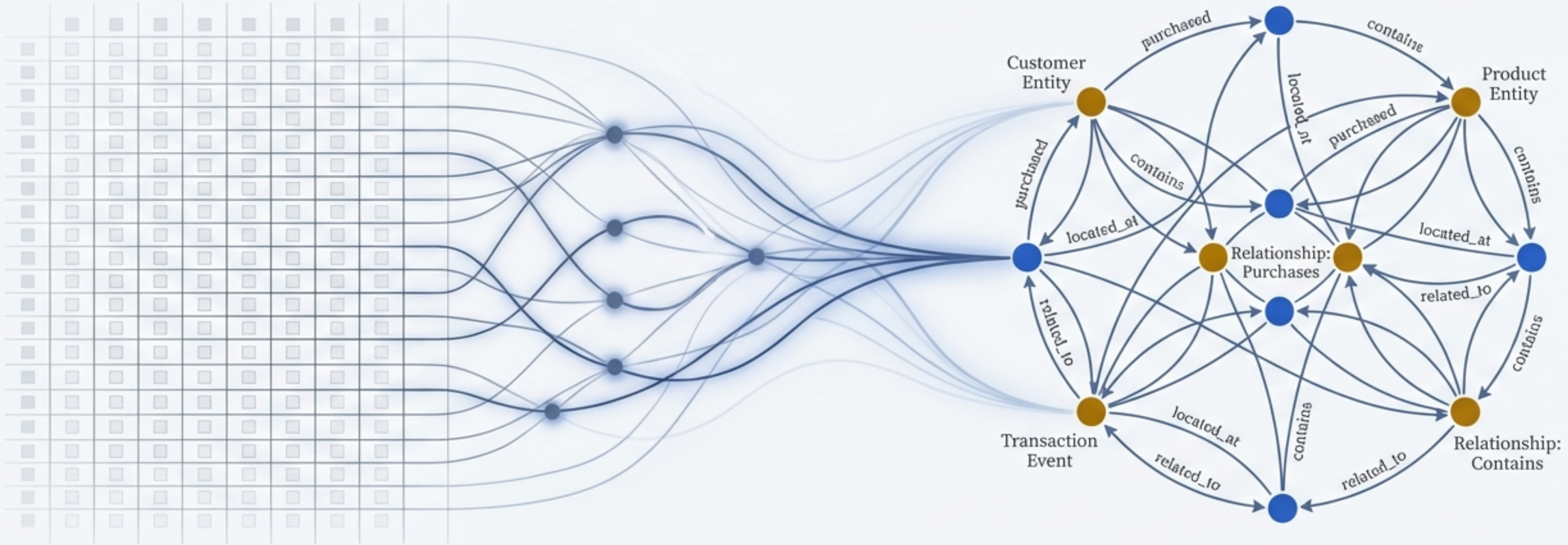
From SQL Tables to a Live Knowledge Graph

A Guided Demonstration of AI-Augmented Data Transformation



OpenLink Virtuoso | OPAL OpenLink AI Layer

NotebookLM

The Objective: Transforming a Legacy Asset with with a Modern Tool

1. The Asset: The Northwind Database

A well-known, industry-standard relational database schema.

