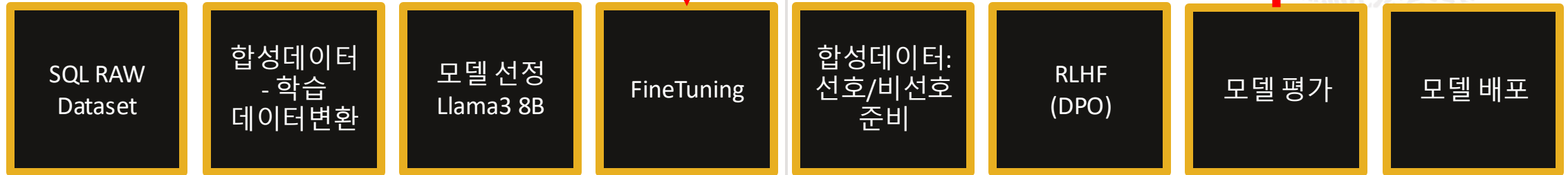
NL2SQL, Text to SQL



NVIDIA A100 40GB



NL2SQL, Text to SQL



File Edit View Run Kernel Tabs Settings Help

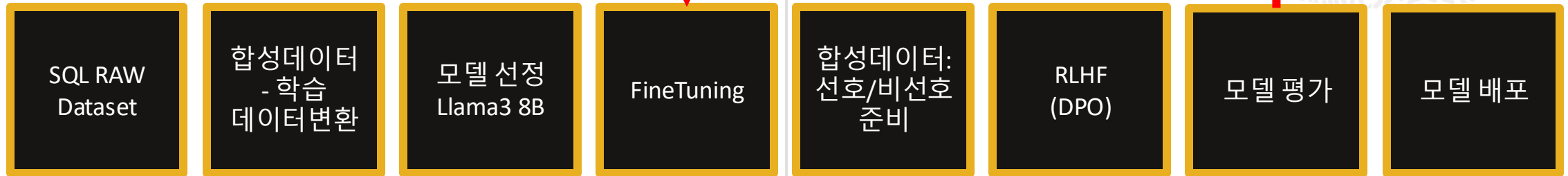
Launcher

Filter files by name

Name	Last Modified
checkpoint-100	21 days ago
checkpoint-200	21 days ago
checkpoint-300	21 days ago
checkpoint-400	21 days ago
trainer_log.jsonl	21 days ago

