ORACLE

Siebel Genie

Natural-Language Access to Siebel CRM Data using Oracle Select AI & APEX

July 2025, Version [1.0]

Purpose statement

This document provides a solution approach and is for informational purposes only. It is intended solely to provide valuable insights and guidance on the subject matter while serving as a reference for stakeholders. This document does not constitute a binding agreement or official policy.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement, nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the solution described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

ORACLE

List of figures

List of tables

List of figures and tables

Summary

For decades, Oracle's Siebel CRM has excelled as a highly flexible, mission-critical Enterprise CRM solution, providing a trusted foundation for customer experience and to manage heavily integrated operational data.

Our customers have long relied on the key strengths of our robust data model, our deep industry and specialized process capabilities across the full CRM spectrum, our best-in-class security controls, and flexible extensibility options.

The combination of Siebel CRM on AI makes for an exciting transformation from "system of record" to "system of intelligence" by activating your CRM data. Your data is a strategic asset built over decades, and it is crucial to leverage it in your organizational AI initiatives.

This white paper solution introduces the concept of "Siebel Genie", a solution that extends access to that data using natural language, while preserving the governance and security posture of Siebel CRM.

Siebel Genie is built with Oracle Select AI, Autonomous Database 23ai, and Oracle APEX. The solution converts natural-language questions into governed SQL statements, executed on the existing Siebel CRM schema. The approach allows organizations to:

- **Democratize insight** – business users ask questions directly; no SQL skills needed.
- **Preserve governance** – leverages Siebel view-mode security; execution is read-only and auditable.
- **Avoid core changes** – implementation of Siebel Genie should not impact existing Siebel deployments or configurations.

The modular nature of Siebel Genie allows organizations to adopt similar solutions incrementally. Anything is possible, starting with a simple use-case for super users or specific personas, to scaling to additional business domains as needed. Siebel Genie is a simple solution to inspire the broader adoption of AI-driven use cases across business processes.

New Opportunities for Data Insights

Siebel CRM houses a wealth of high-quality operational data – opportunities, service requests, contact interactions, and more – governed by mature view-mode security. Yet, potential "insights" remain locked within data – behind screens, reports, or SQL expertise. The rising availability of generative-AI tooling turns that latent asset into an immediate opportunity: empower every authorised employee to explore Siebel data using natural language, shorten decision cycles, and amplify the return on an existing CRM investment.

Table 1. Potential opportunities for optimization & unlock paths with Siebel Genie.

Opportunity	Today's Reality	Unlock Path with Siebel Genie
Speed to Insight	Users wait days for new or modified BI reports.	On-demand answers in seconds via natural-language prompts.
Self-Service Adoption	Only analysts or power users can craft SQL or complex queries.	Business staff run governed NLQ queries with zero SQL knowledge.
360° Context	Dashboards cover slices of data; cross-object questions require custom joins.	AI agents assemble joins on the fly – e.g., *"Show open opportunities with unresolved SRs"*.

Opportunity	Today's Reality	Unlock Path with Siebel Genie
IT Efficiency	Report backlog consumes developer hours.	Target ~ 50% reduction in ad-hoc report tickets (target KPI; see pg.9 – Recommended Success Metrics).
Governed Innovation	Fear of bypassing view-mode security slows experimentation.	Select AI profiles embed Siebel view-mode logic; execution stays read-only and auditable.

Siebel's rich metadata and view-mode rules already describe both the data relationships and the access constraints. **Siebel Genie uses these assets "as-is".** It converts a plain-language question into a governed, read-only SQL statement and returns the results in the same session – without touching Siebel's schema or security model. The same orchestration can be extended to support predictive scoring, proactive service scheduling, guided chats as needed. However, the core problem that Siebel Genie solves is to enable natural language to governed SQL queries.

Internal trials show that many routine questions are resolved on the spot rather than becoming new report tickets, but the precise impact on turnaround time and backlog will differ by organisation and could be tracked against the KPIs recommended in pg. 9 (See **Recommended Success Metrics**).

Solution Overview

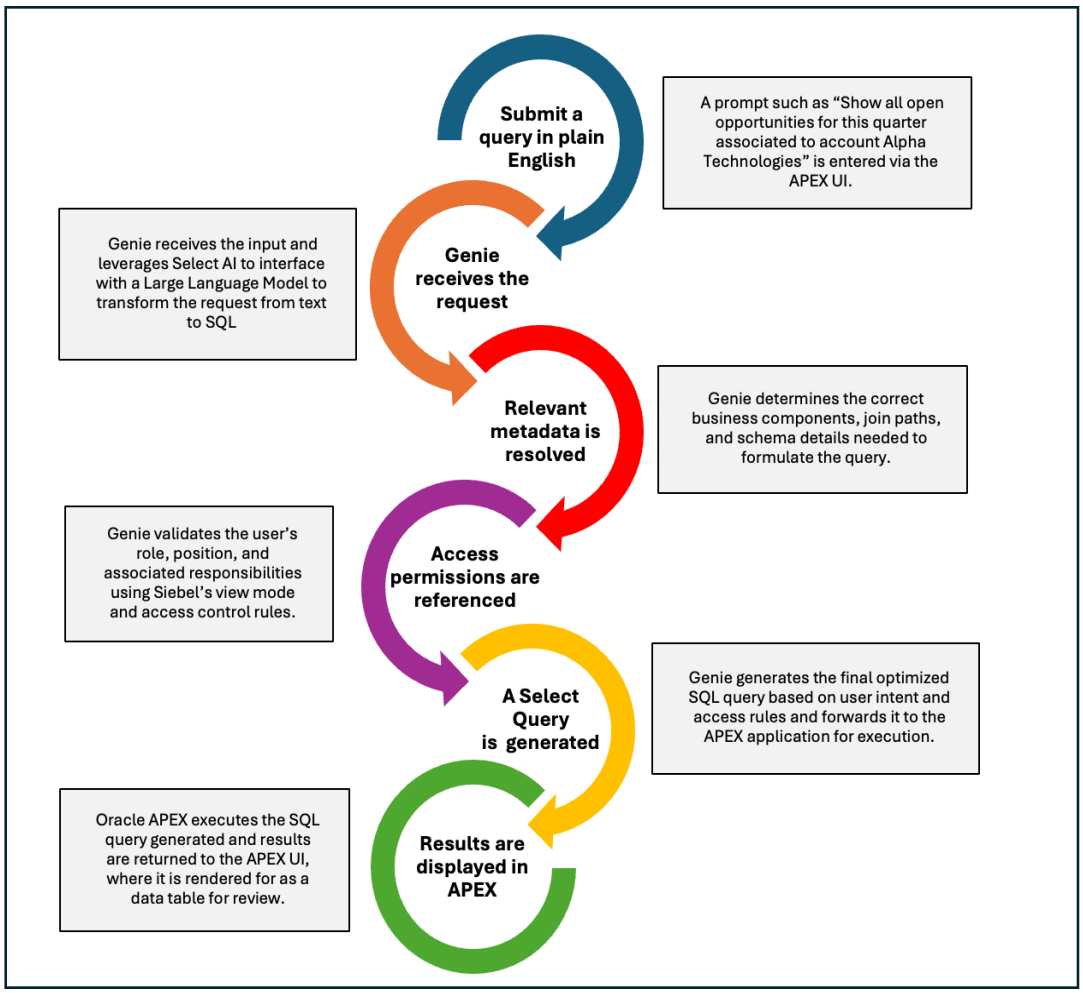
Siebel Genie introduces a natural language interface for accessing Siebel CRM data – allowing users to ask questions in natural language and receive structured, secure answers. It removes the need to navigate complex UIs or understand SQL, while preserving Siebel's access rules, metadata integrity, and governance model.

Siebel Genie can be a fantastic solution for:

- **Business users** who need on-demand access to insights and depend on accurate data for decision-making.
- **IT teams** looking to modernize user interaction or improve adoption for super users and senior leaders without disrupting the core Siebel CRM application.

