KOIOS · ORACLE STACK

via

Visual Interactive Analysis

Column-level lineage, forward impact analysis, and time-travel across your entire db — interactive, and 100% Oracle-native.

LINEAGE IMPACT TIME-TRAVEL

KOIOS.

MOTIVATION

Why We Built VIA

No Column-Level Lineage

Tracing a column through many layers was manual, slow and error-prone — only senior developers could do it.

ODI / METADATA

No Impact Analysis

Source column changes broke things downstream — and the failures were silent, discovered only after the fact.

IMPACT / NO ALERTS

No Historical View

Once a mapping was overwritten, the previous version was gone forever — no way to look back in time.

SCD2 / NO VERSIONS

Manual Maintenance

Thousands of ODI mappings tracked entirely by hand. Zero automation, and endless, repetitive upkeep.

MANUAL / MAPPINGS

SYSTEM WALKTHROUGH

How VIA Works — Step by Step

1

Metadata Repository

Read mappings from any ETL tool

2

SCD2 Tracking

Capture every change over time

3

Graph Inference

Resolve full lineage chains

4

Output Tables

Lineage · Impact · Intersections

5

APEX + Graphs

Interactive temporal property graphs

100% ORACLE-NATIVE | No external tooling · no additional licensing

SYSTEM WALKTHROUGH

1 Metadata Repository

Metadata Sources

Any ETL / EL-T tool with accessible metadata is supported.

Column Expressions

Transformation logic — joins, filters and derivations per column.

ETL TOOL
Any ETL / EL-T

METADATA
Column mappings

COLUMNS
All attributes

COLUMNS NEEDED — THE BARE MINIMUM

1 · MAPPINGS & FOLDERS

`FOLDER_NAME` `EXECUTION_NAME` `MAPPING_NAME`

2 · MAPPING TABLES

`SCHEMA_NAME` `TABLE_NAME` `TABLE_TYPE` `TABLE_NAME_IN_MAPPING` `DIRECTION`

3 · COLUMN EXPRESSIONS

`OBJECT_TYPE` `COLUMN_NAME` `EXPRESSION`

4 · JOINS

`JOIN_NAME` `JOIN_TYPE` `JOIN_POSITION`

5 · PHYSICAL COLUMN CATALOG

`REFERENCE_NAME`

5 simple extracts, ~15 columns — all VIA needs. That's exactly why any ETL / EL-T tool's metadata plugs straight in.

SYSTEM WALKTHROUGH

2 SCD2 Tracking

Temporal Tracking

Full history preserved — nothing is ever deleted, only added and closed.

Loads From One Source

The same SCD2 tables feed both the initial load and incremental loads.

SCD2
Type 2 dimension

TRACKING
Historic versions

HISTORY
Time-travel

HISTORIC TABLE — ONE ROW PER VERSION

VERSION	VALID_FROM	VALID_TO	IS_CURRENT
v1	2024-01-01	2025-03-10	N
v2	2025-03-10	2026-02-01	N
v3	2026-02-01	—	Y

SYSTEM WALKTHROUGH

3 Graph Inference

Graph Builder

Turns metadata into nodes & edges — one node per column.

Inference Engine

Rule-driven traversal that resolves every dependency.

GRAPH
Node network

INFERENCE
Rule-driven

RESOLVE
Full chain

RULE-DRIVEN RESOLUTION

GRAPH BUILDER

↓

INFERENCE ENGINE

Iterative and rule-driven — until every chain is traced.

SYSTEM WALKTHROUGH

4 Output Tables

Three SCD2 Tables

Each output is itself fully SCD2-tracked and historised.

Lineage, Impact and Intersections — together they give both the detail and the big picture.

← LINEAGE

Detailed **backward** trace — column to raw source.

→ IMPACT

Detailed **forward** trace — what each column feeds.

↔ INTERSECTIONS

Bi-directional **big picture** across the db.

SYSTEM WALKTHROUGH

5

APEX & Graphs

APEX Interface

Interactive UI — the explorer you'll see live in a few minutes.

Property Graphs

Temporal graphs queried via `GRAPH_TABLE()`, rendered with Cytoscape.js — Oracle's GVL plug-in is also supported.

