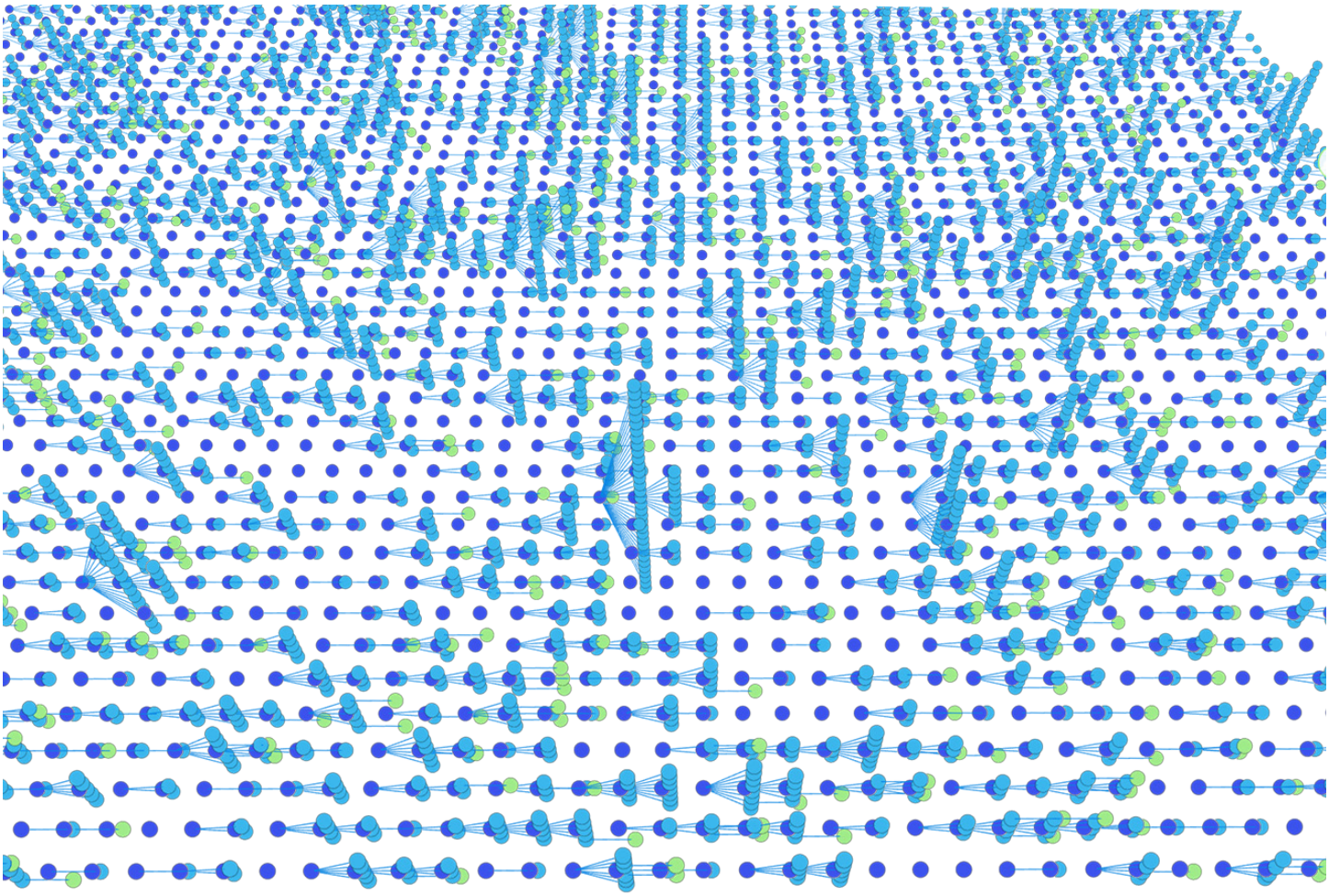
KINEVIZ

WHITE PAPER

Graph Visualization for Enterprise Data Products



Contents

Overview

The ability to access and interpret patterns and trends in data from disparate sources has repeatedly proven to provide a competitive advantage to enterprises of various kinds. In wide-ranging public and private domains, such as public safety, banking, manufacturing, and marketing, supporting timely yet secure access to data and to its flexible interpretation and re-use is recognized to be of paramount importance.