The figure below outlines the key stages of how a user's natural language query is processed by Siebel Genie – from submission to secure, governed data retrieval:

Figure 1. Functional flow of what processes occur when a natural language prompt is processed by Siebel Genie



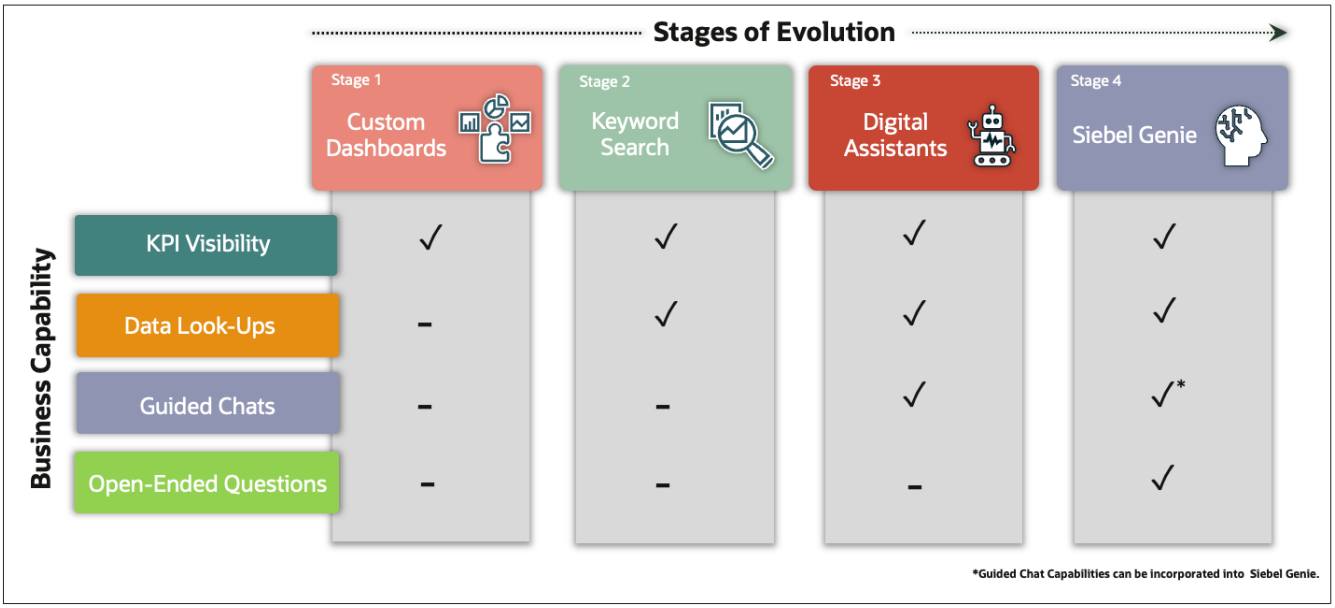
The end-to-end process of converting plain text to data involves three stages:

- **User Prompt** – A user enters a question in the Siebel Genie APEX UI.
- **AI Orchestration** – Modular agents built using Oracle Select AI
    - Identify the relevant Siebel metadata,
    - Enforce Siebel view-mode security and,
    - Generate the final SQL statement.
- **Query Execution** – The read-only SQL query runs on Autonomous Database 23ai against the Siebel schema, and the resulting data is returned to the UI as a data table, which users can manipulate into charts as needed. Users can additionally review the underlying query if required.

Incremental Capability Gains

Siebel Genie builds on existing tools for information retrieval – not by replacing them, but by layering in a final capability: plain-language prompts that bridge data silos without new custom builds. Figure 2 below, illustrates how self-service access to Siebel data has evolved: each step expands what users can do without discarding the investments that came before.

Figure 2. Progressive evolution of self-service capabilities – Siebel Genie completes the journey enabling open ended natural language questions.



Dashboards still deliver real-time KPIs for known metrics; keyword search speeds single-record lookups; scripted digital assistants handle routine tasks. Siebel Genie closes the last gap – open-ended, cross-object questions asked in natural language which honors access controls guard-rails. Together, these layers form a continuum of insight rather than an either-or replacement.

The cumulative effect is faster decision-making, a lighter IT backlog, and fully auditable data access. A starter set of success metrics (see pg. 9 under **Recommended Success Metrics**) and target ranges that organizations can adopt or tailor to their own baselines; they are recommendations, not an exhaustive list.

Deployment Blueprint

Organisations can adopt Siebel Genie in **four incremental phases**; the table below provides a starting point for partner engagements or in-house IT plans.

Table 2. A phased implementation path to Siebel Genie

Phase	Typical Duration	Objective	Key Activities	Lead Roles*
1 – Assess	≈ 1 week	**Confirm** Scope & Security baseline	Identify high-value domains (Sales, Service), export Siebel metadata, map view-mode rules	Siebel consultant, DBA
2 – Configure	≈ 2-3 weeks	**Build** a minimum viable NLQ environment	Create Select AI credentials & profiles, build metadata search index, deploy APEX NLQ page	DBA, AI/ML Engineer, APEX Dev
3 – Pilot	Scope Dependent	**Validate** usability and governance	Onboard pilot users, refine prompts, log KPIs, run security review	Business Lead, IT Ops, Change-Management Lead

Phase	Typical Duration	Objective	Key Activities	Lead Roles*
4 – Scale	Ongoing	**Extend** value and embed support	Add domains, tune models, integrate audit logs, publish support run-book	Partner PMO, Customer IT

***Roles may be combined in smaller teams.**

Prerequisites : Autonomous Database 23ai with Select AI and APEX enabled; exported Siebel metadata; Siebel Data on Autonomous Database 23ai.

Recommended Success Metrics

The following table captures baseline KPIs recommendations & give sponsors a starting point to identify relevant metrics to optimize cost, adoption, quality, and governance. Once baseline data is captured during the discovery phase and Phase 3 (Pilot) of the Deployment Blueprint, baseline target KPIs can be redefined.

Table 3. Baseline KPIs for measuring success.

Focus Area	KPI	Why It Matters	Initial Target*
Speed to Insight	Median time from question to answer	Indicator of user productivity	**< 5 minutes**
IT Effort	% change in open ad-hoc report tickets	Measures reduction in analyst workload	**≥ 50 % drop**
User Autonomy	Share of NLQ sessions by business users	Gauges self-service uptake	**≥ 70 %**
Query Accuracy	Prompts executed without manual correction	Confirms usable SQL generation	**≥ 90 % success**
User Satisfaction	Post-pilot survey score (1-5)	Validates perceived value	**≥ 4.0**
Model Latency	Average AI-orchestration response time	Impacts user experience	**≤ 3 seconds**
Governance & Audit	Prompts and results logged	Ensures traceability and compliance	**100 % logged**

***Note:** Adjust targets after baselines are established in the pilot's first two weeks. A lightweight KPI dashboard – updated weekly – should track these metrics and feed continuous tuning of prompts, profiles, and user enablement.

Conclusion

Siebel's metadata depth and view-mode security already hold the ingredients for governed self-service analytics. **Siebel Genie** unlocks that latent value with an incremental layer: plain-language prompts converted into read-only SQL, executed on the existing schema, and delivered back to users in seconds.

Throughout this document we have tried to showcase:

- **Why it matters** – current reporting backlogs and cross-object questions slow decision-making.
- **How it works** – a three-stage flow of prompt → AI orchestration → governed execution.
- **What it adds** – the final capability in an evolution that began with dashboards and now culminates in open-ended, auditable NLQ.
- **How to deploy** – a four-phase blueprint that starts small and scales safely.

- **How to measure success** – a recommended KPI set covering speed, effort, adoption, quality, and governance.

While Siebel Genie makes it possible to access most Siebel data using natural language, it is not intended to replace existing screens or dashboards for data lookups. Users should continue using existing tools for routine lookups like "open opportunities". Siebel Genie is most effective when applied to dynamic, cross-object, or exploratory questions that benefit from conversational input and flexible reasoning – helping users improve decision speed and data-driven actions without waiting on IT.

Immediate actions for sponsors

1. **Secure executive sponsorship** for a six- to eight-week pilot in one domain (e.g., Sales).
2. **Assign roles** per the Deployment Blueprint and confirm prerequisites: Autonomous Database 23ai, Select AI, APEX workspace, and exported Siebel metadata.
3. **Run the Assess & Configure phases**, then onboard a pilot user group and track the starter KPIs weekly.
4. **Evaluate pilot results** against the recommended targets; adjust prompts, profiles, or onboarding as needed.
5. **Plan the Scale phase** – additional domains, audit integration, and support run-book – once pilot goals are met.