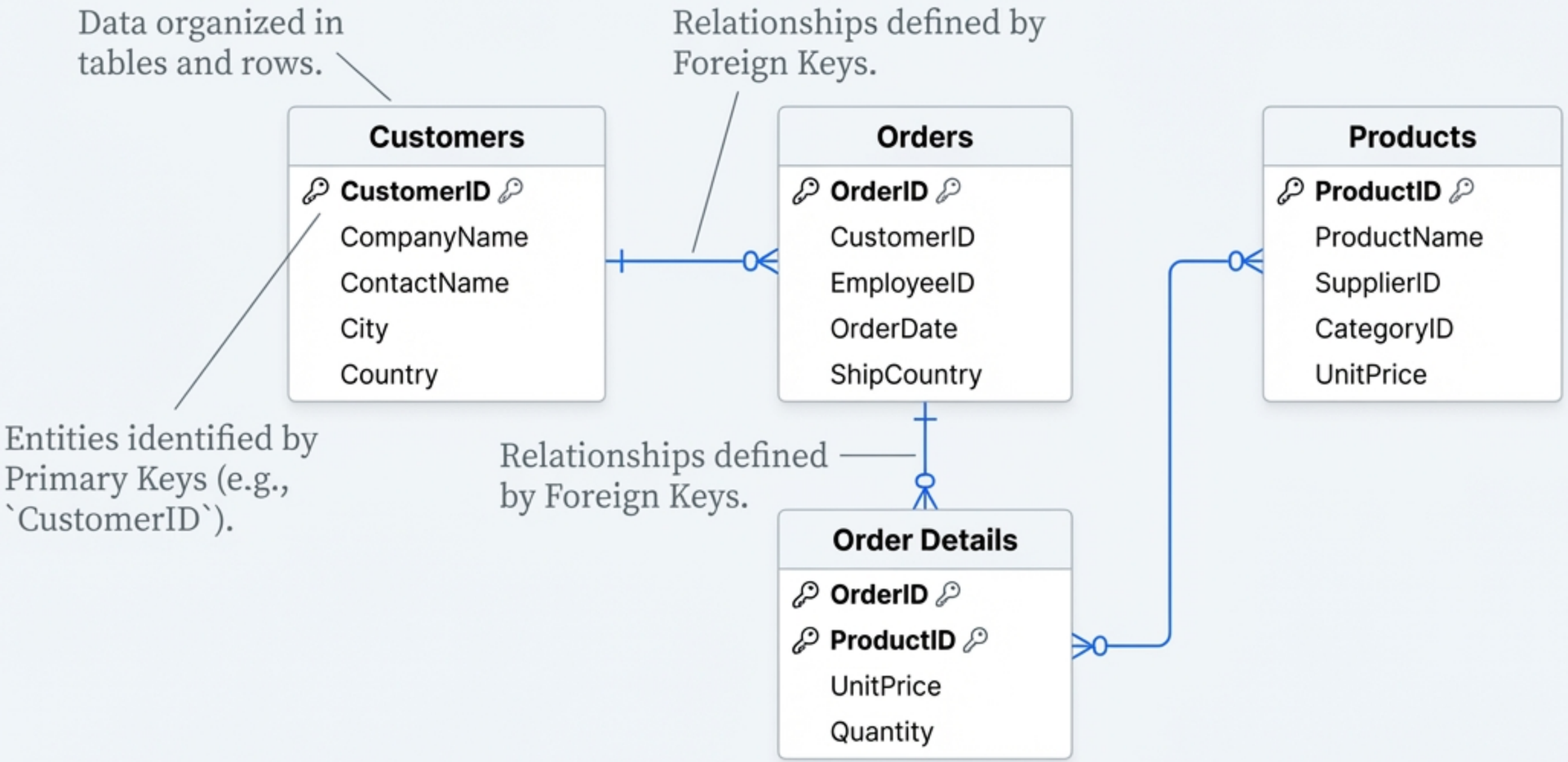
2. The Objective: Generate a Web-Native Knowledge Graph

Create a semantically rich, hyperlink-based graph layer without migrating the original data. Adhere to Linked Data Principles.

3. The Tool: The Virtuoso Support Agent

A specialized AI agent, powered by the OpenLink AI Layer (OPAL), designed to augment the skills of data professionals in generating and deploying knowledge graphs.

The Starting Point: A Classic Relational Schema



This structure is robust for transactions but limited in its ability to express complex, traversable relationships.

The Enabler: An AI-Powered Collaboration

The transformation is driven by the **Virtuoso Support Agent**, an AI agent that augments human expertise. It operates within the **OpenLink AI Layer (OPAL)**, a platform for creating and deploying specialized agents.

