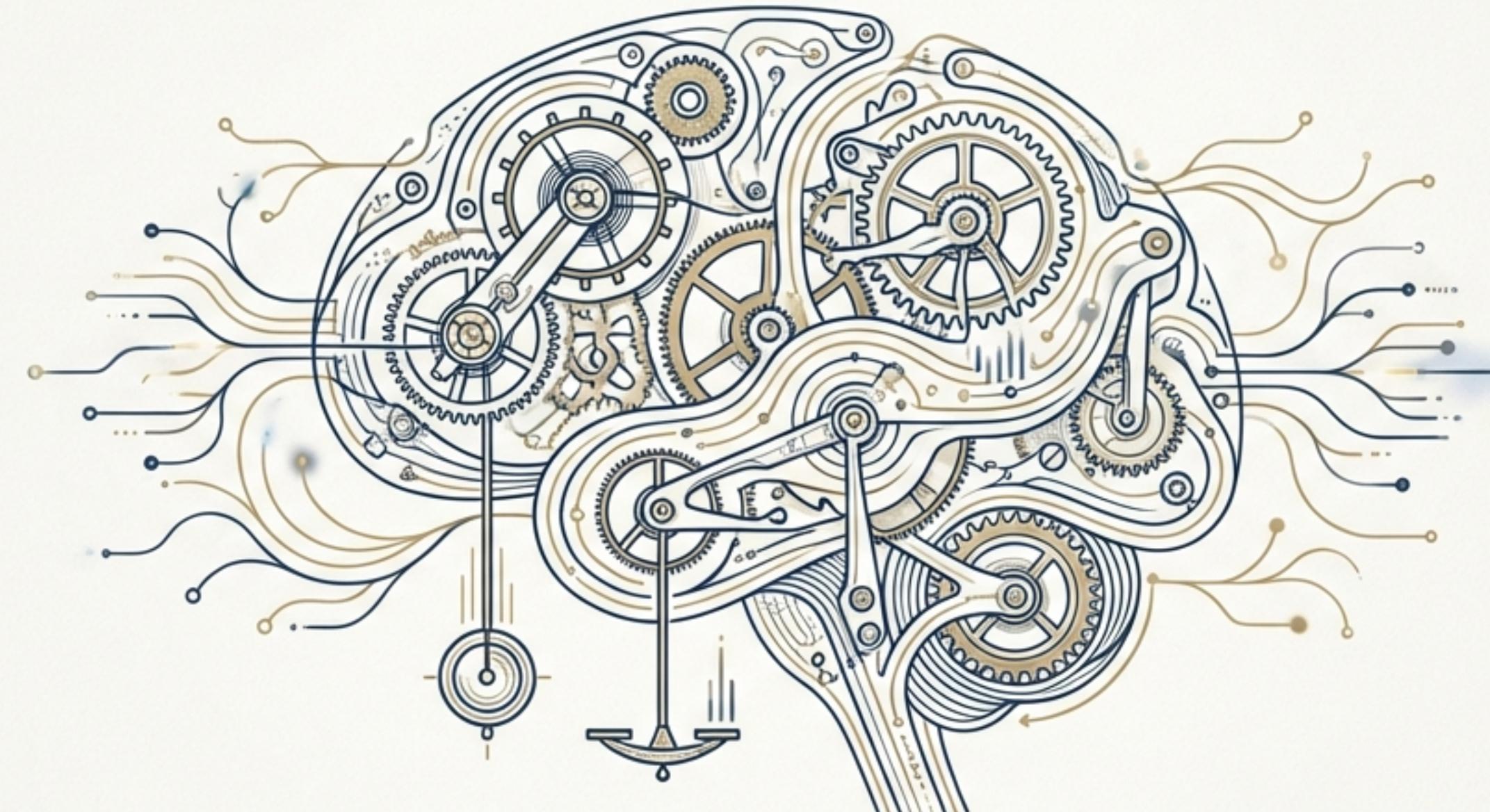


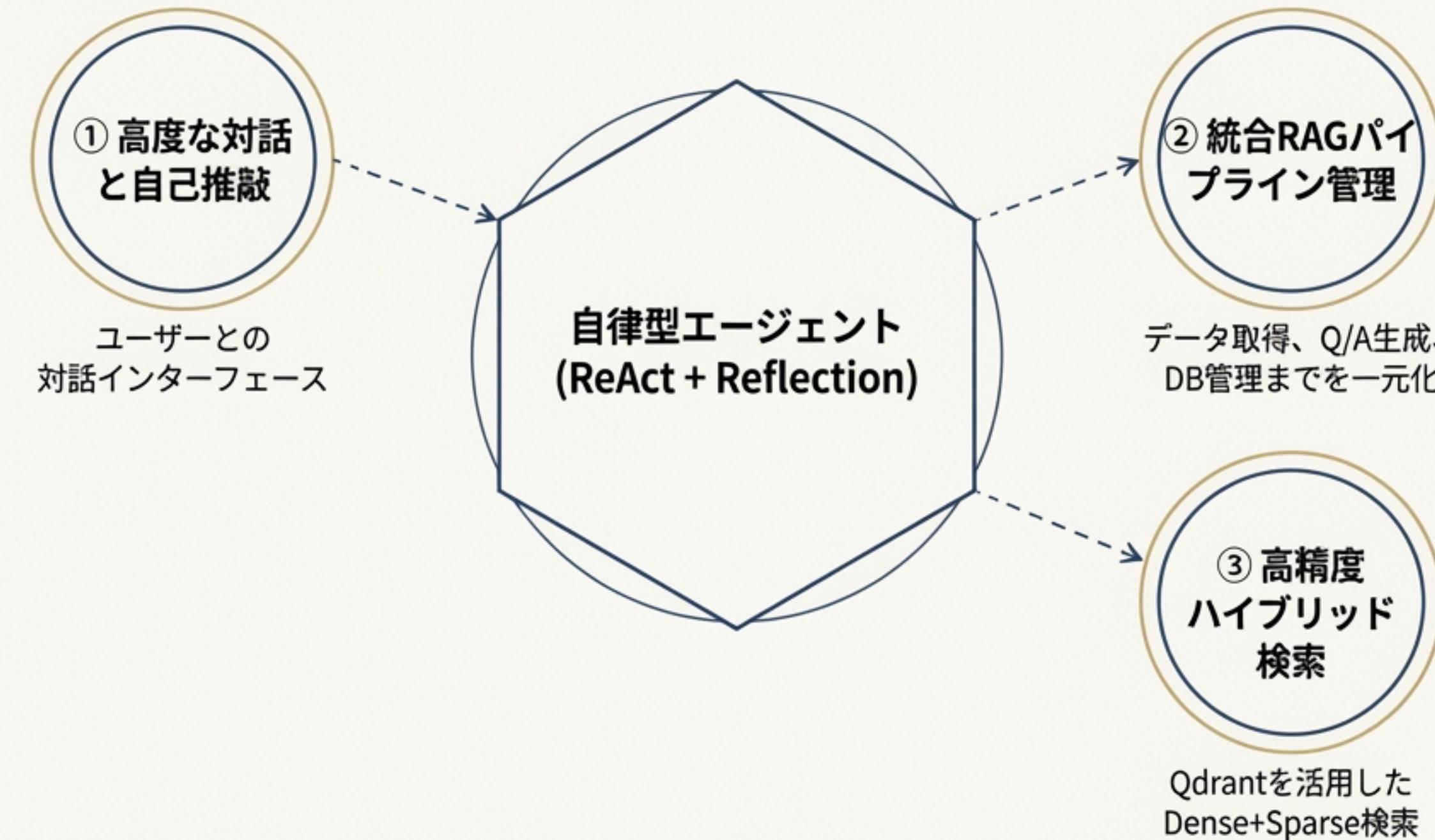
自律型RAGエージェントシステムの全貌

ReActとReflectionが拓く、次世代ナレッジ対話基盤