For architects and developers, ***Appendix A – Technical Implementation Guide*** *provides the detailed architecture, code snippets, and configuration steps referenced throughout this business overview.*

Next steps

1. **Partner Engagement:** Engage an Oracle Consulting or SI partner.
2. **Business Requirements & Success definition:** Finalise requirements, KPIs and target thresholds.
3. **Executive sponsorship:** Approve a 6-week pilot in one domain.
4. **Implementation & Pilot launch:** Configure Select AI profiles, deploy APEX UI, onboard users.
5. **Review & Expand:** Evaluate results, optimize edge cases & extend to additional domains.

Appendix A: Implementation Guide

Key Terms

Understanding these core terms will help you follow the functional and architectural discussion that follows. For a full glossary, see Appendix A.

Table 4. Reference for important terms for review when reading the implementation Guide

Term	Definition
Natural Language Query (NLQ)	A plain English question or command (e.g., "Show all open opportunities") used to retrieve data without writing SQL.
Oracle Select AI	A capability within Oracle Autonomous Database that uses generative AI to convert natural language into SQL queries.
Oracle APEX	A low-code development platform built into Oracle Database, used to build modern web interfaces with minimal coding.
Retrieval-Augmented Generation (RAG)	A technique that enriches AI responses with additional, domain-specific knowledge to improve accuracy.
AI Agent	A modular component that performs a specific task in a query workflow, such as metadata resolution or security validation.
AI Profile	A configuration object that defines which AI provider, model, and contextual metadata the system uses to process queries.
View Mode (Siebel)	A security mechanism in Siebel that controls which records users can access based on their roles or positions.
Business Component	A logical entity in Siebel that represents a business object (like Opportunity or Account) and is tied to one or more database tables.
Vector Index	A specialized index that enables semantic search over documents or metadata using AI-generated embeddings.

If you encounter an unfamiliar acronym while reading the step-by-step guide, jump to **Appendix D** for the full glossary rather than searching the web.

What this section delivers

- **Install-ready blueprint** – step-by-step instructions, reference PL/SQL, and APEX patterns for deploying Genie on the Oracle stack:
    - **Oracle Autonomous Database 23ai** – hosts Siebel data, Select AI, and its built-in *vector database* (no separate vector store required).
    - **Oracle Select AI** – converts natural-language prompts into SQL, enriched by Retrieval-Augmented Generation (RAG).
    - **OCI Object Storage** – holds domain metadata used to build vector indexes for RAG.
    - **Oracle APEX** – provides the secure, low-code UI for prompts, results, and audit visibility.

- **Modular agent design** – shows how discrete PL/SQL/Select AI "agents" resolve metadata, apply view-mode security, generate SQL, and validate results.
    - Non-OCI equivalents (e.g., OpenAI models, AWS S3 for object storage) can be substituted, however that is outside the purview of this paper.
- **Governance built in** – every example enforces Siebel view-mode rules, uses read-only DB roles, and can be configured to log prompts & SQL for audit as needed.
- **Phased rollout recipe** – Assess → Configure → Pilot → Scale, mapped to technical tasks, success metrics, and rollback checkpoints.

Who should read this

Enterprise architects, DBAs, Siebel consultants, and APEX developers who need a single, authoritative playbook for turning the high-level vision into a production-ready solution – while retaining the freedom to extend or swap cloud components as their roadmap evolves.

The following table highlights the different personas that may be involved in realizing this solution. For smaller teams, one individual may perform multiple roles highlighted in the below table:

Table 5. Personas, their designated roles & responsibilities

Persona	Role	Responsibilities
OCI Administrator	Cloud Infrastructure Lead	Provision DB, configure Object Storage, manage IAM policies.
Siebel Consultant	Siebel Architect / Developer	Extract business components, metadata relationships, and access model context.
Database Admin	ADB Specialist	Set up Select AI, create AI profiles, configure vector indexes, manage DB users.
APEX Developer	Low-Code Developer	Build user interface, configure result visualization, integrate PL/SQL workflows.
AI/ML Engineer	AI Integration Expert	Optimize embedding models and tune RAG configuration if needed.

Pre-requisites

Before implementing Siebel Genie, ensure that your environment and access configurations meet the following prerequisites.

Platform & Technology Requirements

Table 6. Key platform enablers when building Siebel Genie

Requirement	Description
OCI Tenancy	Active tenancy with permissions to manage Autonomous Database, Object Storage, and IAM.
Oracle ADB 23ai	Siebel schema must be hosted on a 23ai instance to support Select AI and vector search.
Oracle APEX	Must be enabled on the same database instance to serve as the user interface.

Requirement	Description
Oracle Select AI	Should be enabled and connected to a supported AI provider (e.g., OCI GenAI, OpenAI).
OCI Object Storage	Used for storing domain-specific metadata in natural language format for RAG.

Access and Privileges

Table 7. Access permissions required to enable setting up Siebel Genie.

Role	Required Access
ADB User	SELECT access to Siebel tables and EXECUTE privileges on DBMS_CLOUD, DBMS_CLOUD_AI, DBMS_LOCK, DBMS_SCHEDULER.
AI Provider Credentials	Includes API Key, Fingerprint, Tenancy ID, and Compartment ID (for OCI GenAI).
APEX Developer	Access to APEX workspace and relevant schema for integrating PL/SQL logic.

Deployment Considerations

- Identify a domain & relevant repository-based metadata(business components, links, joins & view modes).
- Validate region support for your chosen AI provider.

Siebel Genie Architecture

Siebel Genie has a modular architecture and leverages Oracle Select AI, Oracle APEX, and Oracle Autonomous Database 23ai to deliver natural language querying – without modifying the core Siebel application or bypassing its governance model.

Solution Architecture

Siebel Genie is composed of three core architectural layers:

i. **User Interaction**

Oracle APEX provides the user interface for entering natural language prompts and viewing query results. It also manages contextual session data, including authenticated identity, role, and position.

ii. **AI Orchestration**

This layer governs the secure translation of user prompts into executable SQL. It includes modular agents that interact with Oracle Select AI through domain-specific profiles:

a. The **Metadata Agent** identifies business components, relationships, and key schema objects based on the query context.

b. The **View Mode Agent** enforces Siebel's access control logic by checking user role, position, and view mode privileges using metadata and access-control mappings.

c. The **Query Builder Agent** constructs the final SQL using identified metadata & view-mode.

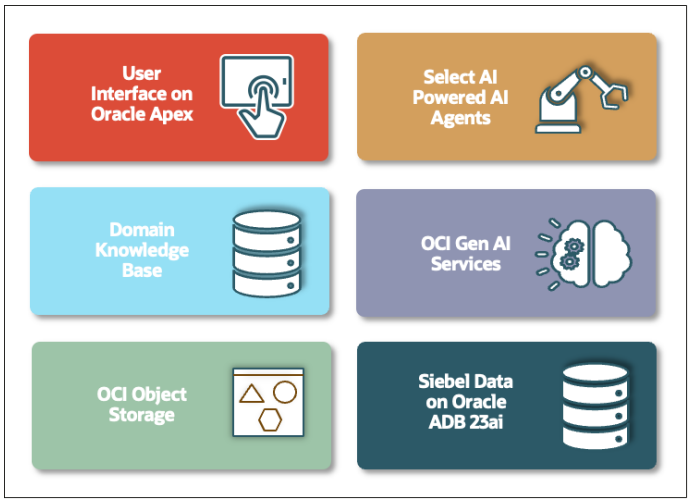
All agents invoke Oracle Select AI as the processing engine, with Retrieval-Augmented Generation (RAG) enhancing prompt understanding using Siebel-specific metadata.

iii. **Query Execution**

The final SQL query is executed against the Siebel schema within Oracle Autonomous Database 23ai. The results are returned to the APEX UI for visualization in tabular or chart formats.