1 {"current_steps": 10, "total_steps": 1560, "loss": 0.5407, "learning_rate": 9.998986144924251e-05, "epoch": 0.064, "percentage": 0.64, "elapsed_time": "0:01:02", "remaining_time": "2:41:22"}
2 {"current_steps": 20, "total_steps": 1560, "loss": 0.2536, "learning_rate": 9.995944998057849e-05, "epoch": 0.128, "percentage": 1.28, "elapsed_time": "0:01:57", "remaining_time": "2:18:37"}
3 {"current_steps": 30, "total_steps": 1560, "loss": 0.2432, "learning_rate": 9.99087777116589e-05, "epoch": 0.192, "percentage": 1.92, "elapsed_time": "0:02:52", "remaining_time": "2:12:01"}
4 {"current_steps": 40, "total_steps": 1560, "loss": 0.2148, "learning_rate": 9.983786540671051e-05, "epoch": 0.256, "percentage": 2.56, "elapsed_time": "0:03:48", "remaining_time": "2:12:33"}
5 {"current_steps": 50, "total_steps": 1560, "loss": 0.2222, "learning_rate": 9.974674175313228e-05, "epoch": 0.32, "percentage": 3.2, "elapsed_time": "0:04:43", "remaining_time": "2:12:33"}
6 {"current_steps": 60, "total_steps": 1560, "loss": 0.1941, "learning_rate": 9.96354437049827e-05, "epoch": 0.384, "percentage": 3.85, "elapsed_time": "0:05:37", "remaining_time": "2:12:29"}
7 {"current_steps": 70, "total_steps": 1560, "loss": 0.2083, "learning_rate": 9.95048163980582e-05, "epoch": 0.448, "percentage": 4.49, "elapsed_time": "0:06:31", "remaining_time": "2:10:53"}
8 {"current_steps": 80, "total_steps": 1560, "loss": 0.2212, "learning_rate": 9.93525131189564e-05, "epoch": 0.512, "percentage": 5.13, "elapsed_time": "0:07:28", "remaining_time": "2:10:08"}
9 {"current_steps": 90, "total_steps": 1560, "loss": 0.2019, "learning_rate": 9.918099534735718e-05, "epoch": 0.576, "percentage": 5.77, "elapsed_time": "0:08:22", "remaining_time": "2:10:40"}
10 {"current_steps": 100, "total_steps": 1560, "loss": 0.2155, "learning_rate": 9.898953268211338e-05, "epoch": 0.64, "percentage": 6.41, "elapsed_time": "0:09:16", "remaining_time": "2:15:23"}
11 {"current_steps": 110, "total_steps": 1560, "loss": 0.2185, "learning_rate": 9.877826254235471e-05, "epoch": 0.704, "percentage": 7.05, "elapsed_time": "0:10:18", "remaining_time": "2:15:58"}
12 {"current_steps": 120, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.85470908713826e-05, "epoch": 0.768, "percentage": 7.69, "elapsed_time": "0:11:12", "remaining_time": "2:14:33"}
13 {"current_steps": 130, "total_steps": 1560, "loss": 0.2257, "learning_rate": 9.829629131445342e-05, "epoch": 0.832, "percentage": 8.33, "elapsed_time": "0:12:07", "remaining_time": "2:13:18"}
14 {"current_steps": 140, "total_steps": 1560, "loss": 0.1964, "learning_rate": 9.80259858156862e-05, "epoch": 0.896, "percentage": 8.97, "elapsed_time": "0:13:02", "remaining_time": "2:12:13"}
15 {"current_steps": 150, "total_steps": 1560, "loss": 0.2171, "learning_rate": 9.77360432542729e-05, "epoch": 0.96, "percentage": 9.62, "elapsed_time": "0:13:56", "remaining_time": "2:11:05"}
16 {"current_steps": 160, "total_steps": 1560, "loss": 0.2095, "learning_rate": 9.746741753132228e-05, "epoch": 1.024, "percentage": 10.24, "elapsed_time": "0:14:50", "remaining_time": "2:09:57"}
17 {"current_steps": 170, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.721629131445342e-05, "epoch": 1.088, "percentage": 10.88, "elapsed_time": "0:15:44", "remaining_time": "2:08:53"}
18 {"current_steps": 180, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.696516516516516e-05, "epoch": 1.152, "percentage": 11.52, "elapsed_time": "0:16:38", "remaining_time": "2:07:47"}
19 {"current_steps": 190, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.671403903903904e-05, "epoch": 1.216, "percentage": 12.16, "elapsed_time": "0:17:32", "remaining_time": "2:06:42"}
20 {"current_steps": 200, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.64629131445342e-05, "epoch": 1.28, "percentage": 12.8, "elapsed_time": "0:18:26", "remaining_time": "2:05:39"}
21 {"current_steps": 210, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.621178701178701e-05, "epoch": 1.344, "percentage": 13.44, "elapsed_time": "0:19:20", "remaining_time": "2:04:35"}
22 {"current_steps": 220, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.596066086608661e-05, "epoch": 1.408, "percentage": 14.08, "elapsed_time": "0:20:14", "remaining_time": "2:03:31"}
23 {"current_steps": 230, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.570953473571871e-05, "epoch": 1.472, "percentage": 14.72, "elapsed_time": "0:21:08", "remaining_time": "2:02:27"}
24 {"current_steps": 240, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.545840860860861e-05, "epoch": 1.536, "percentage": 15.36, "elapsed_time": "0:22:02", "remaining_time": "2:01:23"}
25 {"current_steps": 250, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.520728247824782e-05, "epoch": 1.6, "percentage": 16.0, "elapsed_time": "0:22:56", "remaining_time": "2:00:19"}
26 {"current_steps": 260, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.495615634956156e-05, "epoch": 1.664, "percentage": 16.64, "elapsed_time": "0:23:50", "remaining_time": "1:59:15"}
27 {"current_steps": 270, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.47050302030203e-05, "epoch": 1.728, "percentage": 17.28, "elapsed_time": "0:24:44", "remaining_time": "1:58:11"}
28 {"current_steps": 280, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.445390407272727e-05, "epoch": 1.792, "percentage": 17.92, "elapsed_time": "0:25:38", "remaining_time": "1:57:07"}
29 {"current_steps": 290, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.420277794242424e-05, "epoch": 1.856, "percentage": 18.56, "elapsed_time": "0:26:32", "remaining_time": "1:56:03"}
30 {"current_steps": 300, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.395165181181182e-05, "epoch": 1.92, "percentage": 19.2, "elapsed_time": "0:27:26", "remaining_time": "1:54:59"}
31 {"current_steps": 310, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.370052568156816e-05, "epoch": 1.984, "percentage": 19.84, "elapsed_time": "0:28:20", "remaining_time": "1:53:55"}
32 {"current_steps": 320, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.344940055151515e-05, "epoch": 2.048, "percentage": 20.48, "elapsed_time": "0:29:14", "remaining_time": "1:52:51"}
33 {"current_steps": 330, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.319827442127442e-05, "epoch": 2.112, "percentage": 21.12, "elapsed_time": "0:30:08", "remaining_time": "1:51:47"}
34 {"current_steps": 340, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.294714829102947e-05, "epoch": 2.176, "percentage": 21.76, "elapsed_time": "0:31:02", "remaining_time": "1:50:43"}
35 {"current_steps": 350, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.269602216078261e-05, "epoch": 2.24, "percentage": 22.4, "elapsed_time": "0:31:56", "remaining_time": "1:49:39"}
36 {"current_steps": 360, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.244489603053571e-05, "epoch": 2.304, "percentage": 23.04, "elapsed_time": "0:32:50", "remaining_time": "1:48:35"}
37 {"current_steps": 370, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.219377090028881e-05, "epoch": 2.368, "percentage": 23.68, "elapsed_time": "0:33:44", "remaining_time": "1:47:31"}
38 {"current_steps": 380, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.194264477004191e-05, "epoch": 2.432, "percentage": 24.32, "elapsed_time": "0:34:38", "remaining_time": "1:46:27"}
39 {"current_steps": 390, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.169151863979501e-05, "epoch": 2.496, "percentage": 24.96, "elapsed_time": "0:35:32", "remaining_time": "1:45:23"}
40 {"current_steps": 400, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.144039250954811e-05, "epoch": 2.56, "percentage": 25.6, "elapsed_time": "0:36:26", "remaining_time": "1:44:19"}
41 {"current_steps": 410, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.118926637930121e-05, "epoch": 2.624, "percentage": 26.24, "elapsed_time": "0:37:20", "remaining_time": "1:43:15"}
42 {"current_steps": 420, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.093814024905431e-05, "epoch": 2.688, "percentage": 26.88, "elapsed_time": "0:38:14", "remaining_time": "1:42:11"}
43 {"current_steps": 430, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.068701411880741e-05, "epoch": 2.752, "percentage": 27.52, "elapsed_time": "0:39:08", "remaining_time": "1:41:07"}
44 {"current_steps": 440, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.043588808856051e-05, "epoch": 2.816, "percentage": 28.16, "elapsed_time": "0:40:02", "remaining_time": "1:40:03"}
45 {"current_steps": 450, "total_steps": 1560, "loss": 0.2086, "learning_rate": 9.018476195831361e-05, "epoch": 2.88, "percentage": 28.8, "elapsed_time": "0:40:56", "remaining_time": "1:38:59"}
46 {"current_steps": 460, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.993363582806671e-05, "epoch": 2.944, "percentage": 29.44, "elapsed_time": "0:41:50", "remaining_time": "1:37:55"}
47 {"current_steps": 470, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.968250969781981e-05, "epoch": 3.008, "percentage": 30.08, "elapsed_time": "0:42:44", "remaining_time": "1:36:51"}
48 {"current_steps": 480, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.943138356757291e-05, "epoch": 3.072, "percentage": 30.72, "elapsed_time": "0:43:38", "remaining_time": "1:35:47"}
49 {"current_steps": 490, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.918025743732601e-05, "epoch": 3.136, "percentage": 31.36, "elapsed_time": "0:44:32", "remaining_time": "1:34:43"}
50 {"current_steps": 500, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.892913130707911e-05, "epoch": 3.2, "percentage": 32.0, "elapsed_time": "0:45:26", "remaining_time": "1:33:39"}
51 {"current_steps": 510, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.867800517683221e-05, "epoch": 3.264, "percentage": 32.64, "elapsed_time": "0:46:20", "remaining_time": "1:32:35"}
52 {"current_steps": 520, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.842687904658531e-05, "epoch": 3.328, "percentage": 33.28, "elapsed_time": "0:47:14", "remaining_time": "1:31:31"}
53 {"current_steps": 530, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.817575291633841e-05, "epoch": 3.392, "percentage": 33.92, "elapsed_time": "0:48:08", "remaining_time": "1:30:27"}
54 {"current_steps": 540, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.792462678609151e-05, "epoch": 3.456, "percentage": 34.56, "elapsed_time": "0:49:02", "remaining_time": "1:29:23"}
55 {"current_steps": 550, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.767350065584461e-05, "epoch": 3.52, "percentage": 35.2, "elapsed_time": "0:49:56", "remaining_time": "1:28:19"}
56 {"current_steps": 560, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.742237452559771e-05, "epoch": 3.584, "percentage": 35.84, "elapsed_time": "0:50:50", "remaining_time": "1:27:15"}
57 {"current_steps": 570, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.717124839535081e-05, "epoch": 3.648, "percentage": 36.48, "elapsed_time": "0:51:44", "remaining_time": "1:26:11"}
58 {"current_steps": 580, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.692012226510391e-05, "epoch": 3.712, "percentage": 37.12, "elapsed_time": "0:52:38", "remaining_time": "1:25:07"}
59 {"current_steps": 590, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.666899613485701e-05, "epoch": 3.776, "percentage": 37.76, "elapsed_time": "0:53:32", "remaining_time": "1:24:03"}
60 {"current_steps": 600, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.641787000461011e-05, "epoch": 3.84, "percentage": 38.4, "elapsed_time": "0:54:26", "remaining_time": "1:22:59"}
61 {"current_steps": 610, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.616674387436321e-05, "epoch": 3.904, "percentage": 39.04, "elapsed_time": "0:55:20", "remaining_time": "1:21:55"}
62 {"current_steps": 620, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.591561774411631e-05, "epoch": 3.968, "percentage": 39.68, "elapsed_time": "0:56:14", "remaining_time": "1:20:51"}
63 {"current_steps": 630, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.566449161386941e-05, "epoch": 4.032, "percentage": 40.32, "elapsed_time": "0:57:08", "remaining_time": "1:19:47"}
64 {"current_steps": 640, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.541336548362251e-05, "epoch": 4.096, "percentage": 40.96, "elapsed_time": "0:58:02", "remaining_time": "1:18:43"}
65 {"current_steps": 650, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.516223935337561e-05, "epoch": 4.16, "percentage": 41.6, "elapsed_time": "0:58:56", "remaining_time": "1:17:39"}
66 {"current_steps": 660, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.491111322312871e-05, "epoch": 4.224, "percentage": 42.24, "elapsed_time": "0:59:50", "remaining_time": "1:16:35"}
67 {"current_steps": 670, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.466008709288181e-05, "epoch": 4.288, "percentage": 42.88, "elapsed_time": "1:00:44", "remaining_time": "1:15:31"}
68 {"current_steps": 680, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.440896096263491e-05, "epoch": 4.352, "percentage": 43.52, "elapsed_time": "1:01:38", "remaining_time": "1:14:27"}
69 {"current_steps": 690, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.415783483238801e-05, "epoch": 4.416, "percentage": 44.16, "elapsed_time": "1:02:32", "remaining_time": "1:13:23"}
70 {"current_steps": 700, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.390670870214111e-05, "epoch": 4.48, "percentage": 44.8, "elapsed_time": "1:03:26", "remaining_time": "1:12:19"}
71 {"current_steps": 710, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.365558257189421e-05, "epoch": 4.544, "percentage": 45.44, "elapsed_time": "1:04:20", "remaining_time": "1:11:15"}
72 {"current_steps": 720, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.340445644164731e-05, "epoch": 4.608, "percentage": 46.08, "elapsed_time": "1:05:14", "remaining_time": "1:10:11"}
73 {"current_steps": 730, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.315333031140041e-05, "epoch": 4.672, "percentage": 46.72, "elapsed_time": "1:06:08", "remaining_time": "1:09:07"}
74 {"current_steps": 740, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.290220418115351e-05, "epoch": 4.736, "percentage": 47.36, "elapsed_time": "1:07:02", "remaining_time": "1:08:03"}
75 {"current_steps": 750, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.265107805090661e-05, "epoch": 4.8, "percentage": 48.0, "elapsed_time": "1:07:56", "remaining_time": "1:06:59"}
76 {"current_steps": 760, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.240000192065971e-05, "epoch": 4.864, "percentage": 48.64, "elapsed_time": "1:08:50", "remaining_time": "1:05:55"}
77 {"current_steps": 770, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.214887579041281e-05, "epoch": 4.928, "percentage": 49.28, "elapsed_time": "1:09:44", "remaining_time": "1:04:51"}
78 {"current_steps": 780, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.189774966016591e-05, "epoch": 4.992, "percentage": 49.92, "elapsed_time": "1:10:38", "remaining_time": "1:03:47"}
79 {"current_steps": 790, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.164662352991901e-05, "epoch": 5.056, "percentage": 50.56, "elapsed_time": "1:11:32", "remaining_time": "1:02:43"}
80 {"current_steps": 800, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.139549739967211e-05, "epoch": 5.12, "percentage": 51.2, "elapsed_time": "1:12:26", "remaining_time": "1:01:39"}
81 {"current_steps": 810, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.114437126942521e-05, "epoch": 5.184, "percentage": 51.84, "elapsed_time": "1:13:20", "remaining_time": "1:00:35"}
82 {"current_steps": 820, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.089324513917831e-05, "epoch": 5.248, "percentage": 52.48, "elapsed_time": "1:14:14", "remaining_time": "0:59:31"}
83 {"current_steps": 830, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.064211900893141e-05, "epoch": 5.312, "percentage": 53.12, "elapsed_time": "1:15:08", "remaining_time": "0:58:27"}
84 {"current_steps": 840, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.039104287868451e-05, "epoch": 5.376, "percentage": 53.76, "elapsed_time": "1:16:02", "remaining_time": "0:57:23"}
85 {"current_steps": 850, "total_steps": 1560, "loss": 0.2086, "learning_rate": 8.013996674843761e-05, "epoch": 5.44, "percentage": 54.4, "elapsed_time": "1:16:56", "remaining_time": "0:56:19"}
86 {"current_steps": 860, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.988889061819071e-05, "epoch": 5.504, "percentage": 55.04, "elapsed_time": "1:17:50", "remaining_time": "0:55:15"}
87 {"current_steps": 870, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.963781448794381e-05, "epoch": 5.568, "percentage": 55.68, "elapsed_time": "1:18:44", "remaining_time": "0:54:11"}
88 {"current_steps": 880, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.938673835769691e-05, "epoch": 5.632, "percentage": 56.32, "elapsed_time": "1:19:38", "remaining_time": "0:53:07"}
89 {"current_steps": 890, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.913566222745001e-05, "epoch": 5.696, "percentage": 56.96, "elapsed_time": "1:20:32", "remaining_time": "0:52:03"}
90 {"current_steps": 900, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.888458609720311e-05, "epoch": 5.76, "percentage": 57.6, "elapsed_time": "1:21:26", "remaining_time": "0:50:59"}
91 {"current_steps": 910, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.863350996695621e-05, "epoch": 5.824, "percentage": 58.24, "elapsed_time": "1:22:20", "remaining_time": "0:49:55"}
92 {"current_steps": 920, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.838243383670931e-05, "epoch": 5.888, "percentage": 58.88, "elapsed_time": "1:23:14", "remaining_time": "0:48:51"}
93 {"current_steps": 930, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.813135770646241e-05, "epoch": 5.952, "percentage": 59.52, "elapsed_time": "1:24:08", "remaining_time": "0:47:47"}
94 {"current_steps": 940, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.788028157621551e-05, "epoch": 6.016, "percentage": 60.16, "elapsed_time": "1:25:02", "remaining_time": "0:46:43"}
95 {"current_steps": 950, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.762920544596861e-05, "epoch": 6.08, "percentage": 60.8, "elapsed_time": "1:25:56", "remaining_time": "0:45:39"}
96 {"current_steps": 960, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.737812931572171e-05, "epoch": 6.144, "percentage": 61.44, "elapsed_time": "1:26:50", "remaining_time": "0:44:35"}
97 {"current_steps": 970, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.712705318547481e-05, "epoch": 6.208, "percentage": 62.08, "elapsed_time": "1:27:44", "remaining_time": "0:43:31"}
98 {"current_steps": 980, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.687597705522791e-05, "epoch": 6.272, "percentage": 62.72, "elapsed_time": "1:28:38", "remaining_time": "0:42:27"}
99 {"current_steps": 990, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.662490092498101e-05, "epoch": 6.336, "percentage": 63.36, "elapsed_time": "1:29:32", "remaining_time": "0:41:23"}
100 {"current_steps": 1000, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.637382479473411e-05, "epoch": 6.4, "percentage": 64.0, "elapsed_time": "1:30:26", "remaining_time": "0:40:19"}
101 {"current_steps": 1010, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.612274866448721e-05, "epoch": 6.464, "percentage": 64.64, "elapsed_time": "1:31:20", "remaining_time": "0:39:15"}
102 {"current_steps": 1020, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.587167253424031e-05, "epoch": 6.528, "percentage": 65.28, "elapsed_time": "1:32:14", "remaining_time": "0:38:11"}
103 {"current_steps": 1030, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.562059640399341e-05, "epoch": 6.592, "percentage": 65.92, "elapsed_time": "1:33:08", "remaining_time": "0:37:07"}
104 {"current_steps": 1040, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.536952027374651e-05, "epoch": 6.656, "percentage": 66.56, "elapsed_time": "1:34:02", "remaining_time": "0:36:03"}
105 {"current_steps": 1050, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.511844414349961e-05, "epoch": 6.72, "percentage": 67.2, "elapsed_time": "1:34:56", "remaining_time": "0:34:59"}
106 {"current_steps": 1060, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.486736801325271e-05, "epoch": 6.784, "percentage": 67.84, "elapsed_time": "1:35:50", "remaining_time": "0:33:55"}
107 {"current_steps": 1070, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.461629188300581e-05, "epoch": 6.848, "percentage": 68.48, "elapsed_time": "1:36:44", "remaining_time": "0:32:51"}
108 {"current_steps": 1080, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.436521575275891e-05, "epoch": 6.912, "percentage": 69.12, "elapsed_time": "1:37:38", "remaining_time": "0:31:47"}
109 {"current_steps": 1090, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.411413962251201e-05, "epoch": 6.976, "percentage": 69.76, "elapsed_time": "1:38:32", "remaining_time": "0:30:43"}
110 {"current_steps": 1100, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.386306349226511e-05, "epoch": 7.04, "percentage": 70.4, "elapsed_time": "1:39:26", "remaining_time": "0:29:39"}
111 {"current_steps": 1110, "total_steps": 1560, "loss": 0.2086, "learning_rate": 7.361198736201821e-05, "epoch":