Key Detail Box

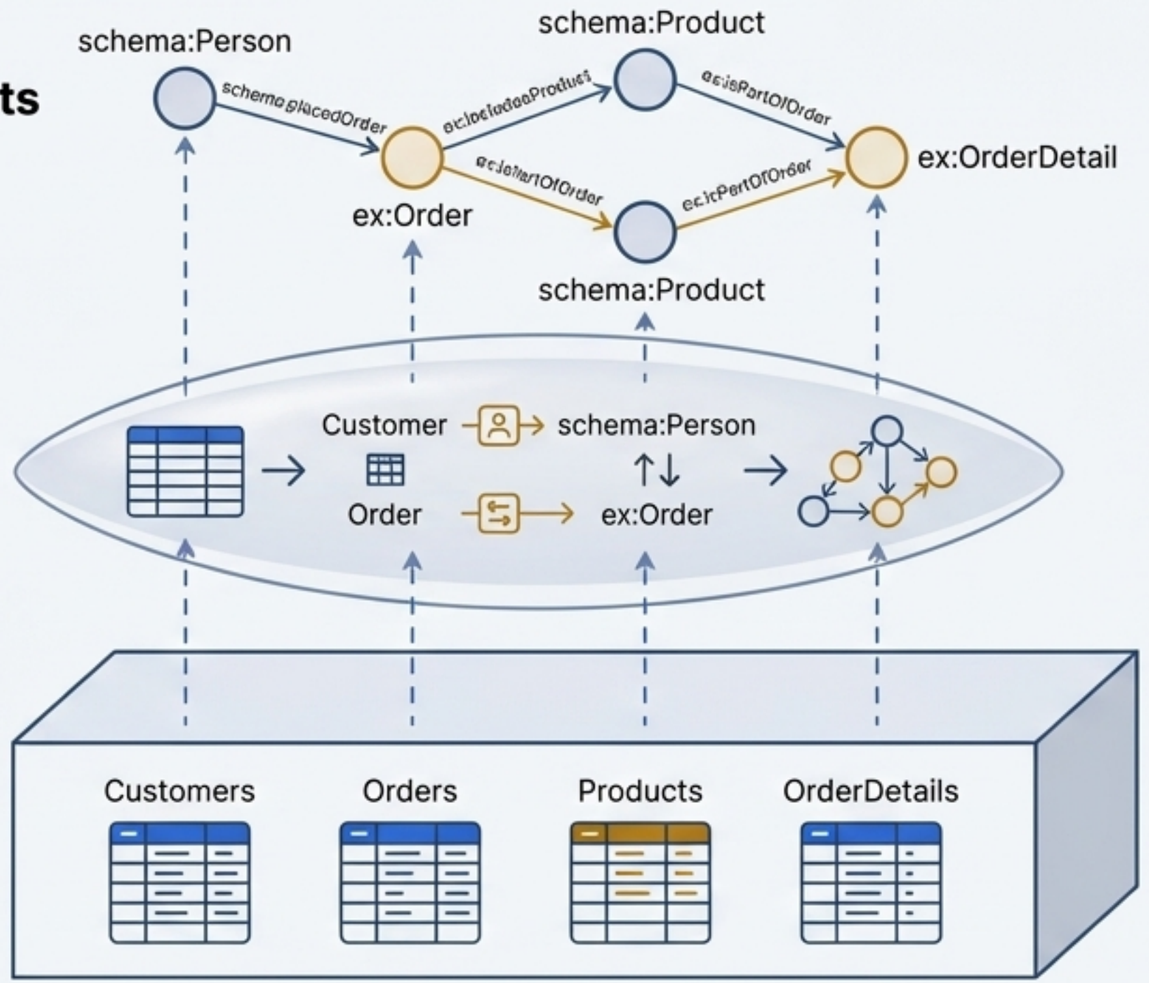
- **Interaction Method:** An OPAL session initiated via a Claude Skill.
- **Auditability:** All interactions are captured in a transparent OPAL session log for monitoring and auditing.
- **Function:** The agent's purpose is to generate, manage, and deploy knowledge graphs from existing systems-of-record.

The Bridge: Generating RDF Views, Not Migrating Data

Knowledge Graph Concepts
New, Queryable Graph Model

Virtuoso RDF View Layer
A Semantic Layer that
Maps SQL to RDF

Source SQL Tables
Original Data Stays in Place



Virtuoso generates a logical, semantic graph layer directly on top of the existing relational tables. The SQL data is not duplicated or moved.
This 'RDF View' is a real-time, virtual representation of the data as a graph.