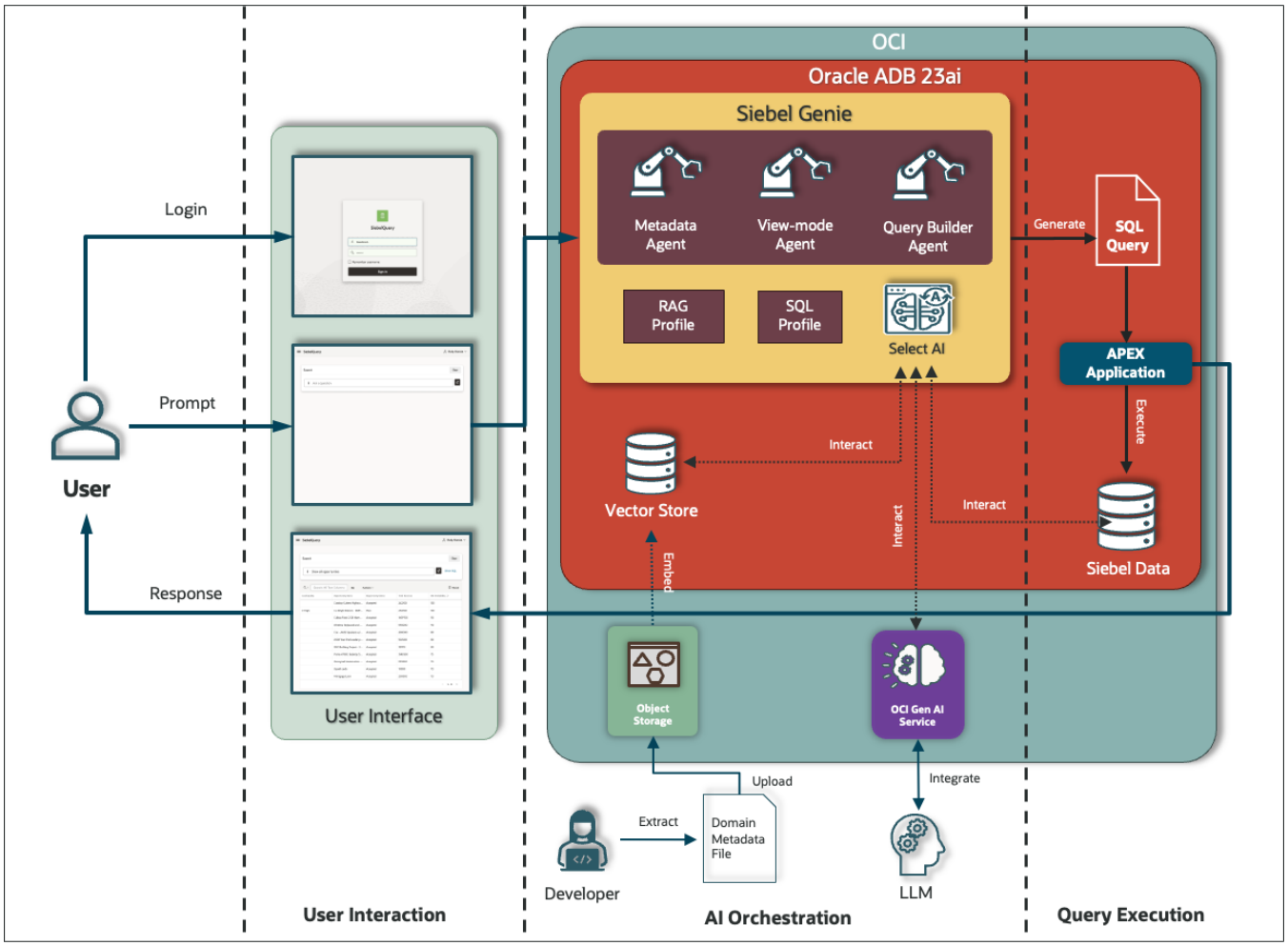
The following figure captures the different components that are utilized to build Siebel Genie:

Figure 3. Tech Stack to build Siebel Genie



The diagram below highlights at a high level the architectural view of the solution:

Figure 4. High Level Architecture for Siebel Genie



Implementation Steps

This section provides a structured, step-by-step approach to implement Siebel Genie. It is designed for technical stakeholders such as DBAs, architects, developers, and cloud administrators responsible for configuring and deploying enterprise solutions.

Siebel Genie is implemented in the following sequence:

i. **Domain Metadata Preparation**
Extract Siebel metadata and convert it into AI-consumable, natural language descriptions.

ii. **Select AI Configuration**
Set up credentials and profiles for schema-based querying, access validation, and RAG.

iii. **Vector Index Creation**
Build a vector store of domain metadata to support contextual enrichment using RAG.

iv. **AI Agent Workflow**
Implement agents that handle metadata resolution, view mode identification & final SQL generation.

v. **UI Integration**
Build the natural language interface using Oracle APEX.

vi. **Validation**
Test prompt processing, security enforcement, and result accuracy.

Each step listed above is modular, traceable, and domain-extensible.

Domain Metadata Preparation

Domain metadata is the foundation of Siebel Genie's contextual intelligence. This step involves extracting business-relevant metadata from Siebel CRM and formatting it in a natural language structure to support accurate SQL generation using Retrieval-Augmented Generation (RAG).

Step 1: Identify Functional Domains

Start by grouping Siebel business components into logical domains. This improves specificity during prompt processing.

Table 8. Starter functional domains for building Siebel Genie

Example Domains	Example Business Components
Sales	Opportunity, Account, Quote
Service	Service Request, Activity

Limit initial rollout to 1–2 domains to validate performance and refine metadata quality.

Step 2: Extract Siebel Metadata

Using Siebel Tools or Web Tools, identify and export the following key elements:

Table 9. Siebel Objects that need to be referenced from Siebel Tools when building domain metadata.

Element	Associated Data to Extract
Primary Business Component	Table name
Secondary Business Components	Table name

Join to secondary BCs	Join table and join specification
Links to secondary BCs	Link details
View modes	View mode links

Store this metadata in plain English text. Avoid tabular formats or schema dumps.

Step 3: Convert Metadata to Natural Language Descriptions

Reformat the extracted metadata into narrative statements that LLMs can understand.

Good Example:

The link name "Opportunity/Account" that describes Associated Account of an Opportunity has parent business component "Opportunity" and child business component "Account" is linked using intersection table "S_OPTY_ORG" with parent column "OPTY_ID" and child column is "OU_ID".

Avoid:

S_OPTY: S_OPTY_ID | ACCNT_ID, S_ACCNT_X: OPTY_ID, ACCNT_ID

These descriptive statements will be used as part of RAG lookups to ground the model's understanding of Siebel's schema. Through these descriptive statements, the LLM will be better able to understand relationships and associations between business components.

Step 4: Upload Metadata Files to OCI Object Storage

- Organize files by domain (e.g., /Sales/BC.txt, /Sales/links.txt,/Sales/viewmodes.txt)
- Store them in a single Object Storage bucket accessible by the Autonomous Database
- Ensure access permissions are in place (read access for DB user running Select AI)

These files will later be embedded into a vector store for semantic search.

Select AI Configuration

Oracle Select AI enables natural language-to-SQL conversion by invoking large language models (LLMs) through configurable profiles. These profiles define how prompts are interpreted, which metadata is referenced, and what credentials are used. In Siebel Genie, two AI profiles are created to support different tasks within the orchestration pipeline.

1. **RAG Profile** – The RAG profile supports multiple functional tasks through dynamic association with different vector indexes (e.g., metadata, access control).
2. **Database (DB) Profile** – Used to generate SQL queries based on physical table definitions. Profiles are domain-specific (e.g., SALES, LOYALTY).

Create AI Credentials

Before creating AI profiles, you must define credentials that allow the Autonomous Database to securely connect to your chosen AI provider (e.g., OCI GenAI, OpenAI).