Offline Export

One-click download of the current graph — any view, filters and as-of date — as a self-contained interactive HTML file that opens anywhere.

APEX
Oracle APEX

CYTOSCAPE
Rendering

QUERY · GRAPH_TABLE ( )

```
SELECT * FROM GRAPH_TABLE (
    via_lineage
    MATCH (c) -[e IS FEEDS]-> (t)
    WHERE c.column = :selected
    COLUMNS ( t.schema, t.table,
              t.column, t.layer )
);
```

SQL/PGQ · Cytoscape.js · APEX plug-in

SYSTEM WALKTHROUGH

How the Inference Engine Works

1

Build the Graph

Nodes & edges from SCD2

2

Start Iteration

Count unresolved nodes

3

Apply Rules

Expand & mark resolved

4

Node States

Unresolved → resolved

5

Convergence

Exit when all traced

↺ repeat steps 2–4 until convergence

INFERENCE ENGINE

1 Build the Graph

Graph Structure

1 node = 1 column · 1 edge = 1 dependency

Dependencies

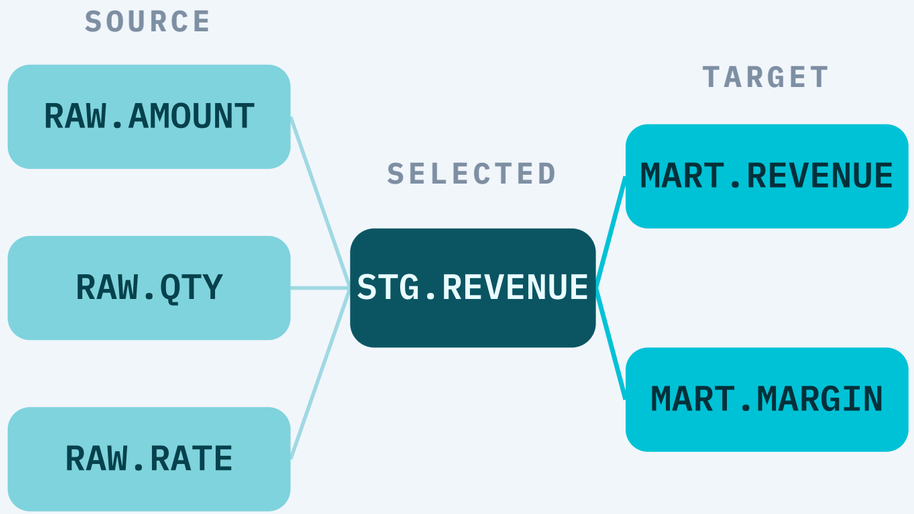
Edges encode the transformation dependencies between columns.

NODES
Column nodes

EDGES
Dependencies

SCD2
Source tables

ONE NODE PER COLUMN, WIRED BY DEPENDENCY



INFERENCE ENGINE

2 Start Iteration

Start the Loop

Begin a fresh iteration over the unresolved graph.

Count Nodes

Track progress each pass to know when to stop.

ITERATE
Fresh pass

COUNT
Unresolved & partial

RULES
Priority order

EACH PASS

- ✓ Start a **fresh pass** over the graph
- ✓ Count **UNRESOLVED** and **PARTIAL** nodes
- ✓ Run resolution rules in **priority order**

INFERENCE ENGINE

3 Apply Rules

Expand Transformations

Traverse the dependency chain — resolve column references, expand transformation values.

Mark Resolved

Tag **SOURCE** nodes once their chain is fully explained.

EXPAND
Transformations

RESOLVE
Column refs

RULES
Priority order

TWO RULES, APPLIED IN PRIORITY ORDER — EVERY PASS

R1

Expand

Substitute each column reference with its defining expression; strip aliases to physical **schema.table.column**; walk through views and pass-through layers, pass after pass.

R2

Mark Resolved

A chain terminates at a physical source column, a named terminal (constant, variable, sequence), or a null reference.

Filters and joins are resolved the same way — down to physical columns, not just the select-list expressions.

INFERENCE ENGINE

4 Node States

State Machine

Three states drive every column from raw input to a fully traced chain.

Only **TARGET** nodes are driven all the way to RESOLVED.

UNRESOLVED

Not yet visited — no expression traced.

PARTIAL

Expansion started — references still unexplained.

RESOLVED

Every reference explained down to a terminal.

INFERENCE ENGINE

5 Convergence

Inference Complete

Done successfully when all **target nodes = RESOLVED**.

In practice, 99%+ of columns resolve in under 10 iterations.

COMPLETE
All resolved

EXIT
Loop ends

SUCCESS
Converged

LOOP EXITS WHEN ANY IS TRUE

- ✓ **UNRESOLVED = 0** — all targets fully traced
- ✓ **No progress** — count before = count after
- ✓ **100 iterations** reached

DATA SOURCES UNDER ANALYSIS

Two Datasets, One VIA

VIA tracks metadata, lineage, and impact across both data domains.

NFL Stats Database — 5 layers

1. IMPORT Layer
2. STAGE Layer
3. **HISTORY Layer**
4. ANALYTICS Layer
5. PREDICTION Layer

VS

Shipping Company DB — 6 layers

1. RAW Data
2. IMPORT Layer
3. STAGE Layer
4. **HISTORY Layer**
5. DATA MARTS — 1st Layer
6. DATA MARTS — 2nd Layer

FROM THE FIELD — ORACLE PROPERTY GRAPHS

Best Practices — Making Property Graphs Fast

01 — INDEXING

Index the columns the graph walks on

A graph over MVs full-scans on **every hop** (one hop = following a single edge, column → its source).
Index the **node & edge keys**, the `schema.table.column` **identity**, and the **validity dates** — on the graph MVs *and* the fact tables.

FULL SCAN → INDEX — EVERY GRAPH HOP

~1,200 ms → ~20 ms — AS-OF DATE LIST

02 — LEAN GRAPH

Keep heavy payloads out of the graph

Vertices & edges carry **identity + validity dates, nothing else**.
Transformation / filter / join CLOBs stay in indexed fact tables, fetched on node-click — the walk never drags CLOBs along.

10–30 ms — CLOB FETCH ON DEMAND

AND THIS IS ALL ON THE OCI FREE TIER

Column-level, time-travel lineage over ~80k edges — renders 25,000–32,000 ms → ~300 ms **(~100×)**.

	MAPPINGS	COLUMN NODES	LINEAGE EDGES
Shipping db	3,300	35,607	77,532
NFL stats db	72	1,503	2,250
Whole graph	**3,403**	**37,233**	**79,964**

KEY CAPABILITIES

What You Just Saw

SCD2 · HISTORY

Column-Level Lineage

Trace any column up to its raw source.

METADATA · TRUTH

Forward Impact Analysis

Instant downstream impact view.

SQL/PGQ · GRAPHS

Temporal Snapshots

Query any date — SCD2 + property graphs.

INSIGHT · OVERVIEW

Interactive Graph

Live graph, instant database overview — downloadable as offline interactive HTML.

RESULTS

What VIA Delivers in Practice

SPEED

1h→5min

Scale & Speed

Automatic tracking, end to end.

CLARITY

Raw→Mart

Transparency & Clarity

Every layer, fully visible.

RESILIENCE

Any date

Resilience & Diagnosis

Complete history, instant queries.

FLEXIBILITY

Any ETL

Flexibility & Automation

Zero manual work, any tool.

SOLUTION ARCHITECTURE

Metadata to Interactive Graph — Fully on the Oracle Stack

Works with any ETL / EL-T tool.

01

ETL Metadata

Transformations, joins, filters.

02

Historic Tables (SCD2)

Track changes, time-travel.

03

Graph Inference Engine

Build nodes/edges, run inference.

04

Output + Property Graphs

Lineage · Impact · Intersections.

05

Oracle APEX

Interactive explorer.

WHAT'S NEXT

VIA Is Actively Evolving

AI

AI-Assisted Explanations

LLM pseudocode for non-developers.

CONNECTORS

Broader ETL Support

dbt, Informatica, PL/SQL, and more.

MULTI

Multi-Workspace

Multiple schemas and teams.

OPEN

Open for Collaboration

Bring us your metadata challenge.

KOIOS

Thank you for your time

info@koios.hr

tomislav.zarkovic@koios.hr