Gemini 3 (2.0 Flash)世代対応 統合プラットフォーム

これは単なるRAGではない。
自律エージェントを心臓部に持つ
統合プラットフォームである。



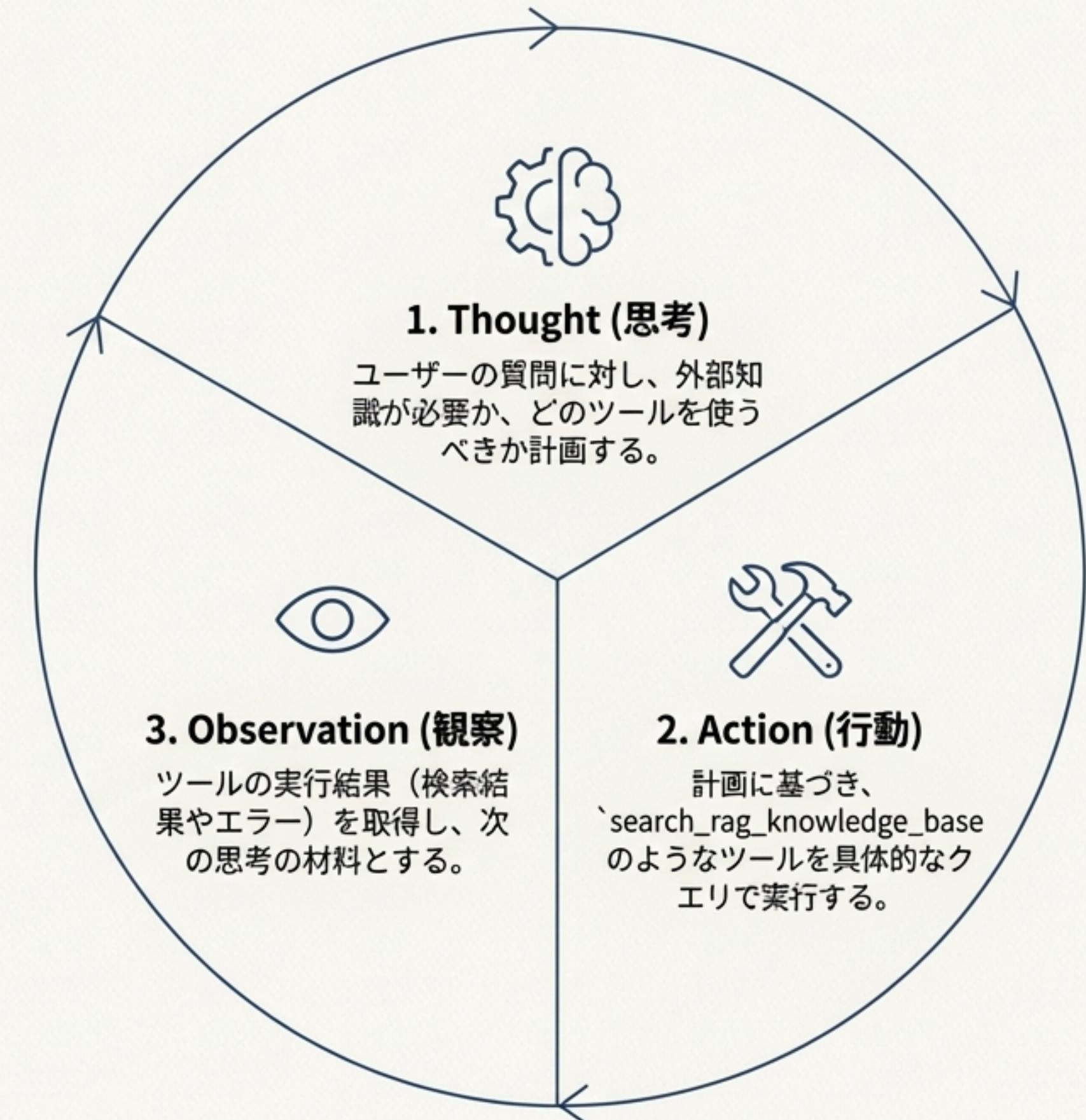
本システムは、Streamlit UIを通じてデータの取得・ベクトル化からQdrantデータベース管理、そして高度なエージェント対話まで、RAGパイプライン全体を一気通貫で管理・運用することができます。

エージェントの思考エンジン: ReAct (Reasoning + Acting)