Use the following procedure to create credentials for OCI GenAI (API key-based):

SQL Snippet

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OCI_GENAI_CRED',
    user_ocid       => '<USER_OCID>',
    tenancy_ocid    => '<TENANCY_OCID>',
    private_key     => '<PEM_PRIVATE_KEY>',
    fingerprint     => '<FINGERPRINT>'
  );
END;
```

- Store the credential name for use in subsequent AI profile configuration steps.
- Ensure the compartment and model identifiers are accessible under your OCI tenancy.

Create AI Profile: Retrieval-Augmented Generation (RAG)

The **RAG Profile** is used by agents that rely on domain-specific metadata embedded in vector indexes. It enables context-aware interpretation of user prompts for metadata resolution and access enforcement.

Purpose

- Powers the **Metadata Agent** and **View Mode Agent**.
- Supports retrieval-augmented generation (RAG) using embedded metadata in natural language format.
- Uses **different vector indexes** to specialize behaviour for metadata discovery and access control.

Configuration Notes

- The RAG profile is tied to an embedding model and supports semantic search.
- Each task (e.g., metadata lookup, view mode identification) is mapped to a specific **vector index**.
- This reduces hallucinations by narrowing the scope of retrieval.

SQL Snippet

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'SIEBEL_RAG_PROFILE',
    attributes   => '{
      "provider": "oci",
      "region": "us-chicago-1",
      "oci_compartment_id": "<COMPARTMENT_OCID>",
      "credential_name": "OCI_GENAI_CRED",
      "model": "meta.llama-3.1-70b-instruct",
      "embedding_model": "cohere.embed-english-v3.0"
    }'
  );
END;
```

You can later associate one or more vector indexes (e.g., sales_vector_index, vm_vector_index) to this profile based on the metadata type being queried.

Best Practices

- Use task-specific vector indexes to isolate domains (e.g., sales, view mode) and reduce cross-talk.
- Keep metadata files clean and concise – avoid schema dumps or tabular formats.
- Do not overload this profile with structured schema objects; reserve those for DB profiles.

Create AI Profile: Schema-Based SQL

The **Database (DB) Profile** is used to convert user prompts into SQL queries using structured Siebel metadata. Each business domain (e.g., Sales, Loyalty) should have a separate profile to improve model relevance and reduce noise from overlapping data.

Purpose

- Enables SQL generation using Siebel schema metadata.
- Called by the **Query Builder Agent**.

Configuration Notes

- These profiles reference Siebel tables relevant to the domain.
- Unlike RAG profiles, they do not use vector indexes or unstructured files.
- Each profile must reference the appropriate **credential**, **model**, and **compartment ID**.

SQL Snippet

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'SIEBEL_SCHEMA_PROFILE',
    attributes   => '{
      "provider": "oci",
      "credential_name": "OCI_GENAI_CRED",
      "oci_compartment_id": "<COMPARTMENT_OCID>",
      "model" : "meta.llama-3.1-70b-instruct",
      "object_list": [
        {"owner": "SIEBEL", "name": "S_OPTY"},
        {"owner": "SIEBEL", "name": "S_ACCNT"},
        {"owner": "SIEBEL", "name": "S_SR"}
      ],
      "comments": true
    }');
END;
```

Repeat the process for other domains (e.g., Loyalty, Service) using the respective schema objects.

Best Practices

- Maintain separate DB profiles per domain to prevent cross-domain inaccuracies.
- Ensure only SELECT-safe metadata objects are listed to avoid security risks.

Validate AI Profile Setup

You can validate that Select AI is working using a simple GENERATE command:

SQL Snippet

```
SELECT DBMS_CLOUD_AI.GENERATE (
  prompt       => 'Show all open opportunities for ABC Corp',
  profile_name => 'SIEBEL_SCHEMA_PROFILE',
  action       => 'narrate'
) AS result FROM dual;
```

If the response is meaningful and structurally sound, your configuration is correct.

Vector Index Creation for RAG

To support context-aware interpretation of prompts, Siebel Genie uses Retrieval-Augmented Generation (RAG). This approach allows Oracle Select AI to access to domain-specific metadata or access control rules stored in natural

language format. The information in relevant files are converted into vector embeddings and stored in a vector index, which is then linked to the RAG AI profile created earlier.

Upload Metadata to OCI Object Storage

- Ensure all metadata files (domain & access control) created during are uploaded to the object store.
- Organize files by domain or function, e.g. –

  <file_root>//siebel-metadata/sales/opportunity.txt
  <file_root>//siebel-metadata/service/sr_relationships.txt

- Store files in a bucket accessible to the Autonomous Database instance.
- Files must contain plain English descriptions; avoid raw table structures or SQL.

Create Vector Indexes

Create **multiple vector indexes**, each corresponding to a metadata category (e.g., relationship definitions, view modes). These indexes will be referenced at runtime using the same RAG AI profile.

SQL Snippet

```
BEGIN
  DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
    index_name  => 'SALES_RELATIONSHIP_INDEX',
    attributes  => '{
      "vector_db_provider": "oracle",
      "object_storage_credential_name": "OCI_GENAI_CRED",
      "location":
"https://objectstorage.<region>.oraclecloud.com/n/<namespace>/b/<bucket>/o/sales/joins/",
      "profile_name": "SIEBEL_RAG_PROFILE",
      "vector_dimension": 1024,
      "vector_distance_metric": "cosine",
      "chunk_overlap": 0,
      "chunk_size": 1536,
      "refresh_rate": 1440
    }'
  );

  DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
    index_name  => 'VIEW_MODE_INDEX',
    attributes  => '{
      "vector_db_provider": "oracle",
      "object_storage_credential_name": "OCI_GENAI_CRED",
      "location":
"https://objectstorage.<region>.oraclecloud.com/n/<namespace>/b/<bucket>/o/service/view_modes/
",
      "profile_name": "SIEBEL_RAG_PROFILE",
      "vector_dimension": 1024,
      "vector_distance_metric": "cosine",
      "chunk_overlap": 0,
      "chunk_size": 1536,
      "refresh_rate": 1440
    }'
  );
END;
```

Note: Replace <region>, <namespace>, and <bucket> with your actual Object Storage parameters.

Each index contains metadata relevant to its domain and is scoped for targeted use by specific agents (e.g., Metadata Agent, View Mode Agent).

Associate Vector Index with AI Profile

Although multiple indexes are created, **a single RAG AI profile** is reused across agents. The appropriate index is associated at runtime by setting the vector_index_name attribute dynamically:

SQL Snippet

```
BEGIN
  DBMS_CLOUD_AI.SET_ATTRIBUTE(
    profile_name    => 'SIEBEL_RAG_PROFILE',
    attribute_name  => 'vector_index_name',
    attribute_value => 'VIEW_MODE_INDEX'
  );
END;
```

This approach ensures flexibility and precision – allowing agents to retrieve only the metadata needed for their specific function, while minimizing confusion and hallucination risks.

Validate Vector Pipeline Execution

Use the following queries to verify that the embedding process completed successfully and the respective index is available for querying:

SQL Snippet

```
-- View recent pipeline executions
SELECT * FROM user_cloud_pipeline_history
WHERE pipeline_name = 'SIEBEL_METADATA_INDEX$VECPIPELINE'
ORDER BY start_date DESC;

-- Check current pipeline status
SELECT pipeline_id, pipeline_name, pipeline_type, status
FROM user_cloud_pipelines
WHERE pipeline_name = 'SIEBEL_METADATA_INDEX$VECPIPELINE';
```

Test Vector-Driven Prompt

To confirm that a vector index is properly associated and responsive, test the SIEBEL_RAG_PROFILE using a simple metadata query:

SQL Snippet

```
SELECT DBMS_CLOUD_AI.GENERATE(
  prompt       => 'What is the link between Opportunity and Account?',
  profile_name => 'SIEBEL_RAG_PROFILE',
  action       => 'narrate'
) AS result FROM dual;
```

A correct response should reference the metadata you uploaded (e.g., S_OPTY_X table or join fields).

At this point, you have completed the configuration of Select AI and prepared the metadata context for intelligent prompt handling. Next, we'll implement the AI Agent workflow that orchestrates prompt-to-SQL execution.

AI Agent Workflow Setup

Siebel Genie uses a modular agent-based design to securely interpret and process natural language prompts. Each agent performs a specific task – from metadata resolution to access validation and query generation. Agents operate independently or in sequence and are built to share context identified during prompt resolution.

This section outlines how to implement the orchestration logic using lightweight database structures and modular AI calls.

Required Metadata Table

To enable agent coordination and access enforcement, the following tables must be created:

USER_ACCESS

Captures the mapping between users, business components, and view modes.

SQL Snippet