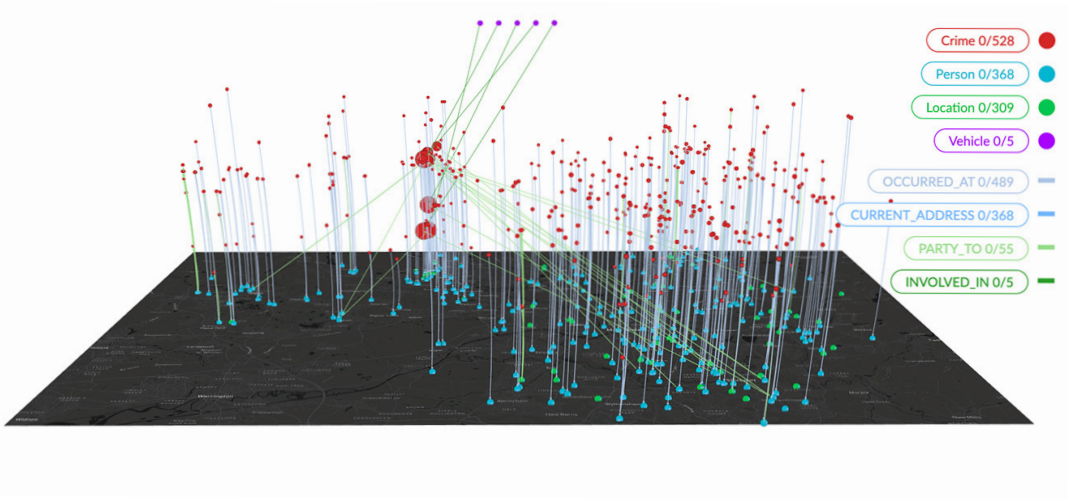
Graph data technology, which explicitly encodes data as entities (nodes) and their relationships (edges), has become a force-multiplier in this arena. Visualizing and analyzing graph networks is an increasingly valuable feature of emerging knowledge base, data fabric, and data mesh frameworks.

In this paper, we review how visualization takes a central role in the iterative processes of data discovery, exploration, modeling, analytics, and, ultimately, delivery of enterprise data products.

Graphs for Visualizing Patterns

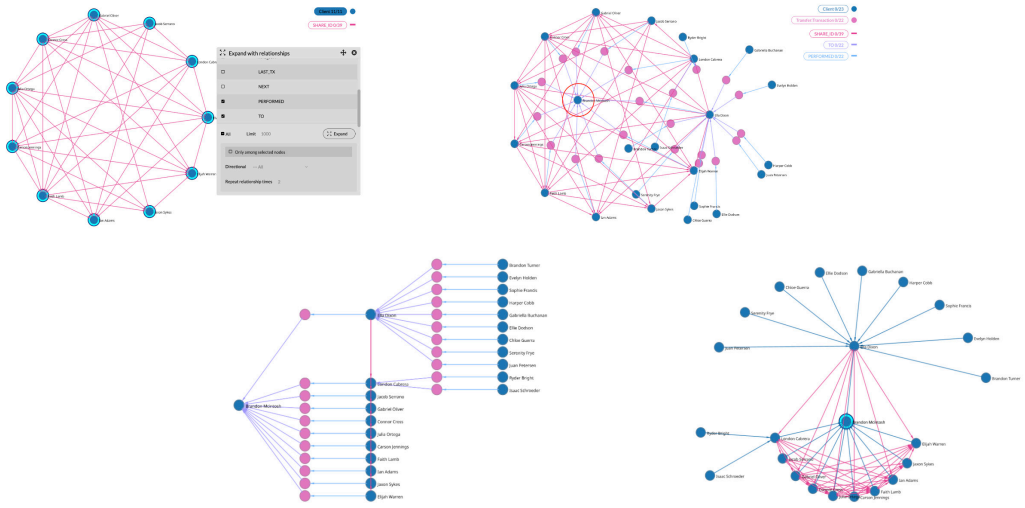
Graph data technologies are central to exploring patterns and visually highlighting their meaning. Encoding data as nodes connected by edges makes it possible to visualize connections in data in a way that humans naturally think about them. A connected graph pattern can immediately show the strength and reach of networks of many different kinds.

