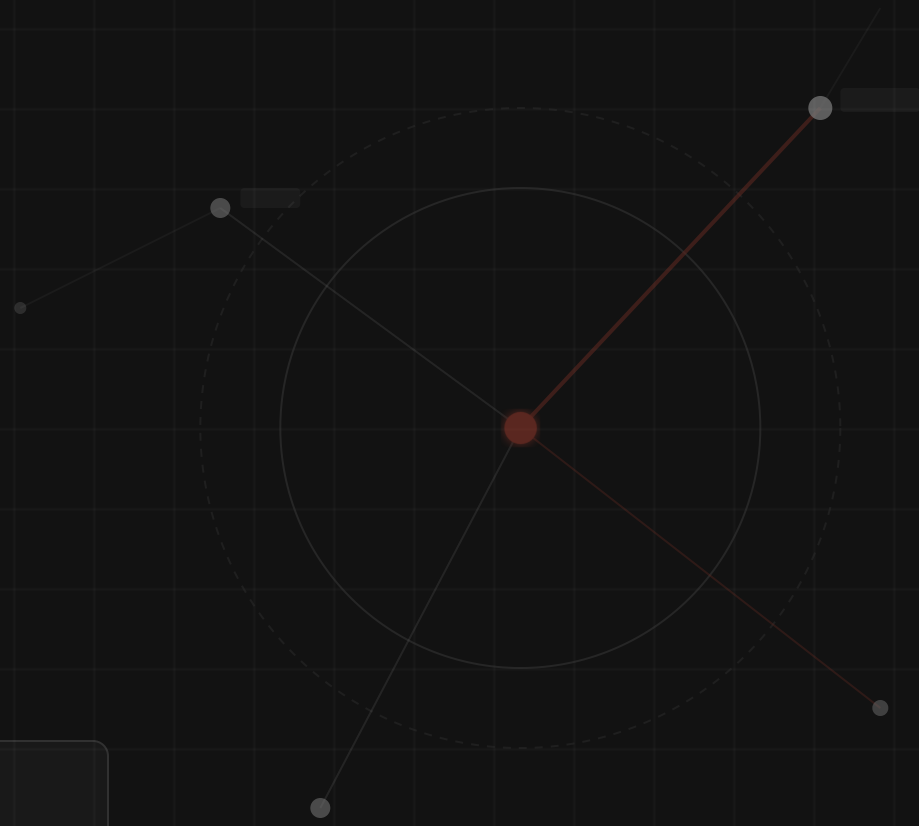
APEX WORLD
2026 CONFERENCE

GRAPH ANALYTICS

SEEING THE CONNECTIONS

Visualizing Graph Data in **Oracle APEX**



> *"Your data already has relationships.*
> **Graphs just make them visible.**"

PRESENTED BY
Marcelo Batalha

Marcelo Batalha

Computer Engineer • Grupo Giap

Oracle APEX • PL/SQL • REST • AI

Experience

- 25+ years in software development
- Computer Engineer
- M.Sc. in Computer Engineering
- Postgraduate professor (Full-Stack)
- Solutions architect & integration specialist
- ORACLE ACE Associate

Technologies

- Certifications: Oracle APEX • Vector Search
- Oracle APEX • PL/SQL
- REST APIs • ORDS
- Data & AI for enterprise apps
- Graph & spatial analytics (Oracle)

Events

- 🇧🇷 LAOUC / GUOB Tech Day • AI Brasil Experience (São Paulo)
- 🇧🇷 Oracle APEX Tour (Brasília)
- 🇧🇷 Oracle Academy / Office Hours
- 🇧🇷 Oracle ACE Connect Brasil (São Paulo)
- 🇺🇸 Kscope (Nashville 2024 • Dallas 2025)
- 🇳🇱 APEX World (March 2026)
- 🇺🇸 Kscope (June 2026)

THE PROBLEM

Why Relationships Are Hard to See

Strong at Storage & Transactions

Relational databases are excellent for storing structured data and handling complex transactional workflows efficiently.

Hidden Behind Joins

Relationships are usually implicit, buried within foreign keys and join tables, making the structure hard to perceive at a glance.

Difficult to Visualize Context

When we ask questions about connections, standard tabular output lacks the visual context needed to understand complex dependencies.

The Invisible Reality

"Relationships exist, but we rarely see them."

THE FUNDAMENTALS

What Is a Graph?