```
CREATE TABLE USER_ACCESS (
  user_id         VARCHAR2(50),
  business_comp   VARCHAR2(100),
  view_mode       VARCHAR2(100)
);
```

RAG_METADATA

Stores agent outputs per prompt to maintain continuity across modular tasks.

SQL Snippet

```
CREATE TABLE RAG_METADATA (
  prompt_id         VARCHAR2(100),
  metadata_type     VARCHAR2(100),
  metadata_details  VARCHAR2(1000)
);
```

Modular Agent Responsibilities

Each AI agent is configured with a dedicated Select AI profile and contributes to a single stage in the prompt lifecycle. Agents are invoked independently and write their outputs using a shared prompt ID.

Metadata Agent

This agent is responsible for identifying the main business components, join relationships, and intersection tables required to fulfil the user's prompt.
It uses the **RAG Profile** with a domain-specific **vector index** to interpret metadata stored in natural language files.
The output includes:

- The identified business component
- Relationship metadata such as joins and links
- Intersection table information

View Mode Agent

This agent determines what the user is allowed to see based on their role and position. It uses the **same RAG Profile**, but with a **separate vector index** that contains view mode mappings and prompt examples. It also refers to the **USER_ACCESS** table to validate access. The output includes:

- The applicable view mode
- Validation result of user access
- View-specific filters and metadata if access is granted

Query Builder Agent

This agent creates the final SQL based on the metadata & view-mode resolved earlier. It uses a **domain-specific DB profile** (e.g., one for Sales, another for Loyalty). The output is:

- A SELECT query that reflects prompt intent and structural metadata with access control rules applied, which is forwarded to APEX for final execution.

Enabling User Interaction through Oracle APEX

Oracle APEX serves as the user interface layer for Siebel Genie, providing a secure and intuitive front end for users to interact with Siebel data through natural language prompts. It bridges user intent with the AI-powered query generation and ensures results are presented in a business-friendly format.

Role of APEX in the Solution

The APEX application captures the user prompt, invokes the Select AI agent workflow, and displays the final result. It enables secure, role-aware access without exposing the complexity of Siebel's data model or SQL syntax.

Functional Highlights

Table 10. Functional capabilities that APEX supports for Siebel Genie

Capability	Description
Prompt Input	Users submit queries in natural language via a simple text field
Context Resolution	User attributes (e.g., USER_ID, POSITION_ID) are loaded post-login
Agent Invocation	Prompt is routed to the AI workflow for metadata, security, and SQL generation
Query Execution	The final SQL query generated is executed using APEX
Result Rendering	Output is displayed using APEX components like Classic Report or Grid
SQL Visibility	Optionally display the final SQL for transparency and auditing

Why Oracle APEX

- Native to Oracle Autonomous Database; no separate infrastructure required
- Integrated with PL/SQL and DBMS_CLOUD_AI for secure backend processing
- Supports declarative UI development and dynamic report generation
- Secure session management and access control for enterprise environments

Implementation Reference

Implementation-specific details – including application items, authentication setup, dynamic SQL binding, and optional voice input are provided in **Appendix B: APEX Integration Guide**.

The appendix includes:

- Step-by-step UI setup
- PL/SQL examples
- Dynamic report handling patterns
- Optional speech-to-text integration

Operational Best Practices

Query Validation

To safeguard against malformed or unsafe SQL, the solution uses two key validation techniques:

- **Syntactic Validation**
  DBMS_SQL.PARSE() is used to check for valid SQL structure before execution. This step ensures the SQL conforms to Oracle syntax rules and can be compiled without error.
- **Query Type Enforcement**
  DBMS_SQL.DESCRIBE_COLUMNS is used to confirm that the generated query is a SELECT statement. This ensures that no DML (INSERT/UPDATE/DELETE) or non-query operations are processed.

Access Control Verification

Before execution, the solution confirms that:

- The user has appropriate access to the business component as per USER_ACCESS
- The resolved view mode is supported for the user's role, position, and BU
- The final query incorporates the correct intersection or join filters based on the user's access model

If access cannot be validated:

- A default view mode (e.g., "Sales Rep") may be applied with limited scope
- If no valid access is found, an exception is returned, instructing the user to check their responsibilities or access rights

Secure Execution

Once validated, the final SQL is executed under the following conditions:

- Run using a **dedicated read-only user** in Oracle Autonomous Database
- Executed via PL/SQL workflow triggered by APEX
- No DML permissions are granted to the execution role, ensuring query-only access

The result set is returned to APEX UI for rendering, with no intermediate file or cache storage.

Optional Audit Logging

Organizations may optionally enable prompt-level logging to support traceability and compliance. Logged attributes may include:

- Prompt text
- User identity and context (e.g., POSITION_ID, BU)
- Generated SQL
- Timestamp of execution

These logs can be stored in a secured table and integrated with enterprise audit workflows as needed.

Error Handling

If validation or execution fails:

- A user-friendly error message is returned to the APEX UI
- Errors can be logged to a system log or support dashboard for diagnostics
- No SQL is executed unless it passes both structural and access validations

This ensures the system fails safely and predictably, with no data leakage or exposure risks.

Additional Notes

- Run the application workload on a read-replicated instance of the production database to avoid performance impact on the primary.
- For asynchronous jobs (e.g., using DBMS_SCHEDULER), monitor run status:

SQL Snippet

```
SELECT OWNER, JOB_NAME, STATUS
FROM USER_SCHEDULER_JOB_RUN_DETAILS
WHERE JOB_NAME = '<JOB_NAME>';
```

- Enable auto-drop for temporary jobs to manage resource cleanup.
- Keep RAG and schema-based Select AI profiles separate.
- Avoid overly broad or overlapping domain metadata to reduce hallucinations.
- Default view modes (e.g., "Sales Rep") should only be used with explicit fallback logic and clear exception messaging.

Final Notes

This document has outlined the architecture, functional flow, implementation guide, and operational guardrails required to enable AI-powered querying of Siebel CRM data. Siebel Genie maintains compliance with Siebel's metadata and security standards, while extending usability through conversational interaction.

Organizations can use this solution as a reference model for enabling secure, scalable AI use cases on Siebel – based on their specific business needs while maintaining full control over access, data, and infrastructure.

Platform and Model Support

- **Availability**
  Select AI is available in all Oracle Cloud regions where Autonomous Database is supported.
  - OCI Generative AI is currently accessible in selected public cloud regions such as Chicago and Frankfurt.
  - Other providers like OpenAI and Anthropic are supported across multiple regions.
- **Model Switching**
  Models can be changed by updating the AI Profile. This does not require modifications to application logic.
- **Client Compatibility**
  Select AI is supported across SQLcl, SQL Developer, JDBC, Python, and more.
  Conversational mode is not currently supported in APEX or SQL Worksheet (Database Actions).

Access and Repository Considerations

- **User Access Without Repository Mapping**
  If a business component is not linked to a valid view mode in Siebel's metadata, access checks will return null – even if the user is listed in USER_ACCESS.
- **View Mode Specific Behaviour**
  For example, a user with "Organization" access but not "Sales Rep" may receive results for prompts like "Show all opportunities" but not for "Show my opportunities." This behaviour reflects accurate enforcement based on view mode configuration.

Troubleshooting Overview

Table 11. Possible Issues when working on implementing Siebel Genie

Question or Issue	Resolution
Conversations not working in APEX	Use supported clients like SQLcl or SQL Developer for conversational mode.
File or volume limits for RAG vector indexes	Individual file size limit is 4 GB. No limit on the number of files.
Scheduler job creation issues	Ensure the user has appropriate DBMS_SCHEDULER privileges.
Data Security Agent returns null access	Check both USER_ACCESS and Siebel repository view mode mappings.
Prompt works for “all” but not for “my”	Verify that the user has access to the specific view mode needed.

Appendix B: Integration with APEX

This section covers the configuration steps required in Oracle APEX to build a database application that accepts natural language queries from users and returns results in tabular or visual formats, powered by Select AI and the agent workflow established earlier.

Create APEX Workspace

Create an APEX workspace by referring to Oracle’s APEX Access to Admin Services documentation.
Important: Ensure that the Database User selected is the one configured with Select AI privileges.

Create APEX Application

From the newly created workspace, initiate a new APEX application that will serve as the front end for capturing prompts and visualizing results.

Configure Authentication Scheme

Set up an appropriate authentication method based on your deployment environment and security policies. Refer to the official APEX Authentication Guide for details.

Create Application Items

Navigate to **Shared Components → Application Items** and create the following global variables to store contextual user information:

Table 12. Global Variables for contextual user information.

Name	Purpose
USER_NAME	Stores the logged-in user’s display name
USER_ID	Stores the logged-in user’s ID
BU_ID	Stores the Business Unit ID of the user
POS_ID	Stores the Position ID of the user

Create Application Processes

Under **Shared Components → Application Processes**, define two processes to run **After Authentication**:

Table 13. Reference configurations for APEX application creation process.

Name	Purpose	Query
Load User Details	Fetch and store USER_ID and USER_NAME of the logged in user	SELECT ID, DISPLAY_NAME INTO :USER_ID, :USER_NAME FROM A_USER_DUMMY WHERE USER_ID = :APP_USER;
Load User Security	Fetch and store BU_ID and POS_ID of the logged in user	SELECT pos.ROW_ID, pos.BU_ID INTO :POS_ID, :BU_ID FROM S_PARTY par INNER JOIN S_POSTN pos ON par.row_id = pos.par_row_id WHERE par.ROW_ID = :USER_ID;

Create Page to Capture Prompt and Display Results

Create a new page in the APEX application that takes user input via text or voice and displays the result in tabular or other visualization formats.

a. ***Prompt Input and Select AI Invocation***

- Add a **Text Field** to capture the user's natural language prompt.
- Add a **Button** that calls the PL/SQL procedure generate_sql_for_user_prompt, passing:
    - USER_ID, POS_ID, BU_ID
    - The user's prompt text

Sample Invocation:

SQL Snippet