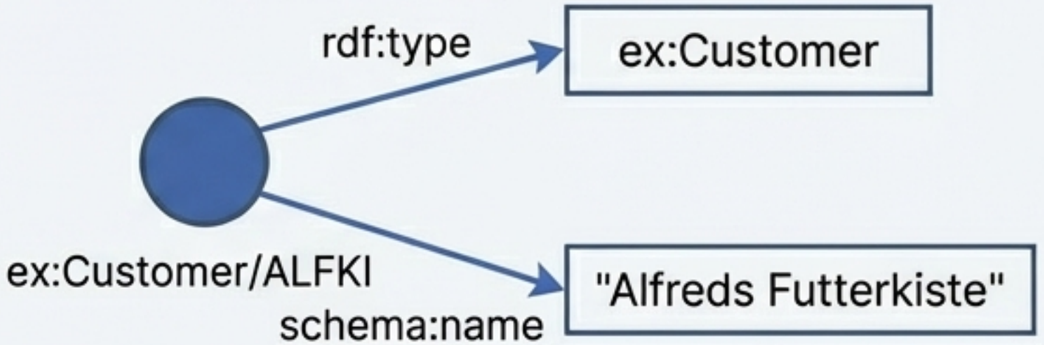
Remodeling the Data: From Tables and Rows to Entities and Properties

FROM: Relational Model

CustomerID	CompanyName
ALFKI	Alfreds Futterkiste

- Table -> `Customers`
- Row -> A specific customer record.
- Column -> `CompanyName`

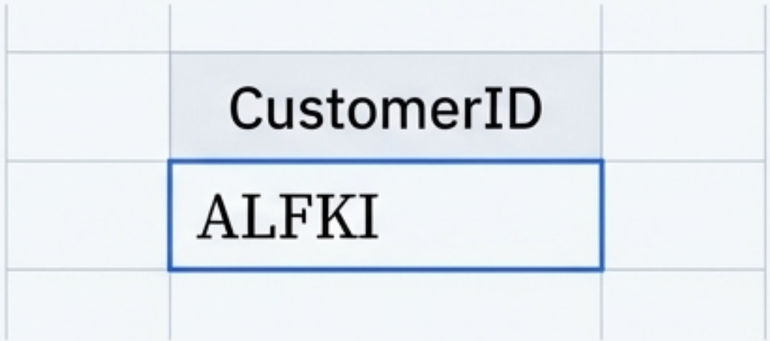
TO: Graph Model (RDF)



- Table becomes -> `rdf:type` (e.g., `ex:Customer`)
- Row becomes -> An Entity (a unique node in the graph)
- Column becomes -> A Property (e.g., `` `schema:name` ``)

The Power of Identity: From Primary Keys to Hyperlinks

SQL Identifier



A local, database-specific identifier. Meaning is confined to the table's context.