コンセプト: エージェントは「考え」、ツールを「使い」、結果を「観察」するサイクルを繰り返すことで、必要な情報が揃うまで自律的に問題解決にあたります。

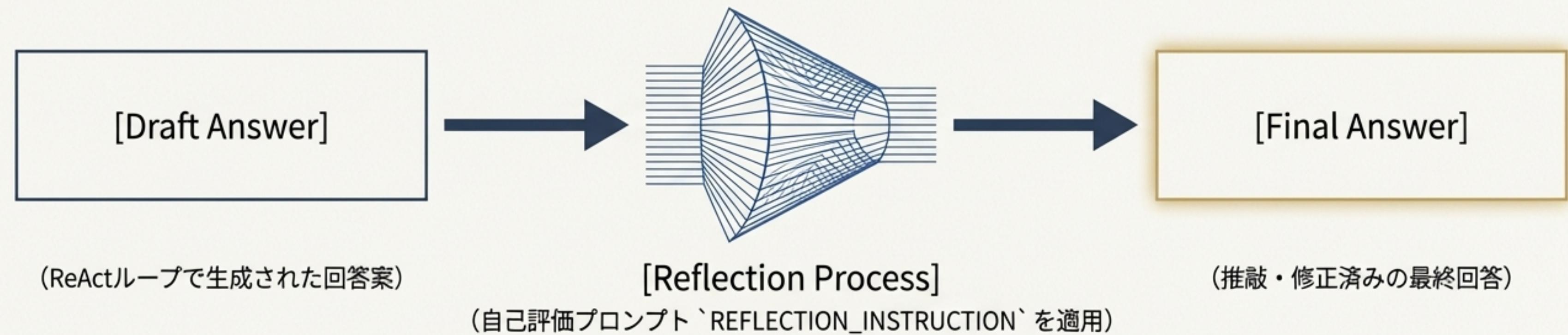
キーポイント

このループにより、エージェントは単一の検索に失敗しても、クエリを言い換えたり別の知識ソースを試したりするリカバリー戦略を自律的に実行できます。



回答品質を向上させる自己推敲エンジン: Reflection

コンセプト: 回答を即座に出力せず、一度立ち止まって「自己評価」するフェーズを設けることで、ハルシネーションを抑制し、より洗練された回答を生成します。



評価基準（プロンプトによる指示）：

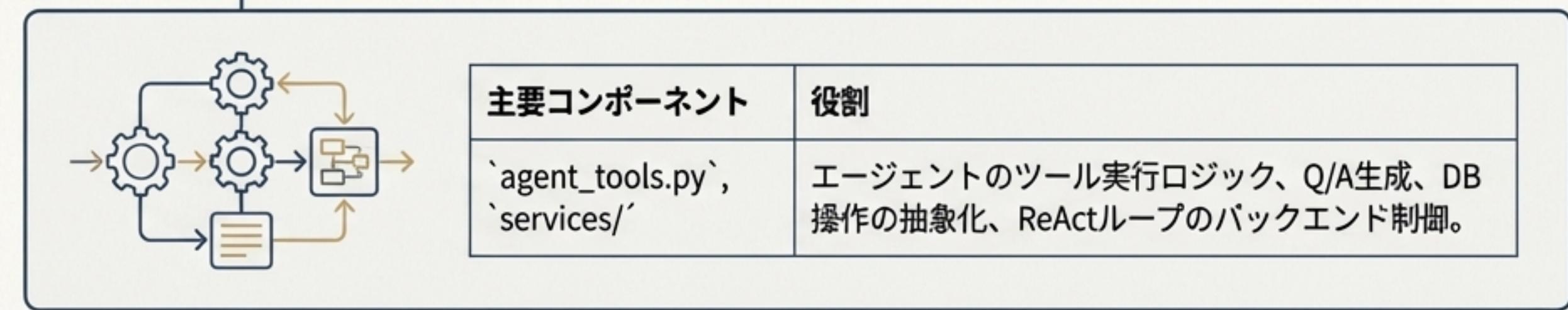
- ☑ 正確性 (Accuracy) : 検索結果に基づいているか？幻覚はないか？
- ⌚ 適切性 (Relevance) : ユーザーの質問に直接答えているか？
- ✍️ スタイル (Style) : 親しみやすく丁寧な日本語（です・ます調）か？