For example, the following map of crime incidence shows various types of connections between crimes and the people involved.



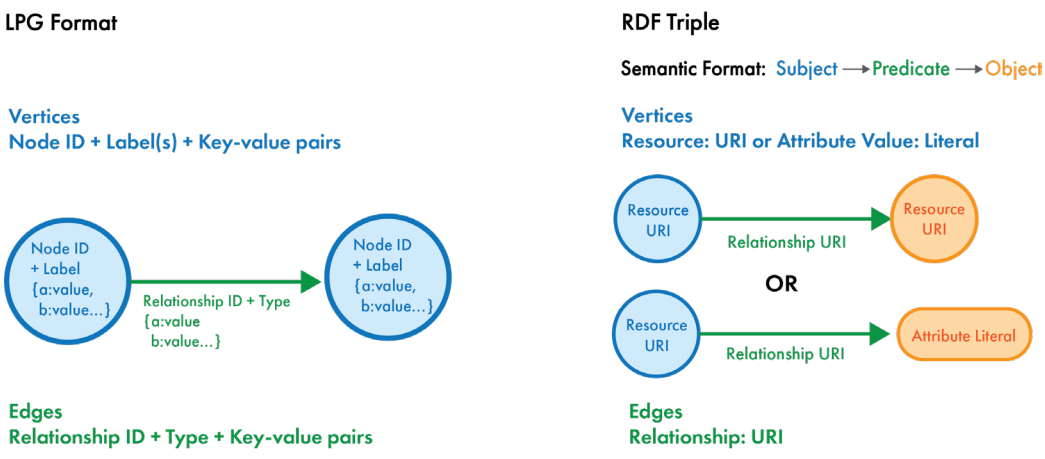
When expressed as nodes connected by edges of specific kinds, the network can be explored. Using graph analytics, patterns can be explicitly measured in terms of path lengths, degree of connection, clusters, and other measurable graph characteristics.

For example, in the following graph of fraud investigation data, connecting transactions with their owners, then tracing owners within three degrees of separation quickly clarifies relationships between those involved in a fraud ring.



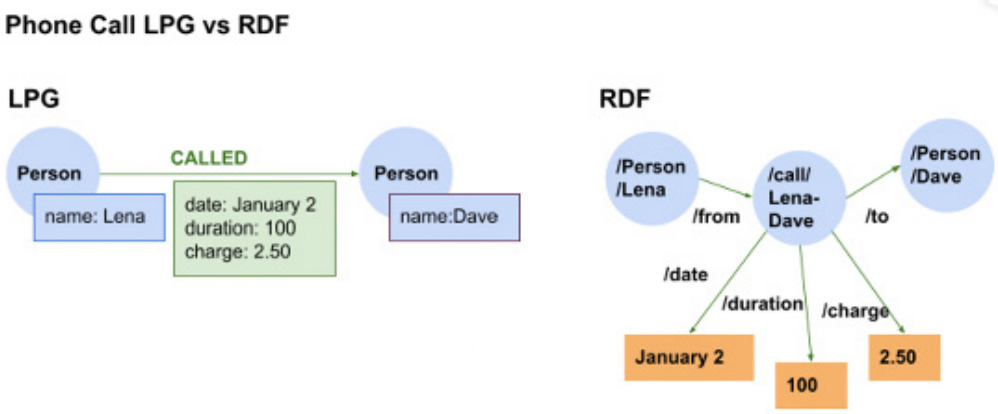
Graph Data Formats

Graph data can be expressed in two main formats: as a labeled property graph or the W3C standard RDF exchange format. These are accessed using different modeling and query processes.



- Labeled property graph (LPG) data is encoded as nodes or edges. Each will have an ID and at least one label or type. A node or edge can also include associated properties. This results in graphs that are relatively easy to query, work with, and visualize. Property graph database formats are not standardized, though they do usually provide ways to share data through open-source query languages and import/export functions.

- RDF data is encoded in the highly standardized interoperable Resource Description Framework (RDF) exchange format. Unlike LPG, every fact is explicitly visible in the graph. Meaning of the data is encoded in a semantic model that is separate from the data itself. This makes it possible to re-define and re-use data at will. And because the structure of the data is completely predictable, the resulting graphs can be easily and rapidly evaluated by automated systems. However, compared to LPG graphs RDF patterns are visually confusing to humans, even for a very simple graph such as the following telephone call example.



Interoperability of data is highly desirable, because it makes it much easier to automate and decentralize the process of serving data to applications. To achieve this, one can either express all the data in a single format and framework (the RDF/Semantic Web path), or build data pipelines that re-format data tailored to the needs of a given application.

Full interoperability is very costly for both implementation and management. And in the real world, knowledge always evolves. Even for successful RDF deployment, translation to an LPG format can ease and speed up visualization and exploration.

Discovery through iteration

The workflow and processes involved in exploratory data analytics – graph-enabled or not – are inherently exploratory and highly iterative. Initial engagement with data is likely to suggest additional and unexpected avenues of investigation. This means that discovery must be both reproducible and flexible. Questions focused on change over time require that we deliver data of similar provenance and quality repeatedly. New questions will likely require access to entirely new data sources and new data models. Domain experts must develop an understanding of where useful data resides, what it looks like, and how to focus it on delivering actionable results. The challenge is to collect, visualize, interpret, package, and manage data, both to communicate insights over time and to enable potential re-use.

A key advantage of graph data models is that they are both flexible and extensible. Once data is imported to a graph environment, further validation, transformation, inspection, sharing, and export can occur. New entities and connections can be added without the need to rebuild or re-design a model. Data can be transformed, both to create explicit graph patterns from implied connections and to reduce and simplify patterns.

For example, tabular data in CSV format can be imported directly into a Neo4j graph database (and most other graph databases), where it can be modeled, validated, and transformed in the process. Graph applications such as GraphXR and Neo4j Bloom enable many of the same processes. For example, the mapping editor feature of GraphXR lets you model any column of a CSV as an entity or relationship, or as a property of either. Once imported, the graph can be transformed by extracting new categories and relationships, or by merging, linking, or aggregating existing ones.

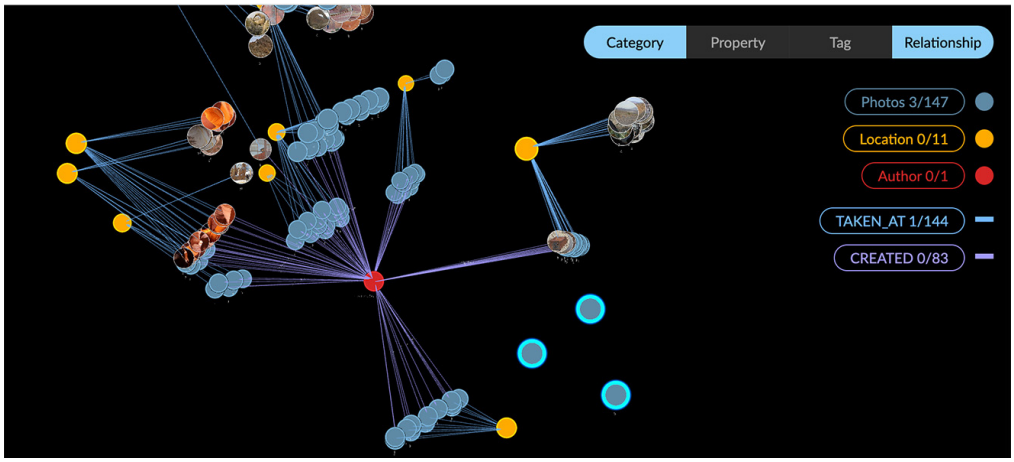
3D visualization for rapid insight

Visualization of patterns by human data analysts is a rapid and reliable way to discover value in data. This is true at almost every point along a discovery path, from viewing data imported from diverse sources to transforming its patterns and communicating actionable insights. Exploration in virtual 3D is especially effective for working with high-dimensional data.

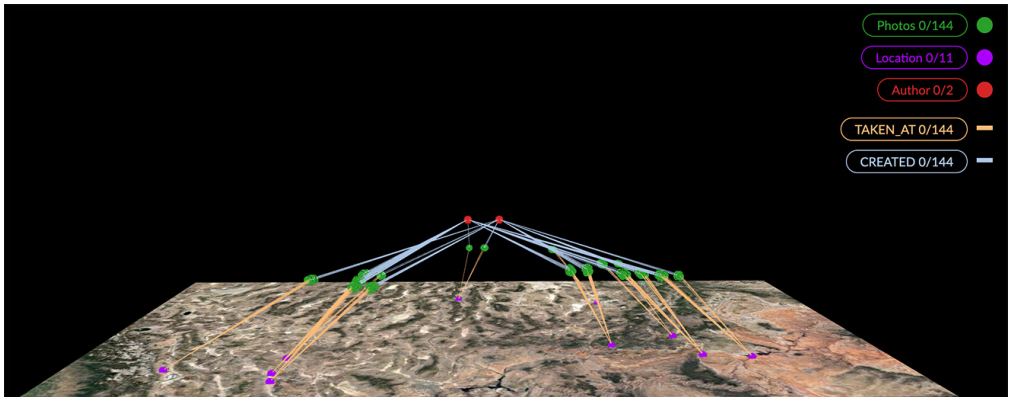
Although 2D visualization remains a preferred mode for communicating business intelligence, adding a virtual third dimension during discovery and exploration is a far more intuitive way to work with data. Enabling human domain experts to visualize patterns is often the most effective way to clarify and focus the stories the data can tell. This is because it encourages explainability—a pivot to the "how" and "why" in a pattern—rather than just presenting black-box metrics.

Even simply reorganizing the layout of data in a virtual space can bring pinpoint focus to exploration. As the following examples illustrate, one can quickly:

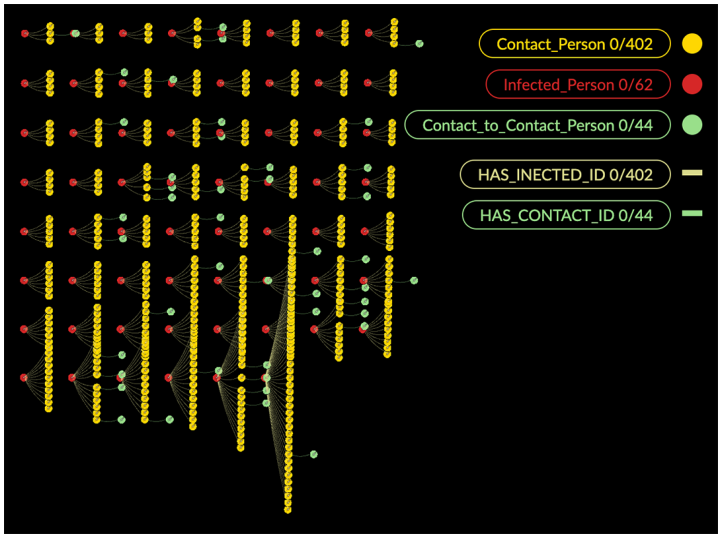
- Highlight outliers or unconnected data,



- Explore geo-located or time series properties,



- Visualize the results of graph analytic functions such as clusters or measures of connectedness.



Data sources for graphs

Enterprise data stores can be in many different formats: document or relational databases, flat tables, documents with standardized exchange formats, or in an unstructured format such as text. Efficient, rapid access to scattered data has traditionally been achieved by extracting data of interest, transforming it as needed, and loading it to one or more centralized warehouses. These repositories are almost always structured as indexed tables or hierarchical documents, to assist with query speed and efficient computation.

To manage rapidly exploding data volumes, there has been a wholesale adoption of cloud computing, which delivers robust and cost-effective infrastructure, platform, and computation services. Environments such as **ElasticSearch**, Google **BigQuery**, **Snowflake**, **Amazon Web Services (AWS)**, **Microsoft Azure**, and others provide data ecosystems in which data can be selected, stored, accessed through query (most often SQL) or other API services, then transformed through microservices. Dashboard software (for example, **Kibana**, **Tableau**, **AWS QuickSight**, and others) is increasingly available to provide both visualization and analytics, including capabilities to display data in terms of graph connections, timelines or geographic locations.

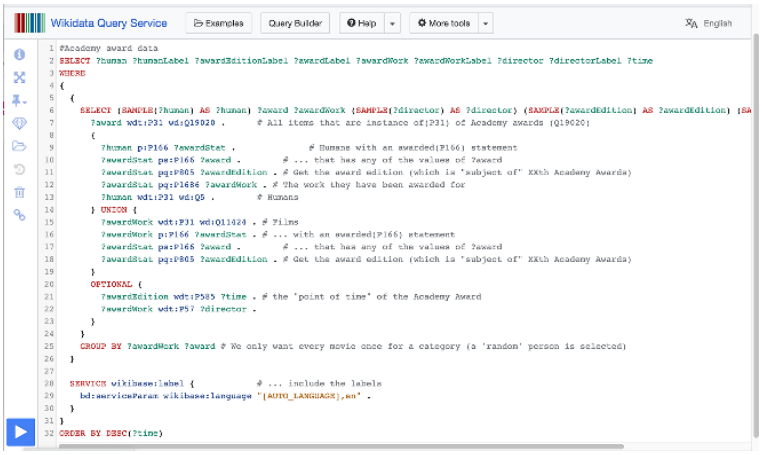
Data access methods

It has become relatively straightforward to access structured data from whatever source and model it as labeled property graph or RDF graph data. Access to data through API, query, or even drag-and-drop is typically conceived of as a one-way flow into a graph environment. That is, the structure of the source data is of interest only insofar as it affects the query process. Since flexibility and extensibility is central to graph technology, most (if not all) graph-enabled applications and databases include support for iterative data modeling. This means that a graph pattern can be modeled as part of the import process, then transformed further from there.

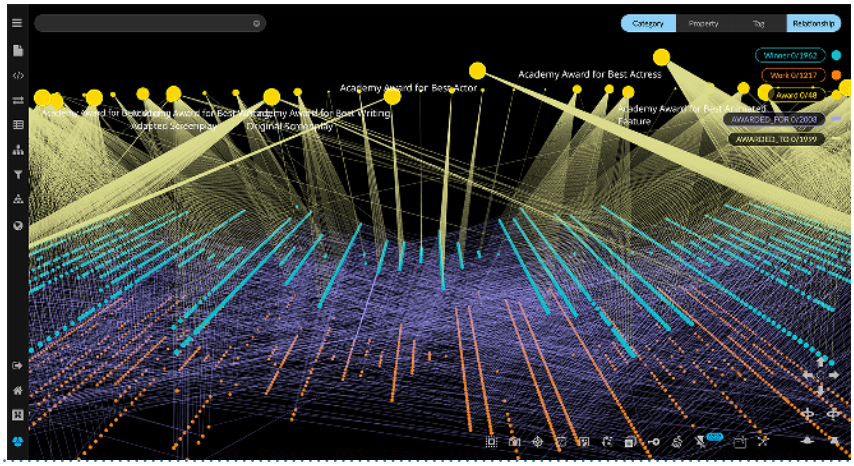
Drag and drop is a rapid way to explore and model data that can be exported as CSV files or JSON documents (from sources as simple as an Excel spreadsheet to relational database tables). A CSV imported as labeled property graph data results in the automatic transformation of table rows to entities (nodes) with properties defined by the column values. Graph databases such as **Neo4j**, **ArangoDB**, **Amazon Neptune**, and many others, and graph platforms such as **Perspectives**, **Ontotext**, and **TopQuadrant** also provide the capability to filter or validate tabular data or documents accessed in this way.

A **query** or **API** provides a more granular way to specify the data to be imported from a given source. Databases or data warehouses can be accessed using a variety of query languages (e.g. SQL, Cypher, AQL, Gremlin, SPARQL, and others) that enable data selection and transformation as well as varying degrees of constraint validation and exception handling. The query language used will depend on the source format.

Open-source data stores like **Wikidata** and **DBpedia** typically provide query front-ends or API endpoints that can be used as is or extended as needed. RDF, the W3C standard document interchange format designed for fast access and re-use of graph-native data, has been widely adopted for these very large interoperable data stores. In one step, you can extract data you need, and model it as graph data. For example, access to Wikidata's RDF stores is provided through the **Wikidata Query Service**. It includes a basic SPARQL query interface, a selection of example queries, a simple no-code query builder, and the capability to export the returned data. One of its example projects provides a SPARQL query for data about the Academy Awards:



The above query returns tabulated data that can immediately be exported in CSV or JSON format and mapped into a graph environment.