```
generate_sql_for_user_prompt(
  p_user_id      => :USER_ID,
  p_pos_id       => :POS_ID,
  p_bu_id        => :BU_ID,
  p_user_prompt  => :P1_PROMPT,
  o_sql_script   => :P1_SQL,
  o_message      => :P1_AI_MESSAGE
);
```

b. ***Visualize the Result***

There are two main approaches depending on the desired flexibility:

Using Classic Report

These are feature-rich components that display data in a tabular format and allow users to visualize the same data in other formats such as bar charts, pie charts, and more. However, as of the latest version of APEX (24.2), they require the SQL query to be configured at design time. This limitation can be overcome with additional processing of the generated SQL, making it a suitable option when multiple forms of visualization are needed. Best for static tabular outputs.

- Add a **Classic Report** region.
- Set **Source Type** to: *Function Body Returning SQL Query*
- Enable: **Use Generic Column Names**

- Set **Generic Column Count** to the maximum expected columns.
- Under **Attributes → Heading → Type**, select **Column Names**
- Configure the PLSQL function body as

SQL Snippet

```
DECLARE
   l_query VARCHAR2(4000);
BEGIN
   l_query := :P1_SQL;
   IF l_query IS NULL THEN
      RETURN 'SELECT 1 FROM DUAL WHERE 1=0';
   END IF;
   RETURN l_query;
END;
```

Using Interactive Grid/Report with Dynamic Structure

These are feature-rich components that display data in a tabular format and allow users to visualize the same data in other formats such as bar charts, pie charts, and more. However, as of the latest version of APEX (24.2), they require the SQL query to be configured at design time. This limitation can be overcome with additional processing of the generated SQL, making it a suitable option when multiple forms of visualization are needed.

- These components allow dynamic visualization (e.g., bar chart) but require the SQL query at design time.
- Workaround: Use a static query format and dynamically map output at runtime.

Logic to convert Dynamic Query to Predefined Query Format

Predefined Query Format:

```
SELECT C001, C002, ..., C009,
       N001, ..., N005,
       D001, ..., D005
FROM DUAL
```

Where:

- C001–C009: String columns
- N001–N005: Number columns
- D001–D005: Date columns

Example Dynamic Query:

```
SELECT CUST_NAME AS "Customer Name", CUST_ID AS "Customer Id"
FROM CUSTOMER;
```

Converted Query:

```
SELECT "Customer Id" AS N001,
       "Customer Name" AS C001,
       NULL AS N002, ..., NULL AS N005,
       NULL AS D001, ..., NULL AS D005,
       NULL AS C002, ..., NULL AS C009
FROM (
  SELECT CUST_NAME AS "Customer Name", CUST_ID AS "Customer Id"
  FROM CUSTOMER
)
```

Create hidden label items in the page to hold column labels:

- P1_VAR_COL_LABEL_1 to P1_VAR_COL_LABEL_9 for Strings
- P1_NUM_COL_LABEL_1 to P1_NUM_COL_LABEL_5 for Numbers
- P1_DATE_COL_LABEL_1 to P1_DATE_COL_LABEL_5 for Dates

Set **Heading** of each column in the format: &P1_VAR_COL_LABEL_1.

Use **Server-Side Condition** for each column: *Item is NOT NULL, where Item is the corresponding label field of that column*. This is to display only applicable columns.

Implementation Steps

- Create an **Interactive Report** with Source Type as *Function Body Returning SQL Query*
- Sample Function Body:

```
IF :P1_SQL IS NULL THEN
RETURN Q'[select C001, C002, C003, C004,
C005, C006, C007, C008, C009,
N001, N002, N003, N004, N005,
D001, D002, D003, D004, D005
from APEX_COLLECTIONS]';
ELSE RETURN :P1_SQL;
END IF;
```

- Create hidden page items to store column labels as described above.
- Convert dynamic SQL to the predefined query format using the above logic.
- Assign the Heading of each column as the corresponding label in the format &P1_VAR_COL_LABEL_1.
- Configure Server-side condition for each column as Item is Not Null and select the corresponding label field as the Item.

Sample Code for Query Conversion

SQL Snippet

```
DECLARE
    l_sql_script CLOB;
    l_message    VARCHAR2(4000);
    l_cursor_id INTEGER;
    l_desc_tab dbms_sql.desc_tab;
    l_col_cnt   INTEGER;
    l_char_columns varchar2(4000);
    l_date_columns varchar2(4000);
    l_num_columns varchar2(4000);
    l_char_counter number := 0;
    l_num_counter number := 0;
    l_date_counter number := 0;
BEGIN
    generate_sql_for_user_prompt(
        p_user_id     => :USER_ID,
        p_pos_id      => :POS_ID,
        p_bu_id       => :BU_ID,
        p_user_prompt  => :P1_PROMPT,
        o_sql_script   => l_sql_script,
        o_message      => l_message
    );

    :P1_AI_RESPONSE := l_sql_script;
```

```
    :P1_AI_MESSAGE := l_message;

    l_cursor_id := DBMS_SQL.OPEN_CURSOR;
    dbms_sql.parse(l_cursor_id, l_sql_script, dbms_sql.native);
    dbms_sql.describe_columns(l_cursor_id, l_col_cnt, l_desc_tab);
    :P1_COL_COUNT := l_col_cnt;

    FOR i IN 1..l_col_cnt LOOP
        IF  l_desc_tab(i).col_type = 1 then -- VARCHAR2
            l_char_counter := l_char_counter + 1;
            if l_char_counter > 9 then
                CONTINUE;
            end if;
            APEX_UTIL.SET_SESSION_STATE('P1_VAR_COL_LABEL_'||l_char_counter,
l_desc_tab(i).col_name, true);
            l_char_columns := l_char_columns || ',"'||l_desc_tab(i).col_name||'" AS
C00'||l_char_counter;
        ELSif l_desc_tab(i).col_type = 2 then -- NUMBER
            l_num_counter := l_num_counter + 1;
            if l_num_counter > 5 then
                CONTINUE;
            end if;
            APEX_UTIL.SET_SESSION_STATE('P1_NUM_COL_LABEL_'||l_num_counter,
l_desc_tab(i).col_name, true);
            l_num_columns := l_num_columns || ',"'||l_desc_tab(i).col_name ||'" AS
N00'||l_num_counter;
        ELSif l_desc_tab(i).col_type = 12 then -- DATE
            l_date_counter := l_date_counter + 1;
            if l_date_counter > 5 then
                CONTINUE;
            end if;
            APEX_UTIL.SET_SESSION_STATE('P1_DATE_COL_LABEL_'||l_date_counter,
l_desc_tab(i).col_name, true);
            l_date_columns := l_date_columns || ',"'||l_desc_tab(i).col_name||'" AS
D00'||l_date_counter;
        END IF;

    end loop;

    if l_char_counter < 9 then
        for i in l_char_counter +1..9 loop
            l_char_columns := l_char_columns ||',null AS C00'||i;
        end loop;
    end if;

    if l_num_counter < 5 then
        for i in l_num_counter +1..5 loop
            l_num_columns := l_num_columns ||',null AS N00'||i;
        end loop;
    end if;

    if l_date_counter < 5 then
        for i in l_date_counter+1..5 loop
            l_date_columns := l_date_columns ||',null AS D00'||i;
        end loop;
    end if;