Vertex → An Entity

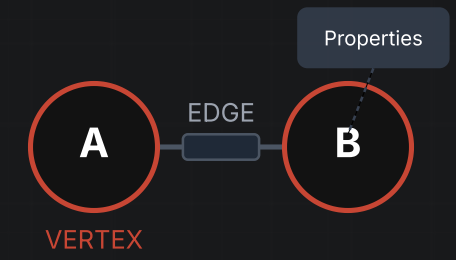
Represents a discrete object, such as a Person, Account, or Location. In relational terms, this is a row in a table.

Edge → A Relationship

Represents a connection between two vertices. It captures how entities relate (e.g., "Works For", "Transferred To").

Properties → Attributes

Key-value pairs that describe details of both vertices and edges (e.g., Name, Date, Amount).



Concept Bridge

"If you know tables, you already know graphs."

FUNDAMENTAL DIFFERENCE

Tables Store Data. Graphs Reveal Context.

Shifting from storing records to analyzing relationships changes how we ask questions.

RELATIONAL THINKING

THE TRADITIONAL VIEW

Rows

Discrete records of information

Joins

Assemble data at query time

Structure

Schema-defined rigid format

VS

GRAPH THINKING

THE CONNECTED VIEW

Connections

Relationships are first-class citizens

Traversal

Follow the path of data naturally

Relationships

Understanding context and patterns

THE FOUNDATION

Starting from Tables

Standard Relational Tables

Our example begins with two simple tables: `AW_PEOPLE` (entities) and `AW_RELATIONSHIPS` (connections).

No Data Migration

The data stays exactly where it is. There is no complex ETL process or need to move data into a specialized graph database.

No Duplicated Data

The graph is defined as a logical layer. We do not replicate the data, ensuring a single source of truth and real-time consistency.

THE SOLUTION

Property Graph Concept

A Logical Overlay

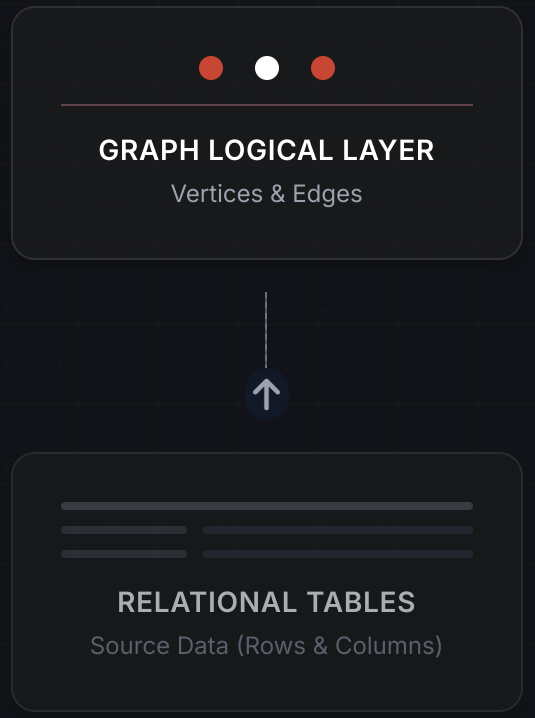
The graph isn't a copy of your data. It acts as a logical layer sitting directly on top of your existing relational tables, providing a new way to view the same information.

Direct Row References

Vertices (entities) and **Edges** (relationships) map directly to rows in your source tables. No data migration is required.

Attributes as Properties

Column data becomes graph **Properties**. These key-value pairs attach to vertices and edges, driving visualization styles and analytical depth.



STRUCTURE DEFINITION

Creating a Property Graph

Defines the logical graph layer on top of relational tables

create_graph.sql

```
CREATE PROPERTY GRAPH aw_people_graph
  VERTEX TABLES (
    aw_people
      KEY (person_id)
      LABEL person
      PROPERTIES (
        full_name AS caption,
        team,
        city,
        hire_year
      )
  )
  EDGE TABLES (
    aw_relationships
      KEY (rel_id)
      SOURCE KEY (src_person_id) REFERENCES aw_people (person_id)
      DESTINATION KEY (dst_person_id) REFERENCES aw_people (person_id)
      LABEL rel
      PROPERTIES (
        rel_type,
        since_year
      )
  );
```

Declarative Definition

No data is moved. The graph is created logically over your existing tables using keys and references.