思考を支える設計思想: 責務を分離した3層アーキテクチャ

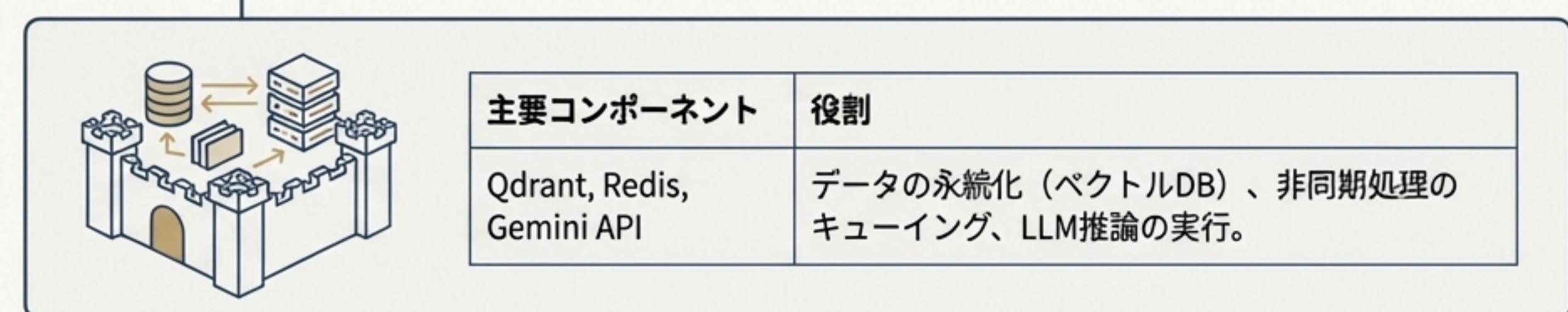
1. UI層



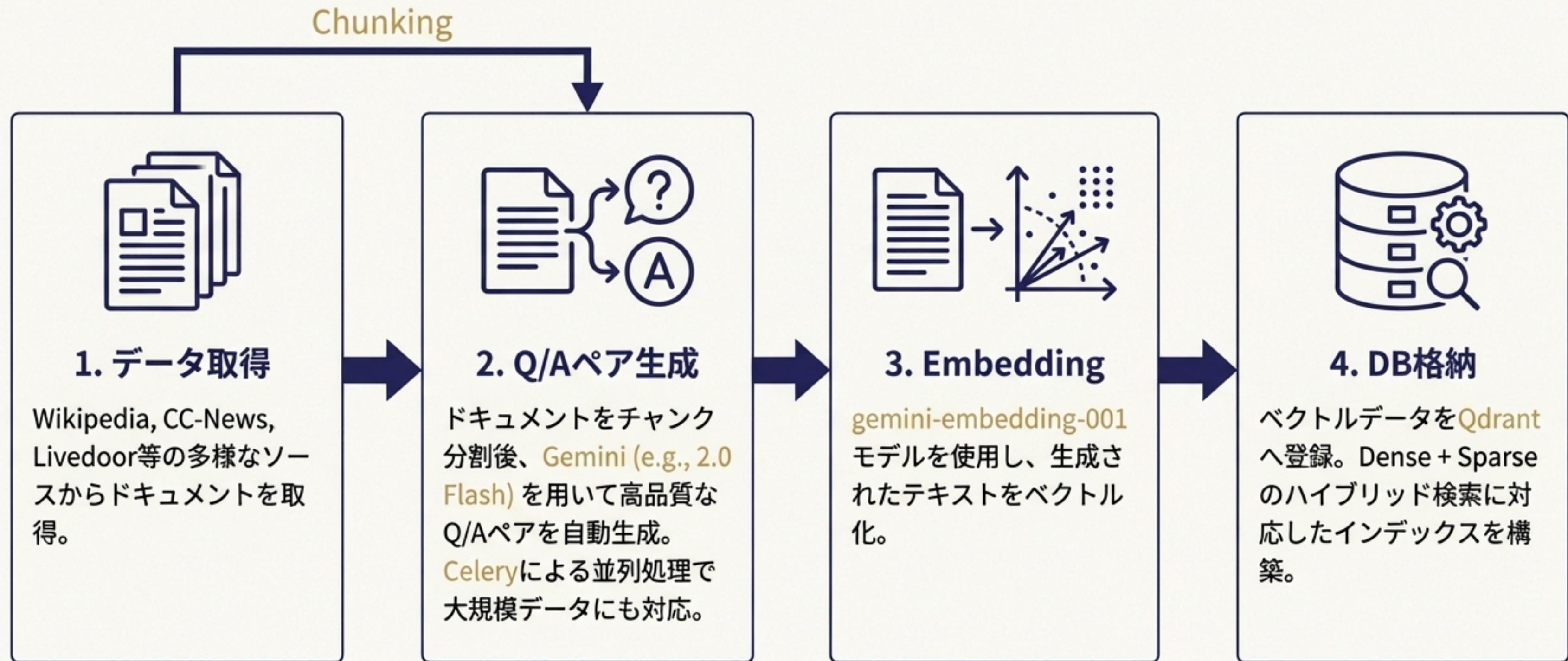
2. サービス・ツール層



3. インフラ層



知識の源泉: エンドツーエンドRAGデータパイプライン



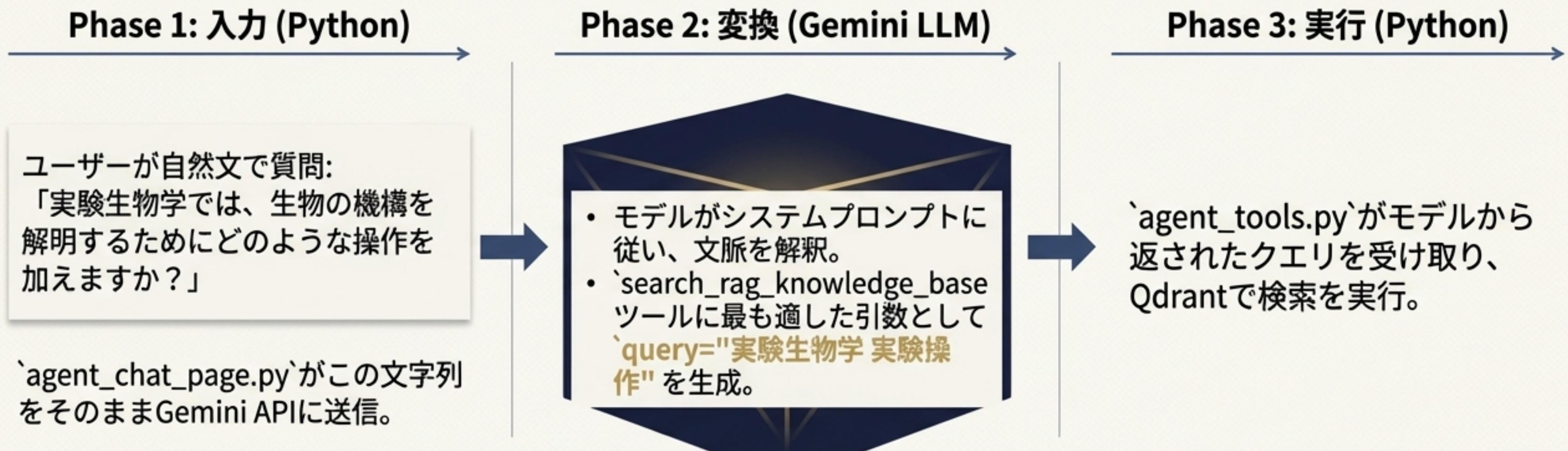
すべてを管理する司令塔: 統合管理UI

コンセプト: Streamlitで構築されたUIから、RAGパイプラインの全工程を直感的に操作・管理可能。

アイコン	画面名	機能概要
	エージェント対話	ReAct + Reflectionエージェントとの対話。思考プロセスをリアルタイムで可視化。
	RAGデータDL	HuggingFace等から学習用データをダウンロード・前処理。
	Q/A生成	GeminiとCeleryを活用し、ドキュメントからQ/Aペアをバッチ生成。
	CSVデータ登録	既存のQ/Aデータをベクトル化し、Qdrantコレクションへ登録。
	Qdrantデータ管理	登録済みコレクションの統計情報や内容を閲覧。
	Qdrant検索	セマンティック検索単体の性能をテスト。
	未回答ログ	エージェントが回答できなかった質問を記録・分析し、知識ベースの改善に活用。