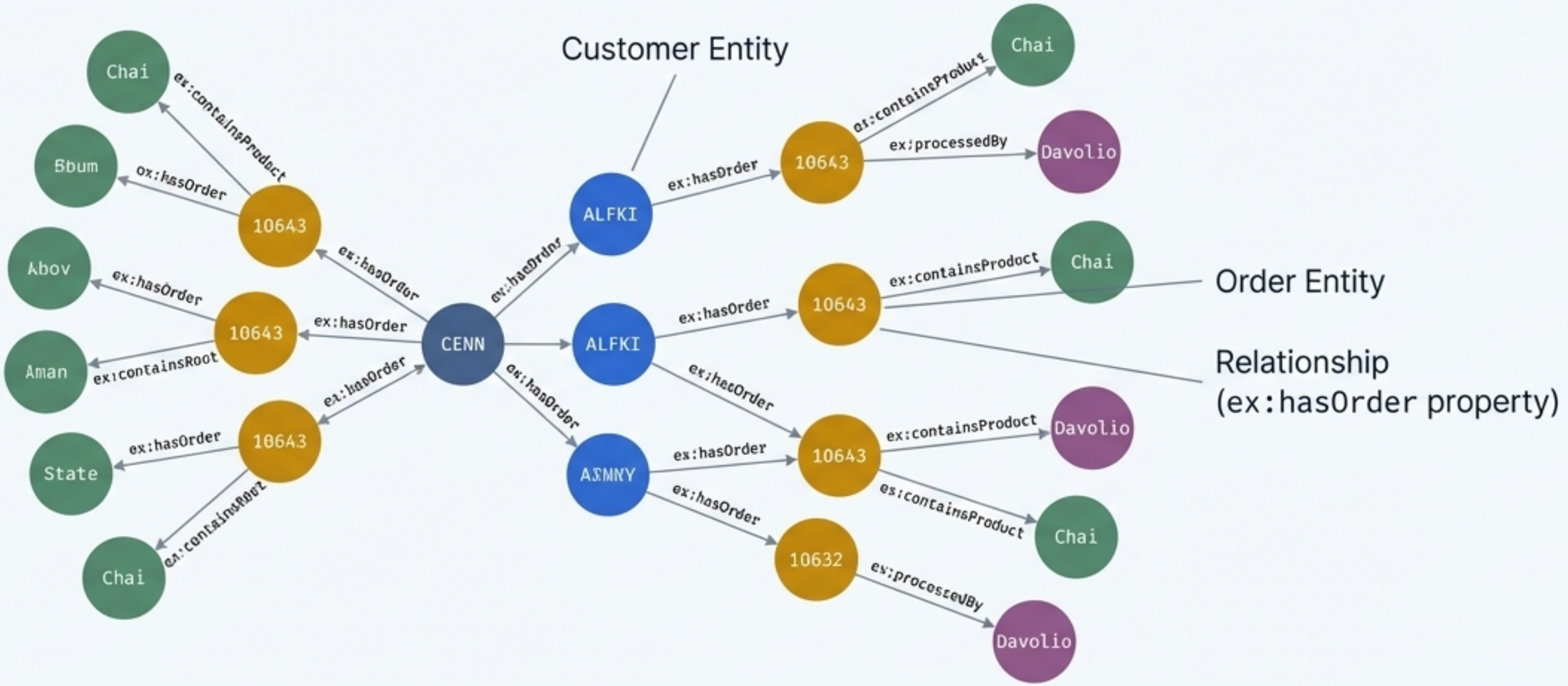
Linked Data Identifier (URI)

<http://{your-server}/Northwind/instance/Customers/ALFKI>

A globally unique, web-native identifier. Resolvable, self-describing, and linkable. This is the foundation of a true Knowledge Graph.

Key Takeaway: This implements the core **Linked Data Principle**: using HTTP URIs as names for things, allowing anyone (or any machine) to look them up.

The Result: A Deployed, Live, and Queryable Knowledge Graph



The original relational data is now accessible as an interconnected web of entities, ready for complex traversal and discovery queries.

Unlocking New Capabilities: SQL vs. SPARQL

The SQL Approach

Querying for Tabular Results

```
-- Example: Get orders for a specific customer
SELECT o.OrderID, o.OrderDate, p.ProductName
FROM Orders o
JOIN "Order Details" od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID
WHERE o.CustomerID = 'ALFKI';
```

Effective for structured, predictable joins. Returns a flat, tabular result set.

The SPARQL Advantage

Querying for Relationships and Paths

```
-- Example: Find all products ordered by a customer
-- and the employees who handled those orders.
PREFIX nw: <http://.../Northwind/ontology#>
SELECT ?order ?productName ?employeeName
WHERE {
  <.../Customers/ALFKI> nw:hasOrder ?order .
  ?order nw:hasDetail [ nw:orderedProduct ?product ] ;
         nw:handledBy ?employee .
  ?product rdfs:label ?productName .
  ?employee rdfs:label ?employeeName .
}
```

Designed to traverse relationships. Can discover complex connections and paths not easily expressed in SQL.

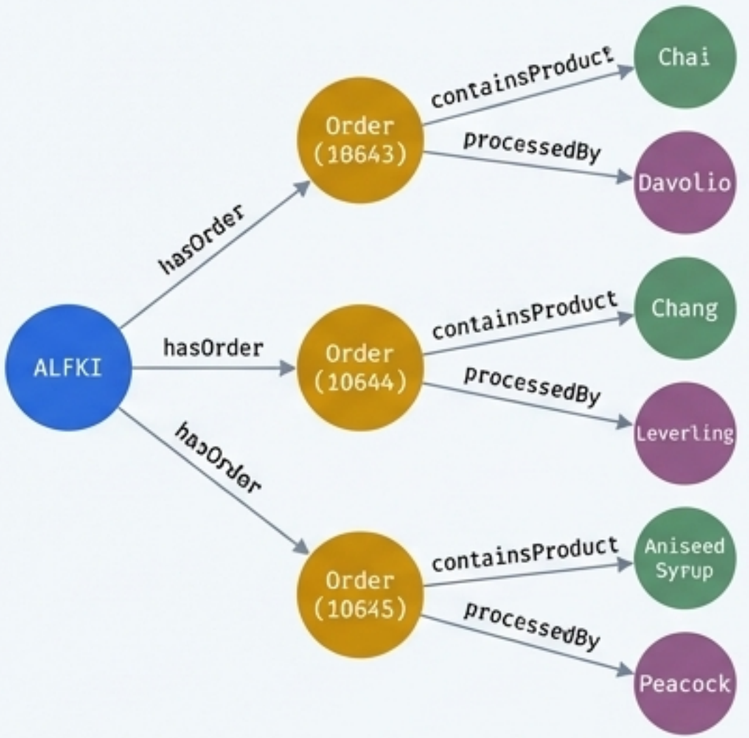
A Tale of Two Results: Tables vs. Connected Subgraphs

SQL Output: A Table

OrderID	OrderDate	ProductName
10643	2023-01-15	Chai
10644	2023-01-16	Chang
10645	2023-01-18	Aniseed Syrup

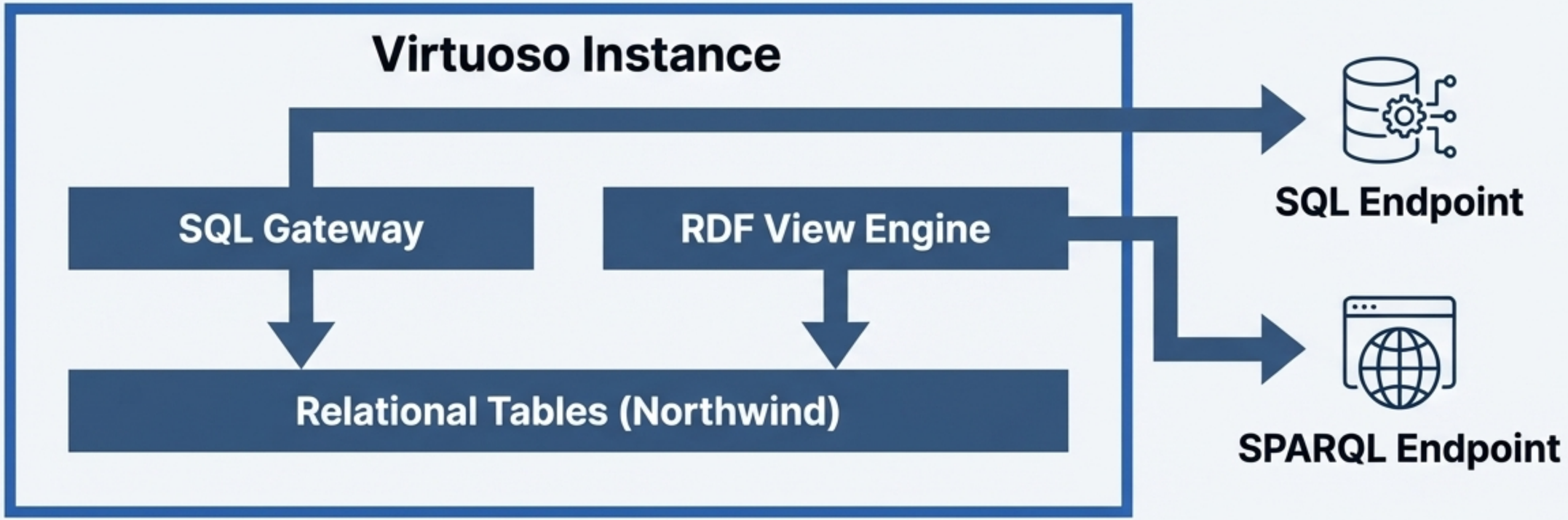
A precise, structured view of a specific slice of data.

SPARQL Output: A Set of Connections



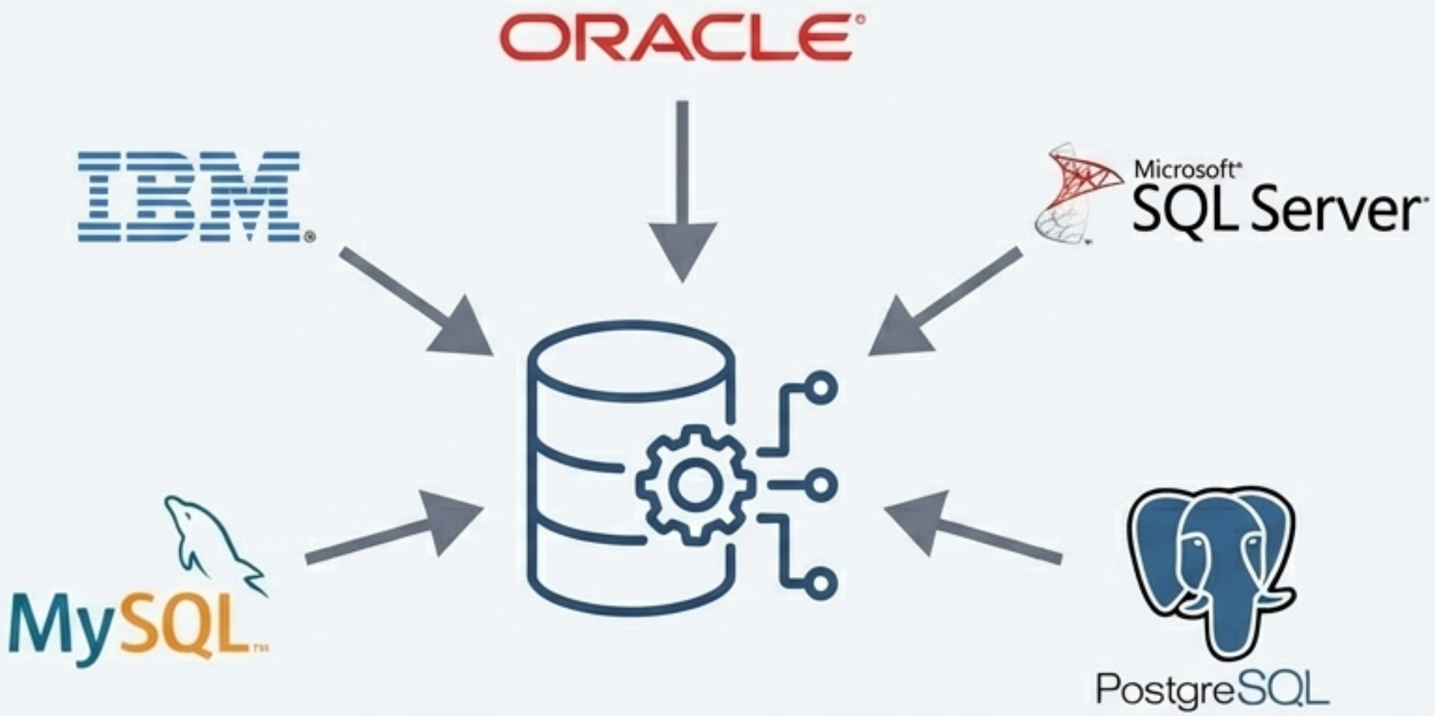
A view of the relationships *between* entities, revealing a network of connections.

The End-to-End Architecture: A Unified Platform



The Knowledge Graph, SQL tables, and hyperlink-based entity names are all deployed and served from a single, live Virtuoso instance. No data silos, no complex ETL pipelines.

Beyond Northwind: A Universal Approach for Relational Data



The Knowledge Graph generation capability applies to any RDBMS equipped with a standard ODBC or JDBC driver.

This is a repeatable, scalable pattern for modernizing existing data infrastructure without disruption.

The New Paradigm: Data as a Live, Interconnected Web

1. Augment, Don’t Replace

Modernize legacy SQL assets by overlaying a semantic graph layer. The original data and applications remain untouched.

2. Unlock Latent Value

Expose deep relationships and enable complex traversal queries impossible with standard SQL.

3. AI-Accelerated Process

Utilize specialized AI agents like the Virtuoso Support Agent to dramatically speed up the complex task of semantic modeling and deployment.

4. Create Web-Native Assets

Transform your data from a siloed resource into a network of addressable, linkable entities, just like the World Wide Web.