PATTERN MATCHING

Querying the Graph

SQL syntax extended for graph pattern matching

graph_query.sql

```sql
SELECT *
FROM GRAPH_TABLE (
  aw_people_graph
  MATCH (a IS person)-[e IS rel]->(b IS person)
  COLUMNS (
    a.full_name AS person_a,
    e.rel_type,
    b.full_name AS person_b
  )
);
```

Thinking in Connections

The `MATCH` clause uses ASCII-art style syntax to visualize the pattern directly in your code. You aren't just joining tables; you are describing the path.

THE MECHANICS

Traversal: Following Relationships

Move Through Connections

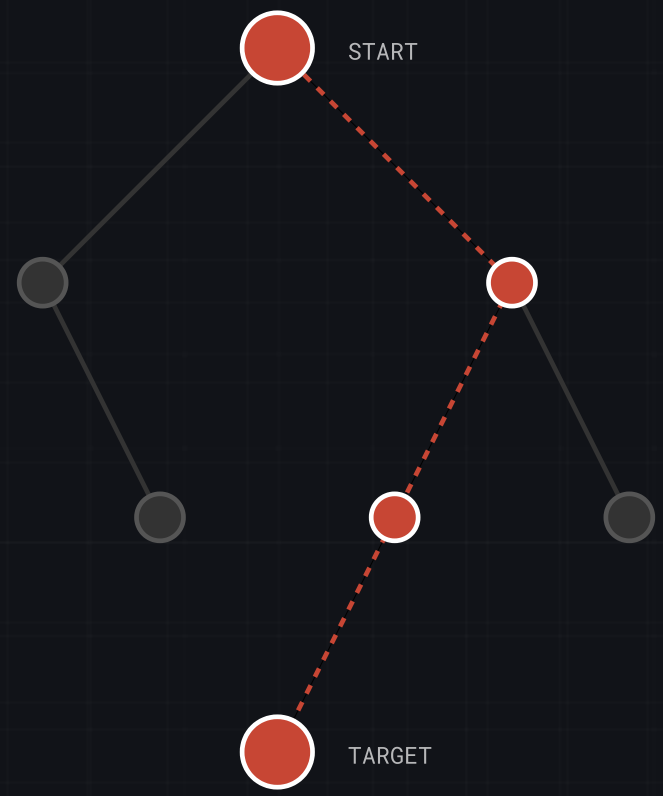
Instead of computationally expensive table joins, we simply "walk" the graph by following direct pointers between entities.

Natural Questioning

Queries align with how we think: "Who is connected to whom?" and "Through whom does this connection exist?"

Measuring Distance

Understanding "degrees of separation" becomes a native operation, allowing us to see how far apart entities really are.



PATH FOUND

3 Hops to Target

Traversal executes in milliseconds without joins.

OBSERVATION GUIDE

What to Look For in the Demo

Connections (Structure)

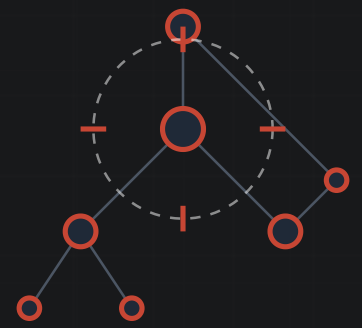
Observe how entities naturally form clusters, hubs, and bridges without manual grouping.

Distance (Paths)

Look for the "degrees of separation" and shortest paths connecting seemingly unrelated entities.

Change Over Time

Watch how the network evolves—new nodes appearing and connections strengthening or dissolving.



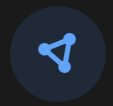
Unified Graph Model

One graph serving structure, pathing, and time analysis simultaneously.

Demo: Network

VISUALIZING CONNECTIONS IN REAL-TIME

WHAT TO LOOK FOR

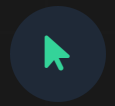


Structure

Observe the layout of **nodes and edges** revealing the underlying data shape.

Clusters

Identify **communities** and central hubs that act as bridges.



Interaction

Watch how **filters and tooltips** provide immediate context.

But networks don't stay still.

What happens when we add **time** to the graph?

SPATIAL

TRAVERSAL

Movement through

Connections

```
(a) --> (b) --> (c)
```

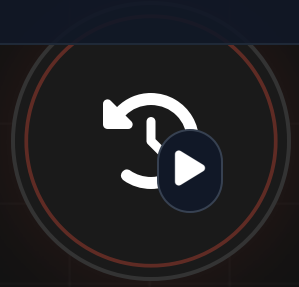
TEMPORAL

EVOLUTION

Movement through

Time

```
t1 --> t2 --> t3
```



Demo: Evolution

VISUALIZING GRAPH DYNAMICS OVER TIME

WHAT TO LOOK FOR

Temporal Animation

Watch the graph grow organically as we **animate by year**.