    IF INSTR(l_CHAR_columns, ',') = 1 THEN
        l_CHAR_columns := SUBSTR(l_CHAR_columns, 2);
    END IF;
```

```
    IF INSTR(l_num_columns, ‘,’) = 1 THEN
        l_num_columns := SUBSTR(l_num_columns, 2);
    END IF;

    IF INSTR(l_date_columns, ‘,’) = 1 THEN
        l_date_columns := SUBSTR(l_date_columns, 2);
    END IF;

    l_sql_script := ‘SELECT ‘ || l_num_columns||
        (CASE WHEN l_date_columns is not null then ‘ , ‘||l_date_columns
        else null end )|||(CASE WHEN l_char_columns is not null then ‘ , ‘||l_char_columns
        else null end ) ||’ FROM (‘||l_sql_script||’ )’;

    :P1_SQL := l_sql_script;
    :P1_AI_MESSAGE := null;

      for c in (select page_id, region_id, report_id from APEX_APPL_PAGE_IG_RPTS
                where application_id = :APP_ID and page_id = :APP_PAGE_ID and session_id =
:APP_SESSION)
      loop
          apex_ig.clear_report(
              p_page_id => c.page_id,
              p_region_id => c.region_id,
              p_report_id => c.report_id
          );
      end loop;

EXCEPTION
    WHEN OTHERS THEN
        :P1_AI_MESSAGE := ‘Error: ‘ || SQLERRM;
END;
```

Show SQL Option

This is configured to allow a user to view the SQL generated for the prompt.

- Create a **Dialog** to show the generated SQL (i.e., the original SQL from generate_sql_for_user_prompt).
- Add a button labelled **“Show SQL”** to open the dialog.
- Do **not** show the version converted for Interactive Grid, as it may not be meaningful for users.

Enable Voice-Based Prompts

To allow users to input prompts using voice, enable speech-to-text conversion using the browser’s Web Speech API (webkitSpeechRecognition). This can be done by adding a JavaScript file to the application.

- Navigate to **Shared Components > Static Application Files**
- Click **Upload File** and add a new file named speechRecognition.js
- Use the following content for the file:

Javascript Snippet

```
if ("webkitSpeechRecognition" in window) {
  let speechRecognition = new webkitSpeechRecognition();
  let recording = false;
  let final_transcript = "";
  speechRecognition.continuous = true;
  speechRecognition.interimResults = true;
  speechRecognition.lang = "en-US";
  speechRecognition.onstart = () => {
  };
  speechRecognition.onerror = () => {
  };
  speechRecognition.onend = () => {
  };
  speechRecognition.onresult = (event) => {
    let interim_transcript = "";
    for (let i = event.resultIndex; i < event.results.length; ++i) {
      if (event.results[i].isFinal) {
        final_transcript += event.results[i][0].transcript;
      } else {
        interim_transcript += event.results[i][0].transcript;
      }
    }
    if (final_transcript) {
        apex.item("P1_PROMPT").setValue(final_transcript);
    }
  };
 document.querySelector(".llm-recording").onclick = function(){
    if (recording){
        speechRecognition.stop();
        recording = false;
        $(".fa-stop-circle").removeClass("fa-stop-circle fa-anim-flash").addClass("fa-
microphone");
    } else{
        speechRecognition.start();
        recording = true;
        apex.item("P1_PROMPT").setValue("");
        $(".fa-microphone").removeClass("fa-microphone").addClass("fa-stop-circle fa-anim-
flash");
    }
 };
} else {
  console.log("Speech Recognition Not Available");
}
```

- In the page **where the prompt field is placed**, go to **Page Attributes > JavaScript > File URLs** and add the following: #APP_FILES#speechRecognition#MIN#.js
- In the **Page Attributes > CSS > Inline add the following style.**

```
.llm-recording {pointer-events: auto !important;}
```

- The provided JavaScript sample adds an event listener to an element with the class llm-recording.
  To enable voice input:
    - Add a microphone icon (e.g., fa-microphone) next to the prompt field.
    - In the CSS Classes section of that icon/button, add llm-recording
    - This ensures that clicking the microphone icon triggers the speech-to-text function defined in speechRecognition.js.

Appendix C: AI Agent Reference Mapping

This appendix maps each AI agent in Siebel Genie to its responsibilities, the type of AI profile it uses, and the expected outputs it contributes to the orchestration pipeline.

Table 14. AI Agent Purpose

Agent	Purpose	AI Profile Used	Output Produced
Metadata Agent	Identifies business components and joins	SIEBEL_RAG_PROFILE	Metadata fragments (joins, BCs) in RAG_METADATA
View Mode Agent	Resolves view mode and enforces user access	SIEBEL_RAG_PROFILE	View mode values and access status
Query Builder Agent	Drafts final SQL Select statement	SIEBEL_SCHEMA_PROFILE	Executable, validated SQL

Additional Notes

- All AI agents are implemented using **modular DB procedures** orchestrated by a PL/SQL controller.
- Intermediate data such as metadata and view mode resolution are stored in **RAG_METADATA** and validated using **USER_ACCESS** mappings.
- Agents can be extended or swapped independently by modifying their associated AI profile or task logic.

Appendix D: Glossary of Key Terms

Table 15. Key Terms

Term	Definition
AI Agent	A modular, task-specific logic unit that performs a discrete function in the natural language to SQL orchestration (e.g., metadata extraction).
AI Profile	A configuration object in Oracle Autonomous Database that defines the LLM provider, model, credentials, and contextual metadata to be used.
APEX (Oracle APEX)	A low-code development framework used to build the user interface layer, allowing users to submit prompts and view results.
Autonomous Database	Oracle's cloud-native, self-managing database platform. The 23ai release supports Select AI and vector-based querying.
Business Component	A Siebel-specific metadata construct representing a logical business entity, mapped to one or more physical tables.
DESCRIBE_COLUMNS	A PL/SQL function used to verify that a generated SQL query is a SELECT statement and not an unsafe or malformed operation.
Intersection Table	A join table in Siebel used to model many-to-many relationships between business components.

Term	Definition
Join Logic	The relationships between Siebel tables and business components, resolved by the Metadata Agent to form a valid SQL query.
Metadata Agent	An AI agent responsible for retrieving the primary business component, join structures, and context needed for query generation.
Natural Language Query (NLQ)	A plain English prompt submitted by the user (e.g., “Show all open opportunities for ABC Corp”).
OCI	Oracle Cloud Infrastructure, the platform that hosts Autonomous Database, APEX, and AI services.
Query Builder Agent	An agent that generates a draft SQL statement based on prompt intent and business domain metadata.
RAG (Retrieval-Augmented Generation)	A method to improve LLM output by enriching prompts with relevant domain-specific knowledge or metadata.
RAG_METADATA Table	A table used to store intermediate metadata generated by agents for a given user prompt.
Select AI	An Oracle Autonomous Database feature that uses generative AI to convert natural language prompts into SQL queries.
USER_ACCESS Table	A custom table used to store user-specific access permissions, including view mode and business component mappings.
Validation Layer	The combination of syntax and access control checks applied before executing a generated SQL query.
Vector Index	A structure built from metadata documents to enable semantic search via embeddings in RAG workflows.
View Mode	A Siebel security setting that determines the type of records a user can access based on their role or position.
View Mode Agent	The agent responsible for validating the user’s access rights to specific Siebel data based on their assigned view modes and repository links

ORACLE

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle