Flowgentic Workflow Design Document

Data Lineage Agent – User Story 4 (Automated Data Lineage & Audit Trail)

Mahesh Kumar Tippanu

November 29, 2025

EPIC: Pre-Built AI Agents for Data Governance

This epic focuses on building plug-and-play intelligent components that support enterprise-grade data governance. One of the most crucial elements in governance is understanding how data flows, who modifies it, and how it transforms over time. The Data Lineage Agent enables automated lineage tracking, auditing, and traceability across multiple database tables.

User Story 4: Automated Data Lineage & Audit Trail

As an Auditor,
I want an agent that generates a complete lineage report for a data flow,
so that I can understand how data originated, moved, and transformed across the system.

Acceptance Criteria

- The agent can be triggered on request for any pipeline or process.
- It must trace source systems, intermediate transformations, and final outputs.
- It should generate a human-readable audit report with timestamps, transformations, and responsible entities.

1 Theoretical Background

Data lineage provides a complete historical record of how data enters, transforms, and exits various systems. It is foundational for auditing, debugging, compliance (GDPR, HIPAA), and understanding the impact of pipeline changes.

The Data Lineage Agent uses a combination of:

- PostgreSQL lineage tables,
- A custom lineage logging function,

- LLM-powered interpretation,
- Flowise workflow orchestration.

This produces an automated, accurate, and intelligent trace of end-to-end data transformations.

2 Lineage Architecture Overview

2.1 1. Start Node: User Trigger

The workflow begins when the auditor enters a question, such as:

- "Give lineage for InterviewID 1001"
- "Who updated CaseID 9001?"
- "Show the full audit trail for reviewer operations"

This user query initializes the lineage-extraction process.

2.2 2. SQL Generator Node: Controlled SQL Creation

An LLM generates a PostgreSQL query that **always queries only the** `lineage_steps` **table**. It must never access business tables directly. All lineage is obtained from metadata logs.

The SQL output follows rules:

- Only SELECT queries
- Always sorted by timestamp
- Filter by matching InterviewID, ReviewerID, CaseID, or keywords

2.3 3. SQL Validator Node: Ensuring Query Safety

This node inspects the generated SQL and ensures:

- No DML (INSERT/UPDATE/DELETE)
- Query references `lineage_steps` only
- No access to raw business tables

Invalid queries are regenerated automatically.

2.4 4. PostgreSQL Query Node: Metadata Retrieval

Once validated, the SQL query executes on the PostgreSQL database. This fetches records from:

`lineage_steps`

Each entry includes:

- Role (Interviewer/Reviewer)
- Input Table
- Output Table
- Operation Type
- Description
- Changed Row ID
- Timestamp

These represent the true lineage path.

2.5 5. Decision Node: Result Check

If a query does not return any results, the workflow:

- Attempts to regenerate a better SQL query, OR
- Returns a "No lineage found" response.

This ensures resilience against malformed user queries.

2.6 6. Lineage Interpretation Node (LLM)

The LLM converts database rows into a clean, auditor-friendly explanation:

- Step-by-step lineage
- Human-readable descriptions
- Role-based accountability
- Time-ordered transformation history

This turns technical metadata into actionable insights.

2.7 7. Output Node: Final Audit Report

The system presents the lineage in readable form, such as:

```
1. Interviewer created InterviewID 1001 in tblInterview
2. Interviewer added InterviewDetailID 2001 in tblInterviewDetail
3. Interviewer added ResponseID 3001 in tblInterviewAnswer
4. Reviewer assigned ReviewerID 4001 in tblReviewer
5. Reviewer updated review decision in tblReviewerChange
```

This fulfills the end-to-end lineage requirement.

3 Lineage Workflow Pipeline

Step-by-Step Overview

1. **Start Node** receives user query.
2. **SQL Generation Node** creates a PostgreSQL SELECT query.
3. **Validation Node** checks query correctness.
4. **Database Query Node** fetches lineage metadata.
5. **Decision Node** handles empty or invalid results.
6. **Interpretation Node** formats lineage into human-readable form.
7. **Output Node** displays the final audit report.

4 Workflow Diagram (Text Representation)

```
(User Question)
        ↓
(Generate SQL Query Node)
        ↓
(Validate SQL Node)
        ↓
(PostgreSQL Query Node)
        ↓
(Check if Results Found?)
        ↓ YES
(Lineage Explanation Node)
        ↓
(Output Node)
```

Conclusion

The Data Lineage Agent provides an automated, accurate, and transparent audit mechanism. It reconstructs the complete flow of information—from initial data creation to final review—enabling auditors, analysts, and developers to trace activities, validate transformations, and ensure compliance across the system. By integrating this workflow in Flowise, teams achieve consistent traceability, reduce manual investigation time, and strengthen data governance infrastructure.