NL2SQL, Text to SQL



query

유기농 제품에서 가장 높은 수익을 올린 판매자의 이름은 무엇인가요?

context

```
CREATE TABLE Vendors (VendorID INT, VendorName TEXT, Country TEXT);CREATE TABLE Products (ProductID INT, ProductName TEXT, Price DECIMAL, Organic BOOLEAN, VendorID INT); INSERT INTO Vendors VALUES (1, 'VendorF', 'UK'), (2, 'VendorG', 'UK'); INSERT INTO Products VALUES (1, 'Carrot', 0.6, true, 1), (2, 'Broccoli', 1.2, true, 1), (3, 'Apple', 1.5, true, 2), (4, 'Banana', 0.8, true, 2);
```

Clear

Submit

output

```
SELECT VendorName,
       SUM(Price * Organic) AS Revenue
FROM Vendors
JOIN Products ON Vendors.VendorID = Products.VendorID
WHERE Organic = TRUE
GROUP BY VendorName
ORDER BY Revenue DESC
LIMIT 1
```

Flag

NL2SQL, Text to SQL → N Step RAG

- 사용자 질의에서 관련 엔티티 추출
- 사용자 엔티티 메타 정보 제공
 - Schema Data: Table, Column, Comment, Table Partitioning....
 - Few Shot => Join 예시

query

유기농 제품에서 가장 높은 수익을 올린 판매자의 이름은 무엇인가요?

context

```
CREATE TABLE Vendors (VendorID INT, VendorName TEXT, Country TEXT);CREATE TABLE Products (ProductID INT, ProductName TEXT, Price DECIMAL, Organic BOOLEAN, VendorID INT); INSERT INTO Vendors VALUES (1, 'VendorF', 'UK'), (2, 'VendorG', 'UK'); INSERT INTO Products VALUES (1, 'Carrot', 0.6, true, 1), (2, 'Broccoli', 1.2, true, 1), (3, 'Apple', 1.5, true, 2), (4, 'Banana', 0.8, true, 2);
```