Emerging Links

Observe how **new connections** form and densify the network.

Snapshots

Compare distinct states of the network at **different points in time**.

IMPACT OF GRAPH ANALYSIS

What Changed?

Moving from static snapshots to dynamic evolutionary understanding.

TRADITIONAL VIEW

STATIC SNAPSHOTS

Static query

Fixed questions at a point in time

Records

Individual rows of data

Isolated data

Silos without temporal context

VS

GRAPH VIEW

DYNAMIC EVOLUTION

Exploration

Navigating through time and connections

Relationships

Dynamic links forming over time

Connected context

The full picture of evolution

WHERE THIS APPLIES

Real-World Use Cases

Fraud Rings & AML

Detecting circular payments, hidden identities, and money laundering networks.

Supply Chain Risk

Analyzing impact propagation and identifying single points of failure in logistics.

Organization Insights

Mapping influence, communication flows, and informal leadership structures.

Recommendations

Finding similarity paths for product suggestions based on user behavior graphs.

IT Dependencies

Root cause analysis by tracing upstream and downstream system dependencies.

Access Control

Visualizing role hierarchies and entitlements to ensure security compliance.

Universal Pattern

"Whether it's money, goods, or people, the underlying graph structures reveal the same fundamental truths."

THE VERSATILITY

Same Graph, Multiple Perspectives

Network Topology

Identify structural elements like hubs, bridges, and communities to understand organization and influence.

Paths & Shortest Routes

Discover connections and calculate the most efficient ways to traverse between distinct entities.

Timelines & Evolution

Visualize how relationships form, break, and evolve, moving beyond static snapshots.

Unified Data Source

"Different questions, same underlying graph."

ARCHITECTURE & TOOLS

How This Demo Was Built

Graph Visualization Plug-in

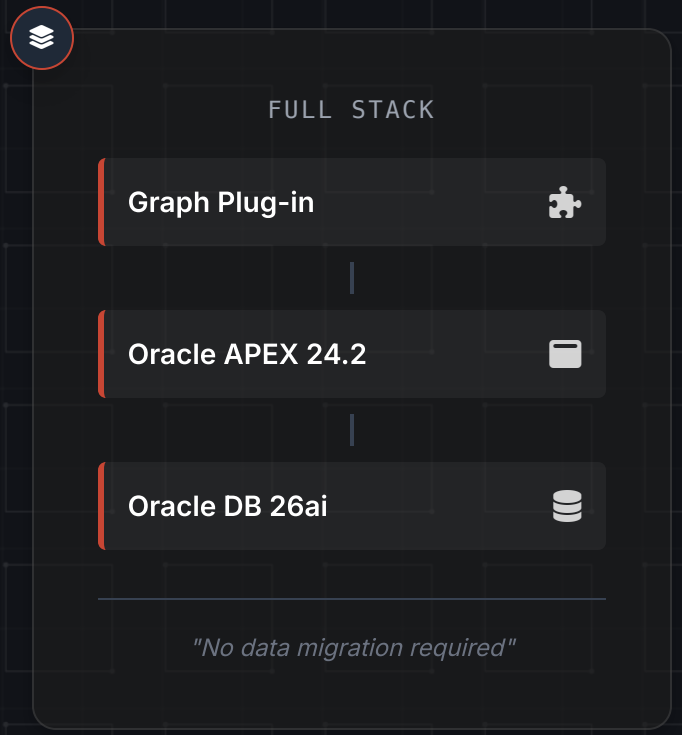
Uses the Oracle Graph Visualization plug-in for APEX. Available for download from the official Oracle APEX GitHub repository (release 24.2).

Sample Application

Oracle provides a "Sample Graph Visualizations" app containing comprehensive usage examples to help you reproduce this experience.

The Tech Stack

Built on `Oracle Database 26ai` and `APEX 24.2`. Leveraging standard Property Graphs and native `GRAPH_TABLE` queries.



The graph didn't create new data.

It revealed what was already there.

VISUALIZATION IS CONTEXT

THANK YOU.

APPRECIATE YOUR TIME

Get in Touch

Have specific questions about the graph implementation?

batalha@me.com

Connect

Follow for more Oracle APEX and Graph tips.

Resources

Download graph plug-in and see instructions.

github.com/oracle/apex →

NL.OUG
NL ORACLE USER GROUP

Seeing the Connections: Visualizing Graph Data in Oracle APEX-Marcelo

Please fill in your evaluations