技術的ハイライト①: クエリ生成はPythonコードではなく、LLMの「推論」である

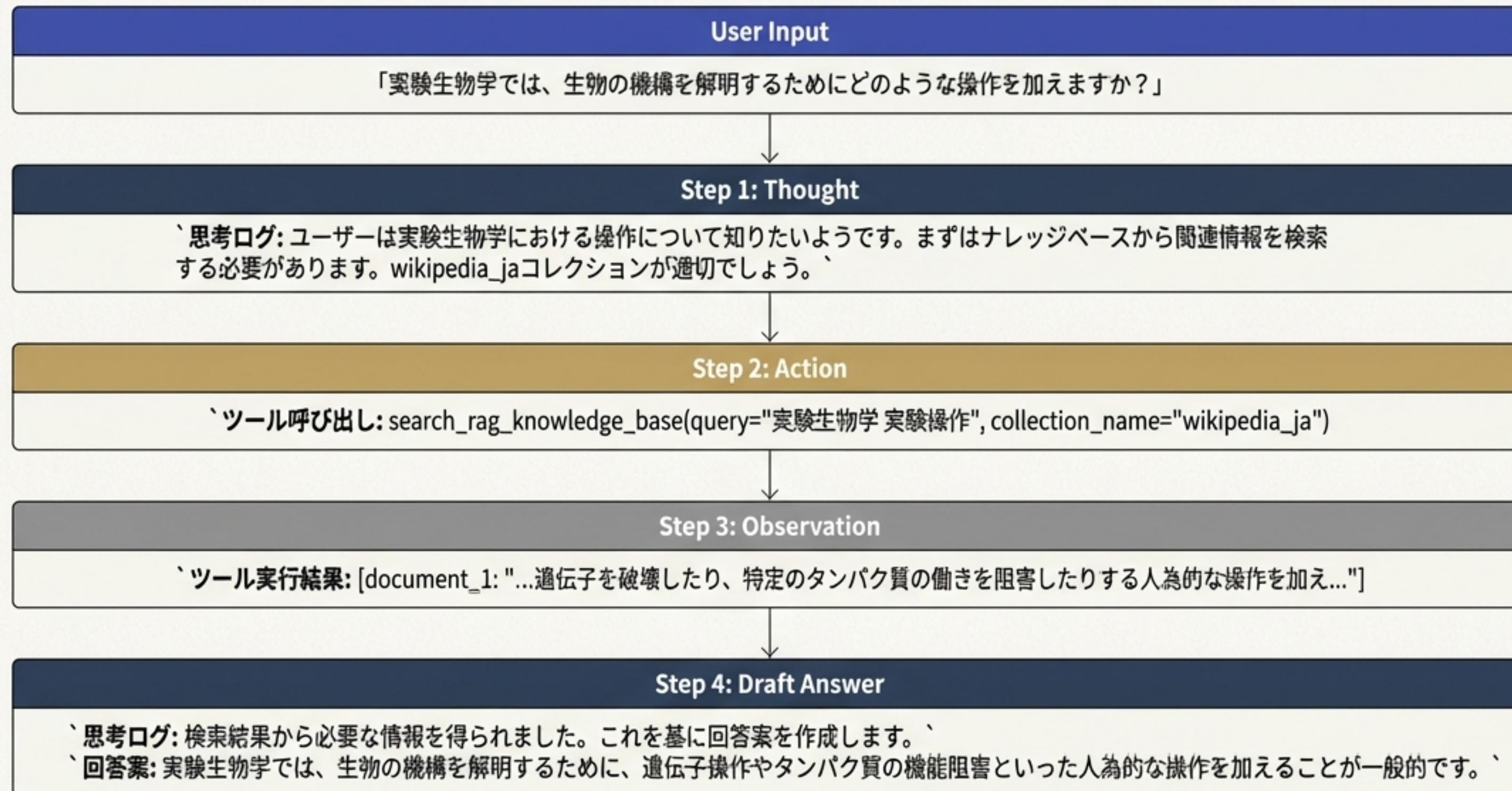
コンセプト: Python側に専用の「クエリ整形関数」は存在しません。Geminiモデル自身が、プロンプトの指示に基づき、ユーザーの自然な入力文を解釈し、ツールに渡すべき最適な検索クエリを「推論」しています。



結論**: 「クエリ生成ロジック」の実体は、LLMの頭脳（推論プロセス）の中にあります。

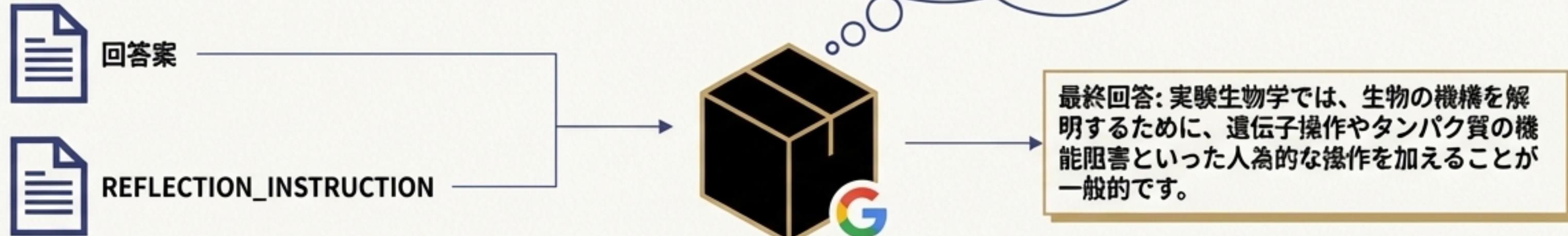
技術的ハイライト②: エージェントの一巡 (Turn) を解剖する

- ReActの思考連鎖



技術的ハイライト③: 最終的な磨き上げ - Reflectionによる自己推敲

推敲プロセス



1. 推敲指示 (Python → Gemini)

前のステップで生成された「回答案」と、評価基準を定義した「REFLECTION_INSTRUCTION」プロンプトを結合して、再度Gemini APIに送信。

2. 自己評価 (Gemini内部)

Reflection Thought: 回答は検索結果に基づいており、正確性に問題はない。ユーザーの質問にも直捷答えている。スタイルも丁寧な「です・ます調」であるため、修正は不要と判断しました。

3. 最終回答の生成 (Gemini → Python)

キーポイント

この追加ステップにより、事実誤認だけでなく、不自然な言い回しや不適切なフォーマットも自律的に修正され、回答全体の品質が保証されます。

システムを支えるコア技術スタック

カテゴリ	技術要素	備考
LLM (Agent / Q/A生成)	Gemini 2.0 Flash, Gemini 1.5 Pro	高速な推論とFunction Calling (ReAct) に対応
Embedding	gemini-embedding-001	デフォルトの埋め込みモデル
Vector DB	Qdrant	Dense + Sparse ハイブリッド検索をサポート
Web UI	Streamlit	インタラクティブなチャットUIと管理画面を迅速に構築
並列処理	Celery + Redis	大規模なQ/A生成タスクをバックグラウンドで高速処理
言語	Python 3.10+	-
形態素解析	MeCab	日本語テキストの高精度な文分割に利用

3ステップで始める自律型エージェント

1

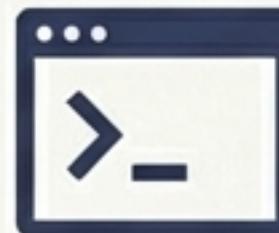


必須環境変数の設定

プロジェクトルートに`.env`ファイルを作成し、APIキーを設定します。

```
# .env file  
Google_API_KEY=your_google_api_  
key_here
```

2



起動コマンド

以下のコマンドでStreamlitアプリケーションを起動します。

```
streamlit run agent_rag.py  
--server.port=8500
```

3



ブラウザで確認

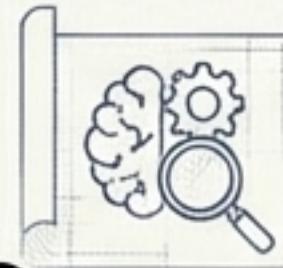
ブラウザで`http://localhost:8500`にアクセスし、エージェント対話画面を確認します。



このプロジェクトが実現したこと: 3つの核心的価値

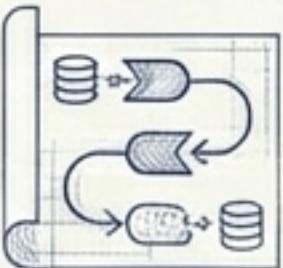
1 自律性と品質の の両立

ReAct + Reflectionエージェントにより、ツールを自律的に使いこなし、かつ自己推敲によってハルシネーションを抑制した高品質な回答を生成。



2 パイプライン全 全体の統合

データ取得から知識ベース構築、対話インターフェースまで、RAGのライフサイクル全体を单一のプラットフォームで管理・運用可能にした。



3 堅牢かつ拡張可能な アーキテクチャ

責務が明確な3層アーキテクチヤを採用し、保守性と拡張性を確保。実用的なアプリケーションとしての技術的基盤を構築した。