Clear

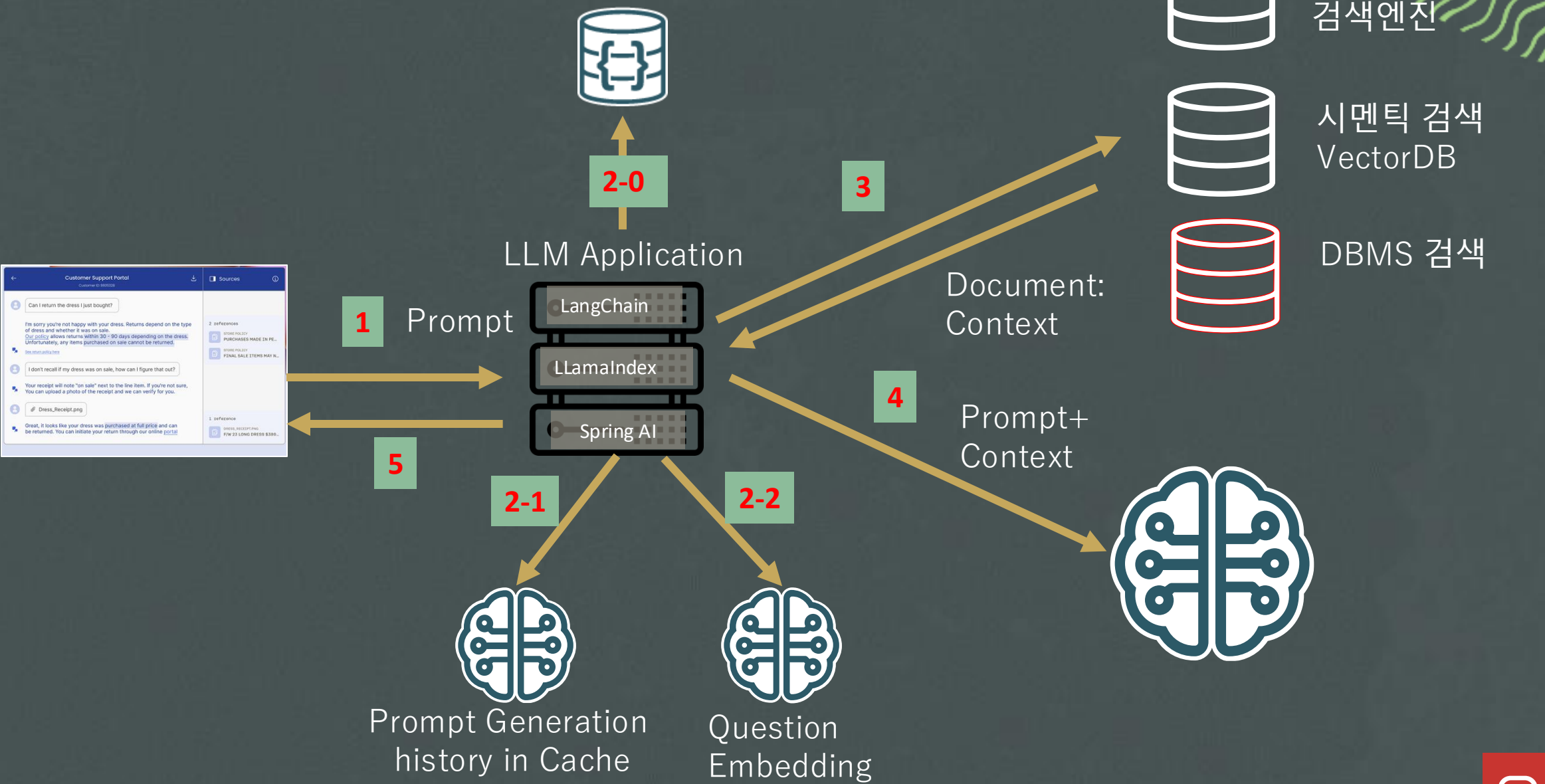
Submit

output

```
SELECT VendorName,  
       SUM(Price * Organic) AS Revenue  
FROM Vendors  
JOIN Products ON Vendors.VendorID = Products.VendorID  
WHERE Organic = TRUE  
GROUP BY VendorName  
ORDER BY Revenue DESC  
LIMIT 1
```

Flag

RAG(Retrieval Augmented Generation) 아키텍처:



Oracle의 전략 3

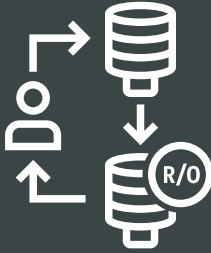
LLM & Human을 위한 데이터 베이스

Oracle Database 23ai : NEW LTS focused on AI

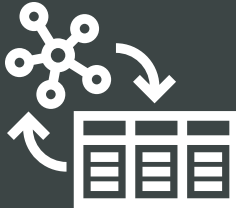


Data Use Case Domains

Schema Privileges



Readable PDB Standby



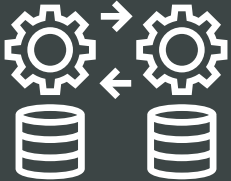
Operational Property Graphs

Real-time SQL Plan Management



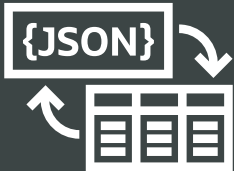
Lock-Free Reservations





Transactional Microservices

JSON / Relational Duality





AI Vector Search

True Cache





SQL Firewall

Priority Transactions



Rolling Patching



JavaScript Stored Procedures



Developer Role

Shrink Tablespace

Cross-tier Precheck Constratins



Globally Distributed Database w/ RAFT



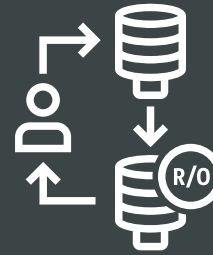
Oracle Database 23ai : NEW LTS focused on AI



Data Use Case
Domains

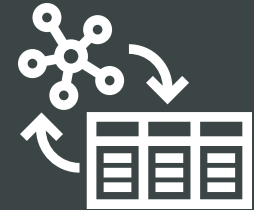
Schema
Privileges

ORACLE 23^{ai}
Database



Readable
PDB Standby

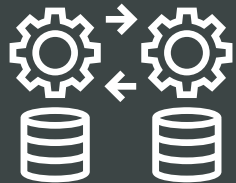
Operational
Property
Graphs



Real-time SQL Plan
Management

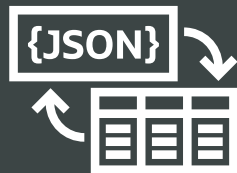


Lock-Free
Reservations



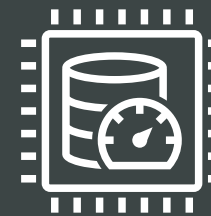
Transactional Microservices

JSON / Relational
Duality



AI Vector Search

True Cache



SQL Firewall

Priority Transactions

Globally Distributed
Database w/ RAFT



Rolling
Patching



JavaScript
Stored
Procedures



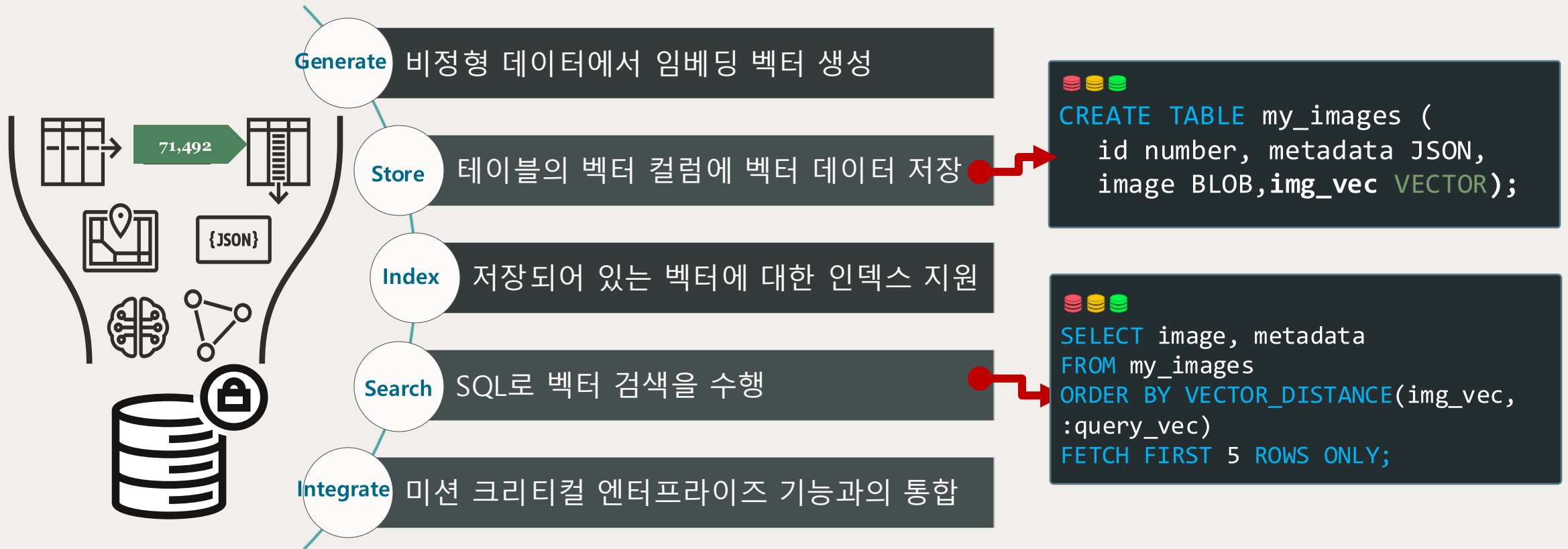
Developer Role

Shrink
Tablespace

Cross-tier
Precheck
Constratins

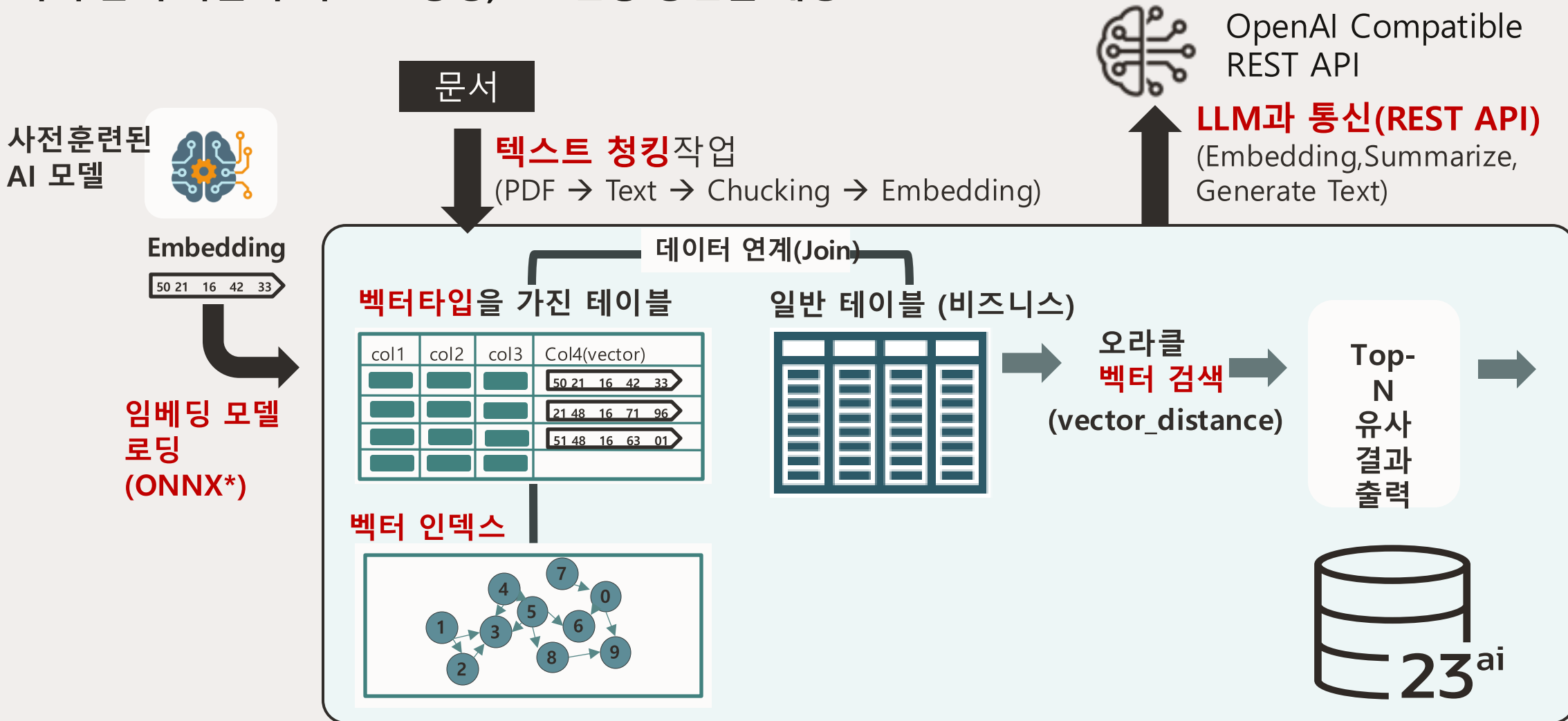
AI Vector Search 특징 For LLM

벡터 임베딩 부터 벡터 검색까지 End-to-End 관리 지원



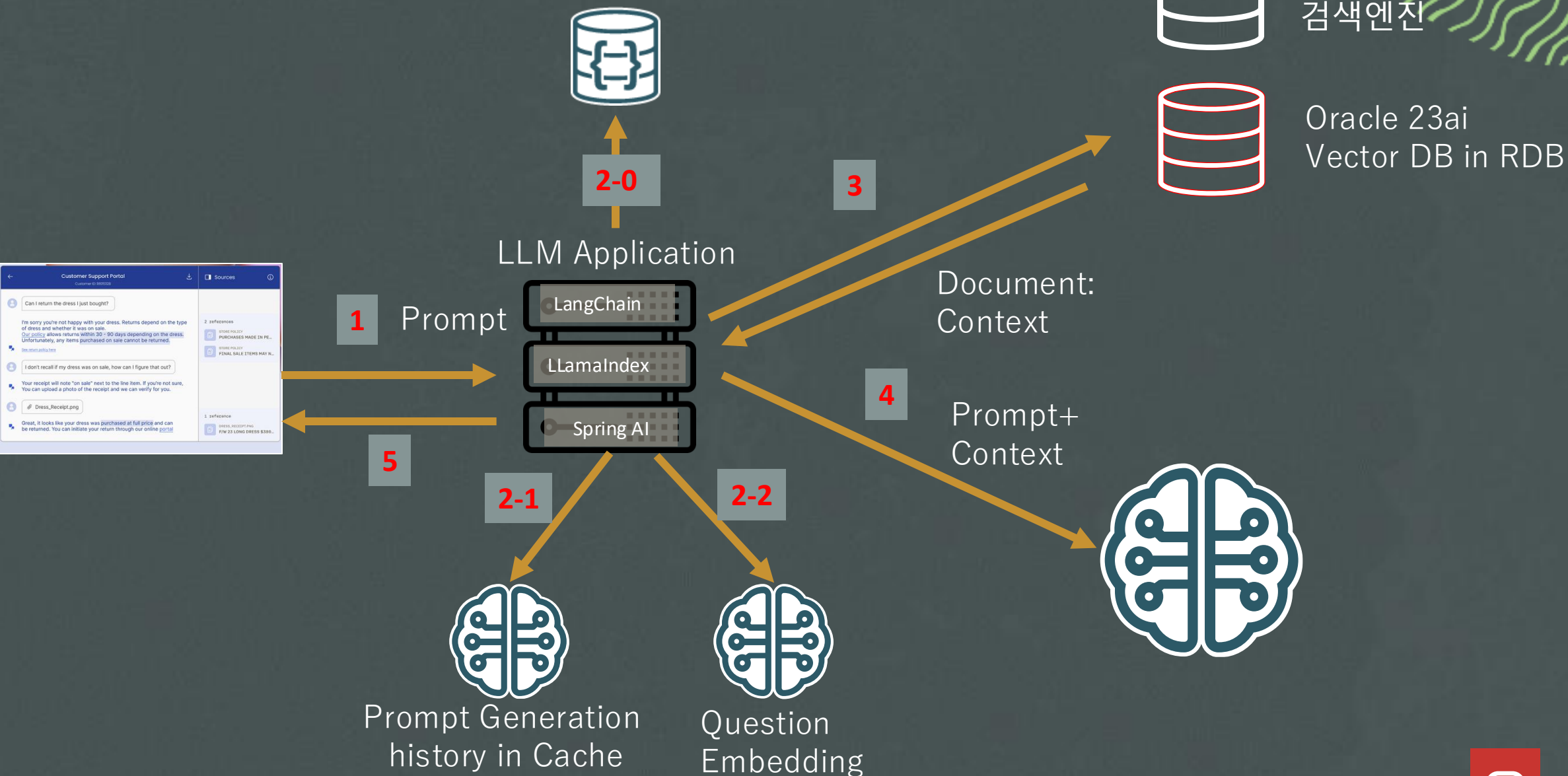
AI Vector Search 특징 For LLM

벡터 검색 기술과 텍스트 청킹, LLM연동 방안을 제공



*ONNX (Open Neural Network eXchange) : ML모델을 표현하는 오픈된 표준형식

RAG(Retrieval Augmented Generation) 아키텍처:

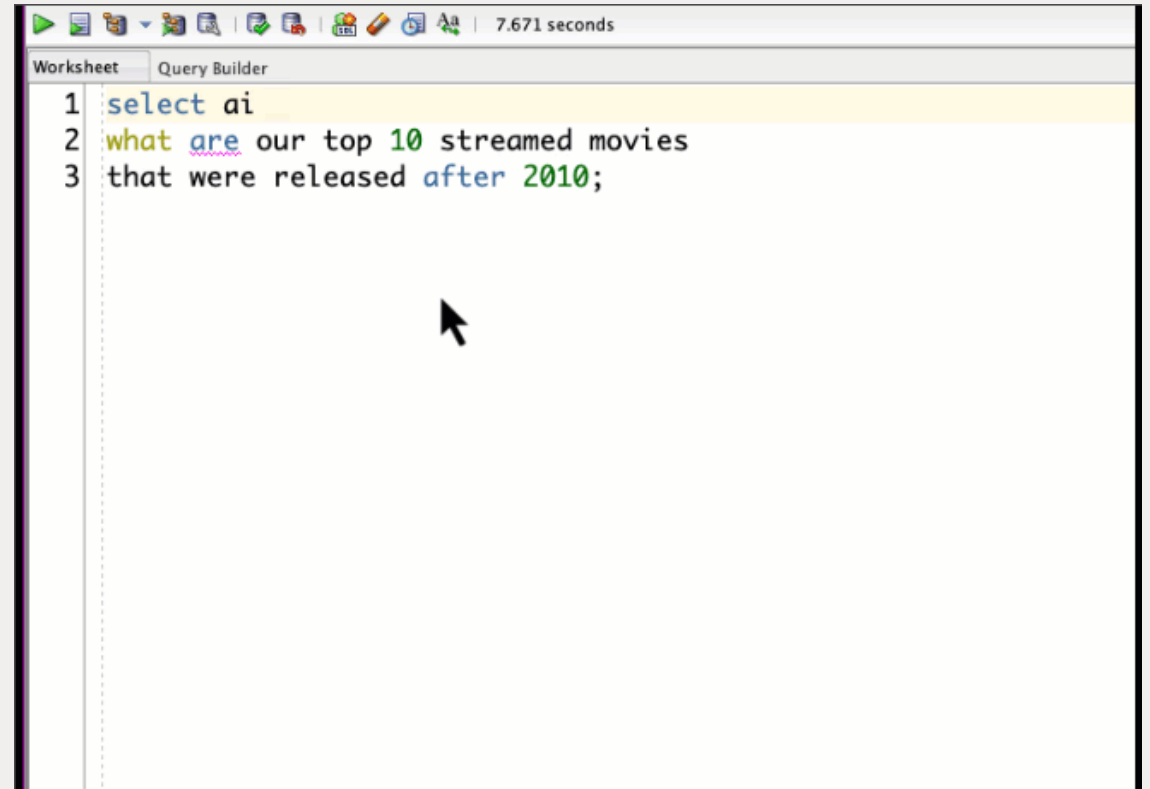


Select AI

자연어를 사용하여 SQL을 통해 데이터베이스 및 LLM과 상호 작용

- 직접 LLM과 자연어 기반으로 질문 응답
- RAG를 위해 스키마 메타데이터와 벡터 데이터베이스 활용
- 자연어로 SQL 쿼리 생성, 실행, 설명
- 다양한 AI 키워드 제공

(chat, narrate, showsql, runsql, explainsql)



The screenshot shows a software interface with a top toolbar and two tabs: 'Worksheet' and 'Query Builder'. The 'Query Builder' tab is active. It contains a list of three lines of text, each preceded by a number in a light blue box. Line 1 is 'select ai', line 2 is 'what are our top 10 streamed movies', and line 3 is 'that were released after 2010;'. The text is color-coded: 'select' is blue, 'ai' is black, 'what' is green, 'are' is blue, 'our' is black, 'top' is green, '10' is black, 'streamed' is black, 'movies' is black, 'that' is black, 'were' is black, 'released' is blue, 'after' is black, and '2010' is green. A mouse cursor is visible in the center of the text area. The top toolbar includes various icons for file operations, editing, and execution, along with a timer showing '7.671 seconds'.

```
1 select ai
2 what are our top 10 streamed movies
3 that were released after 2010;
```

Oracle Select AI

데모 순서

- ☐ 1-1. DB 접속 & Select AI 설정
- ☐ 1-2. Select AI 기초
- ☒ 1-3. SH 스키마 쿼리 데모
- ☐ 1-4. Movie 테이블 추가
- ☐ 1-5. Movie 프로필 등록
- ☐ 1-6. Movie 데이터 조회
- ☐ 2-1. Select AI RAG
- ☐ 3-1. NoCode RAG: OCI Gen AI Agent
- ☐ 4-1. Naive RAG With ADB & Cohere

SH Query 목록

2000년도의 총 판매량은 얼마인가요



SH 복합 쿼리 실행

- RUNSQL: 2000년도의 총 판매량은 얼마인가요

```
select ai runsql 2000년도의 총 판매량은 얼마인가요
```

	Total_Sales_Quantity
0	232,646

- SHOWSQL: 2000년도의 총 판매량은 얼마인가요

```
select ai showsql 2000년도의 총 판매량은 얼마인가요
```



	RESPONSE
0	SELECT SUM("S"."QUANTITY_SOLD") AS "Total_Sales_Quantity" FROM "SH"."SALES"
• Na	SELECT SUM("S"."QUANTITY_SOLD") AS "Total_Sales_Quantity" FROM "SH"."SALES" "S" JOIN "SH"."TIMES" "T" ON "S"."TIME_ID" = "T"."TIME_ID" WHERE "T"."CALENDAR_YEAR" = 2000

```
select ai narrate 2000년도의 총 판매량은 얼마인가요 (한글로 출력해주세요)
```



	RESPONSE
0	2000년도의 총 판매량은 232,646입니다.

Oracle Select AI

데모 순서

- ☐ 1-1. DB 접속 & Select AI 설정
- ☐ 1-2. Select AI 기초
- ☒ 1-3. SH 스키마 쿼리 데모
- ☐ 1-4. Movie 테이블 추가
- ☐ 1-5. Movie 프로필 등록
- ☐ 1-6. Movie 데이터 조회
- ☐ 2-1. Select AI RAG
- ☐ 3-1. NoCode RAG: OCI Gen AI Agent
- ☐ 4-1. Naive RAG With ADB & Cohere

SH Query 목록

Nason Mann 이 구매한 상품들의 총액은 얼마인가요

SH 복합 쿼리 실행

- RUNSQL: Nason Mann 이 구매한 상품들의 총액은 얼마인가요

```
select ai runsql Nason Mann 이 구매한 상품들의 총액은 얼마인가요
```

	Total_Amount_Sold
0	67,773

- SHOWSQL: Nason Mann 이 구매한 상품들의 총액은 얼마인가요

```
select ai showsql Nason Mann 이 구매한 상품들의 총액은 얼마인가요
```

	RESPONSE
0	SELECT SUM("S"."AMOUNT_SOLD") AS "Total_Amount_Sold" FROM "SH"."SALES" "S" JOIN "SH"."CUSTOMERS" "C" ON "S"."CUST_ID" = "C"."CUST_ID" WHERE "C"."CUST_FIRST_NAME" = 'Nason' AND "C"."CUST_LAST_NAME" = 'Mann'
• Nason Mann 이 구매한 상품들의 총액은 얼마인가요	

```
select ai narrate Nason Mann 이 구매한 상품들의 총액은 얼마인가요 (한글로 출력해주세요)
```

	RESPONSE
0	Nason Mann이 구매한 상품들의 총액은 67,773입니다.

Oracle Select AI

데모 순서

- ☐ 1-1. DB 접속 & Select AI 설정
- ☐ 1-2. Select AI 기초
- ☐ 1-3. SH 스키마 쿼리 데모
- ☐ 1-4. Movie 테이블 추가
- ☒ 1-5. Movie 프로파일 등록
- ☐ 1-6. Movie 데이터 조회
- ☐ 2-1. Select AI RAG
- ☐ 3-1. NoCode RAG: OCI Gen AI Agent
- ☐ 4-1. Naive RAG With ADB & Cohere

- 설정 코드: Select AI 대상 프로파일

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OPENAI',
    attributes => '{
      "provider": "openai",
      "credential_name": "OPENAI_CRED",
      "object_list": [
        {"owner": "SH", "name": "CHANNELS"},
        {"owner": "SH", "name": "COSTS"},
        {"owner": "SH", "name": "COUNTRIES"},
        {"owner": "SH", "name": "CUSTOMERS"},
        {"owner": "SH", "name": "PRODUCTS"},
        {"owner": "SH", "name": "PROMOTIONS"},
        {"owner": "SH", "name": "SALES"},
        {"owner": "SH", "name": "TIMES"},
        {"owner": "ADMIN", "name": "ACTOR"},
        {"owner": "ADMIN", "name": "MOVIE"},
        {"owner": "ADMIN", "name": "DIRECTOR"},
        {"owner": "ADMIN", "name": "movie_actor"}
      ],
      "max_tokens": 512,
      "stop_tokens": [";"],
      "model": "gpt-4o",
      "temperature": 0.1,
      "comments": true
    }'
  );
END;
```

Oracle Select AI

데모 순서

- 1-1. DB 접속 & Select AI 설정
- 1-2. Select AI 기초
- 1-3. SH 스키마 쿼리 데모
- 1-4. Movie 테이블 추가
- 1-5. Movie 프로필 등록
- 1-6. Movie 데이터 조회
- 2-1. Select AI RAG
- 3-1. NoCode RAG: OCI Gen AI Agent
- 4-1. Naive RAG With ADB & Cohere

자연어 쿼리

2020년 최다 관객 동원 배우 3명을 알려주세요

Movie 쿼리 실행

- RUNSQL: 2020년 최다 관객 동원 배우 3명을 알려주세요

`select ai runsql 2020년 최다 관객 동원 배우 3명을 알려주세요`

	Actor_Name	Total_Audience
0	이종석	10,000,000
1	박보검	3,800,000
2	김혜수	900,000

- SHOWSQL: 2020년 최다 관객 동원 배우 3명을 알려주세요

`select ai showsql 2020년 최다 관객 동원 배우 3명을 알려주세요`

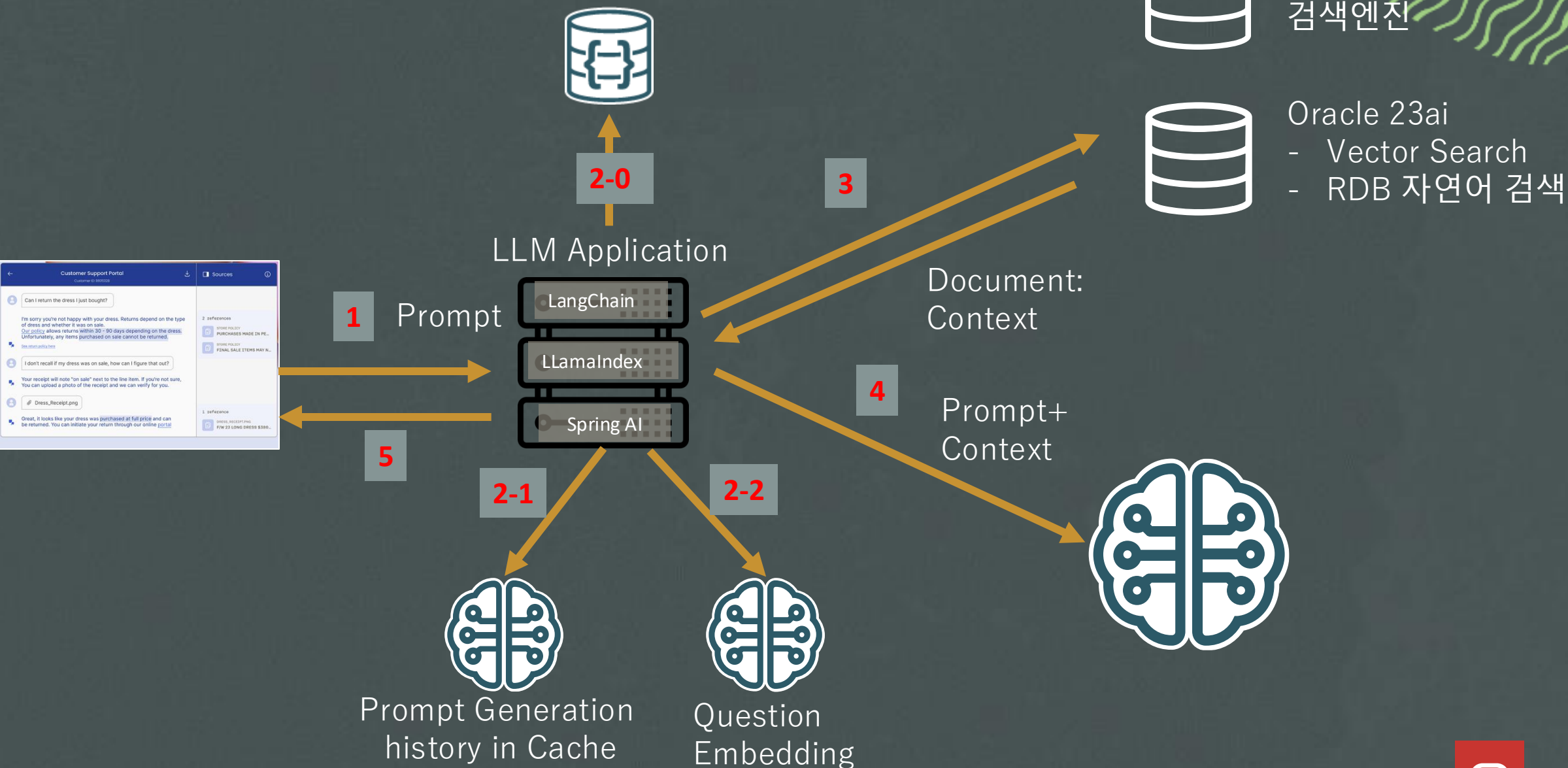


	RESPONSE
0	SELECT "A"."NAME" AS "Actor_Name", SUM("M"."AUDIENCE_COUNT") AS "Total_Audience" FROM "ADMIN"."MOVIE_ACTOR" "MA" JOIN "ADMIN"."ACTOR" "A" ON "MA"."ACTOR_ID" = "A"."ACTOR_ID" JOIN "ADMIN"."MOVIE" "M" ON "MA"."MOVIE_ID" = "M"."MOVIE_ID" WHERE EXTRACT(YEAR FROM "M"."RELEASE_DATE") = 2020 GROUP BY "A"."NAME" ORDER BY "Total_Audience" DESC FETCH FIRST 3 ROWS ONLY
• Na	

`sele`

세요)

RAG(Retrieval Augmented Generation) 아키텍처:



Prompt
show the workorder with the most total scrap quantity.

🔍 Ask

Workorder Name ↑≡	Workorder Id	Scrap Quantity	Assembly Name	Assembly Description	Assembly Category	Assembly Uom
G95-WO1	1011352	7005	G95-Mirror Assembly	G95-Mirror Assembly		Ea



Prompt

Are there any quality issues reported for the assembly of this workorder

🔍 Ask

Collection Plan ⬆≡	Issue Description	Severity	Cost	Status	Date Opened	Date Closed	Nonconformance Number	Workorder	Assembly	Total Scrap Quantity
NTP1_DISP	In process nonconformance	MEDIUM		PENDING	2/14/2022		NC189	G95-WO1	G95-Mirror Assembly	7005
NTP1_DISP	Out of spec assembly	HIGH	1700	CLOSED	2/8/2022	2/8/2022	NC188	G95-WO1	G95-Mirror Assembly	7005
NTP1_DISP	In process damaged assembly	HIGH	1000	CLOSED	2/8/2022	2/8/2022	NC187	G95-WO1	G95-Mirror Assembly	7005
NTP1_NCM_MAST	Assembly issue	HIGH	4000	NEW	2/14/2022		NC202	G95-WO1	G95-Mirror Assembly	7005
NTP1_NCM_MAST	Out of spec assembly	HIGH	1700	CLOSED	2/8/2022	2/8/2022	NC188	G95-WO1	G95-Mirror Assembly	7005
NTP1_NCM_MAST	In process nonconformance	HIGH	2000	NEW	4/17/2023		NC195	G95-WO1	G95-Mirror Assembly	7005



Prompt

are any of these issues due to a faulty component

🔍 Ask

Collection Plan ⬆️	Issue Description	Severity	Cost	Status	Date Opened	Date Closed	Nonconformance Number	Workorder	Assembly	Total Scrap Quantity	Faulty Component
NTP1_DISP	Out of spec assembly	HIGH	1700	CLOSED	2/8/2022	2/8/2022	NC188	G95-WO1	G95-Mirror Assembly	7005	Plate Glass
NTP1_DISP	In process nonconformance	MEDIUM		PENDING	2/14/2022		NC189	G95-WO1	G95-Mirror Assembly	7005	Plate Glass
NTP1_DISP	In process damaged assembly	HIGH	1000	CLOSED	2/8/2022	2/8/2022	NC187	G95-WO1	G95-Mirror Assembly	7005	Plate Glass
NTP1_NCM_MAST	Assembly issue	HIGH	4000	NEW	2/14/2022		NC202	G95-WO1	G95-Mirror Assembly	7005	Plate Glass
NTP1_NCM_MAST	In process nonconformance	HIGH	2000	NEW	4/17/2023		NC195	G95-WO1	G95-Mirror Assembly	7005	Plate Glass



하이브리드 검색 데모

벡터 검색과 키워드 검색

토끼가 잠을 잔
이유

거북이가 승리한
이유

하이브리드 검색

```
SELECT JSON_SERIALIZE(  
  DBMS_HYBRID_VECTOR.SEARCH(  
    json('{  
      "hybrid_index_name": "my_hybrid_idx",  
      "search_text": "토끼가 잠을 잔 이유",  
      "search_fusion": "INTERSECT",  
      "return": {  
        "topN": 5  
      }  
    }')) ) txt  
FROM dual
```

키워드 +
벡터 검색

벡터 검색

키워드검
색

질문과 관련된 검색결과

- ✓ 한참 앞서간 토끼는 거북이가 자신을 따라잡지 못할 것이라 생각하고 길가에 누워 잠을 잤습니다
 - 토끼는 웃으며 제안을 받아들였고, 경주가 시작되었습니다. (관계가 적은 데이터)
 - 토끼는 자신의 빠른 다리를 자랑하며 친구들에게 늘 과시하곤 했습니다. (관계가 적은 데이터)

하이브리드 검색(Hybrid Vector Index)

임베딩 모델 : MULTILINGUAL_E5_SMALL

기능 소개 및 설명

데모 준비 하이브리드 검색

검색할 텍스트를 입력하세요

검색옵션

Top-K

하이브리드...

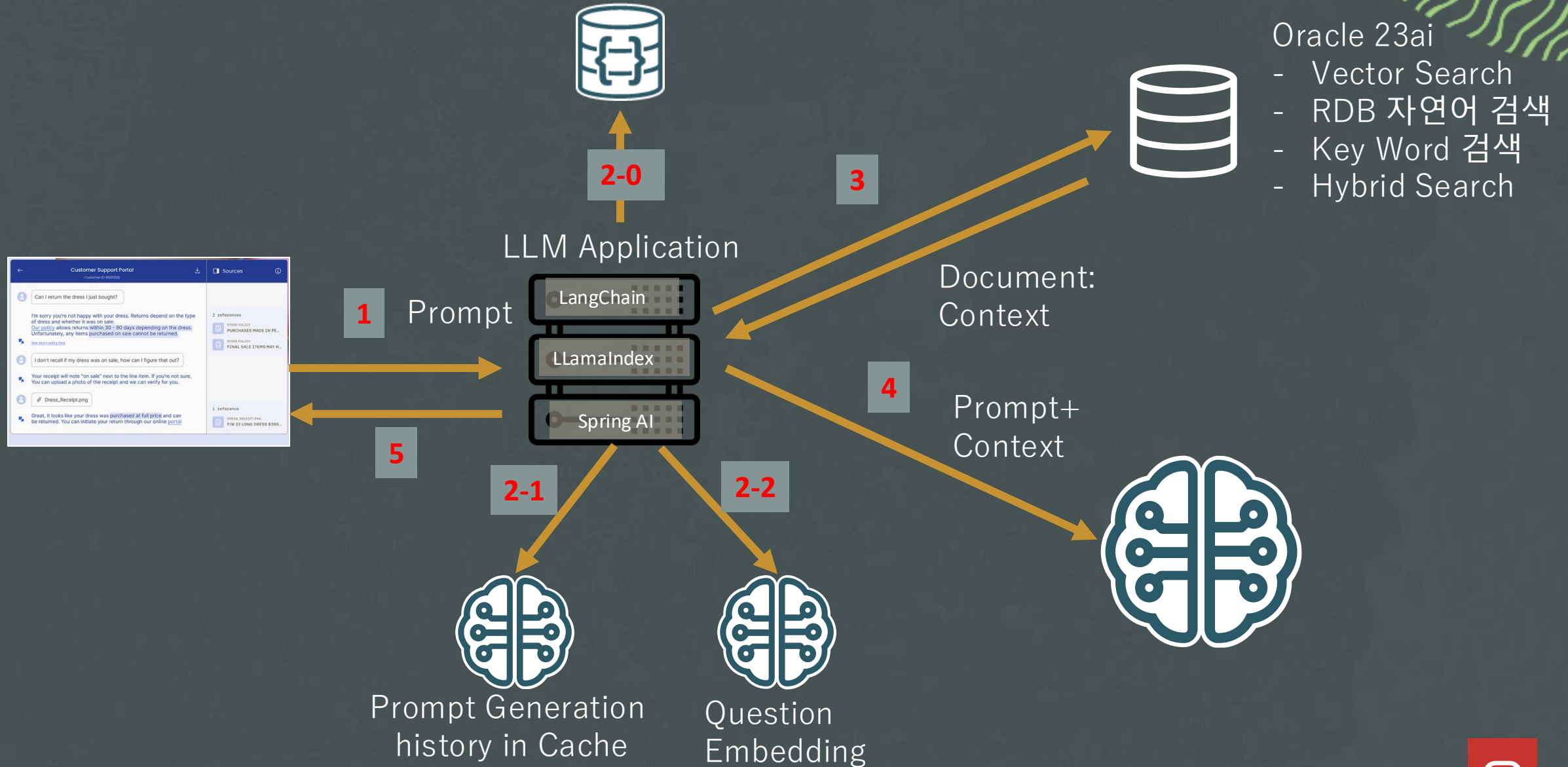
1 5 10

토끼가 잠을 잔 이유

거북이가 승리한 이유

```
SELECT JSON_SERIALIZE(  
  DBMS_HYBRID_VECTOR.SEARCH(  
    json('{  
      "hybrid_index_name": "my_hybrid_idx",  
      "search_text": "",  
      "search_fusion": "INTERSECT",  
      "return": {  
        "topN": 5  
      }  
    }')) ) txt  
FROM dual
```

RAG(Retrieval Augmented Generation) 아키텍처:



AI Solutions Hub 바로 시작해 보세요

빠른 시작 AI 솔루션

자사 OCI 태넌시 솔루션 구성 가능

각 AI 솔루션 구성:

- 예제 코드
- 빠른 시작 가이드
- 튜토리얼 영상

새로운 솔루션은 계속 추가 됩니다

OCI About Services Solutions Pricing Partners Resources

Cloud > Artificial Intelligence >

AI Solutions Hub

Enter a new era of productivity with generative AI solutions for your business. Leverage AI, embedded as you need it, across the full stack.

[Learn more about Oracle AI](#) [Speak to an AI expert](#)

AI solutions

Enhance customer engagement by automating content creation
Use a simple, web-based UI to harness the power of generative AI.

- [Sample code](#)
- [Quick start guide](#)
- [Tutorial \(11:26\)](#)

Improve efficiency and save time by summarizing data from any source
Quickly extract content from web sources and summarize it using Oracle Cloud Infrastructure (OCI) Generative AI.

- [Sample code](#)
- [Quick start guide](#)
- [Tutorial \(11:43\)](#)

Streamline quality control in manufacturing with Object Detection
Develop an AI-infused application with the help of OCI Vision and get detection results.

- [Sample code](#)
- [Quick start guide](#)
- [Tutorial \(16:21\)](#)

[Talk to sales](#)

생성형 AI 시대에 오라클의 3개 전략

전략1: Oracle Enterprise LLM/Generative AI Platform



Model



DATA



INFRASTRUCTURE



Oracle
Database 23C



OpenSearch



Heatwave



ADW



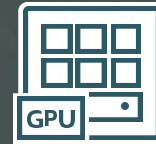
ATP



EXACI



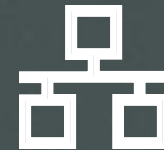
EXACC



H100/A100
/A10 GPU



Cloud Native
OKE



Super Cluster



Chatbot



생성형 AI 시대에 오라클의 3개 전략

전략1: Oracle Enterprise LLM/Generative AI Platform



Model



DATA



INFRASTRUCTURE



Oracle
Database 23C



OpenSearch



Heatwave



ADW



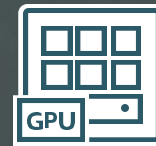
ATP



EXACI



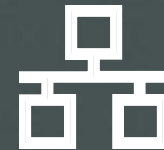
EXACC



H100/A100
/A10 GPU



Cloud Native
OKE



Super Cluster



Chatbot

전략 2: 검증된 LLM



생성형 AI 시대에 오라클의 3개 전략

전략1: Oracle Enterprise LLM/Generative AI Platform



Model



DATA



INFRASTRUCTURE



Oracle
Database 23C



OpenSearch



Heatwave



ADW



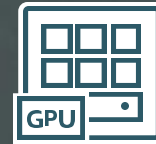
ATP



EXACI



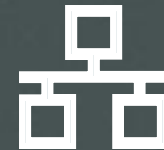
EXACC



H100/A100
/A10 GPU



Cloud Native
OKE



Super Cluster



Chatbot

전략 2: 검증된 LLM

전략 3: Gen AI를 위한 데이터 관리 체계:

Oracle Database 23ai



생성형 AI 시대에 오라클의 3개 전략

전략1: Oracle Enterprise LLM/Generative AI Platform



Model



DATA



INFRASTRUCTURE



Oracle
Database 23C



OpenSearch



Heatwave



ADW



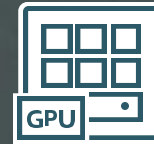
ATP



EXACI



EXACC



H100/A100
/A10 GPU



Cloud Native
OKE



Super Cluster



Chatbot

전략 2: 검증된 LLM

전략 3: Gen AI를 위한 데이터 관리 체계:

Oracle Database 23ai





감사합니다

김태완 상무

taewan.kim@oracle.com

Cloud Engineer Team

Oracle Korea