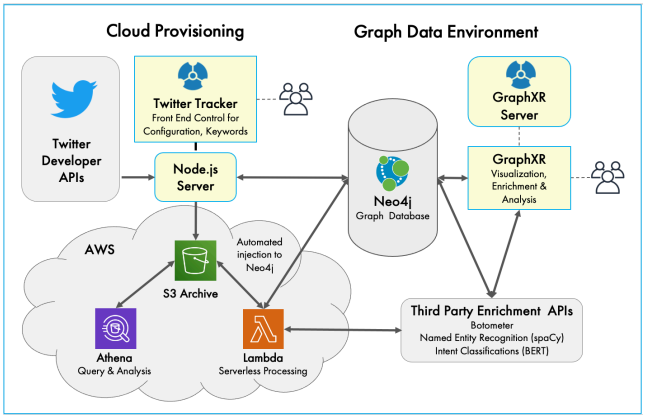
Data modeling platforms can help create reproducible access to diverse data sources. Software platforms such as TopQuadrant's **TopBraid EDG** provide full-featured RDF data modeling using **OWL and RDFS** languages. It's worth noting, though, that **in practice**, SPARQL queries combined with constraints provided through the **SHACL** shapes constraint language turn out to be both effective and easier to implement and manage. Within a controlled-access environment, APIs developed using **REST** or **GraphQL** become elements of an overall data management architecture to supply data to applications on request while complying with security and access constraints. This might involve accessing a data warehouse or data lake, or implementation within a formal integrated knowledge graph.

Data as a dynamic resource

Data sources are increasingly distributed rather than centralized. Useful data may be maintained in curated domains within an enterprise, or in open-source repositories. These data stores are far from static – new data types may be added frequently and the data may be refreshed almost continuously. Any useful knowledge management architecture must support accessing new data sources and seamlessly integrating new information and the new ways of looking at it.

There are many benefits of harnessing large data streams that can be fed into ML or AI automated processes. However, although it is definitely possible to replace human engagement, it is not always appropriate or effective. The trick is to automate large-scale data-driven processes where possible, and at the same time make exploratory data analysis as seamless as possible for human analysts and investigators.

For example, a Kineviz project to use streaming Twitter data for analysis of trends in political sentiment over the 2020 election cycle connected a variety of cloud provisioning services and machine learning applications with graph visualization.

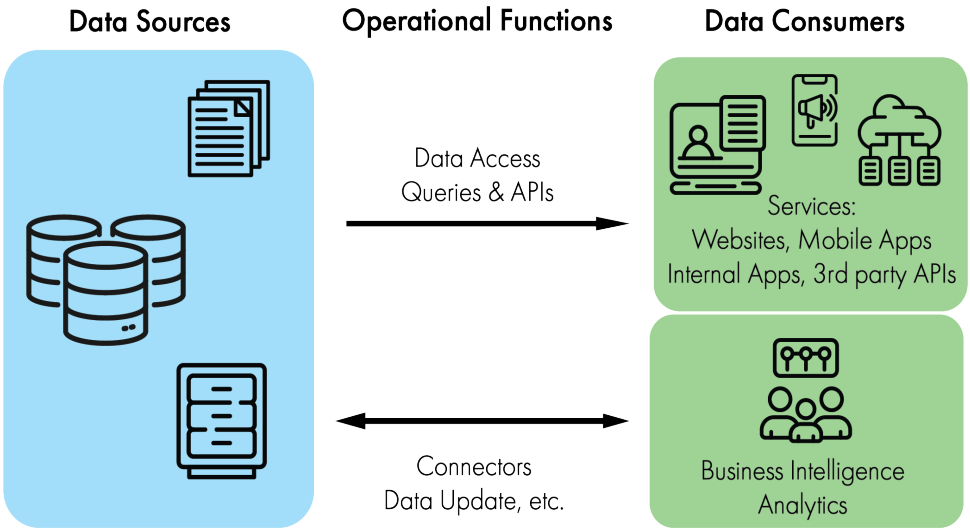


This architecture provided human observers with the near real-time visibility and analytic capabilities needed to investigate and explore shifts in the developing political situation.

Building self-service data products

Now let's take a brief look at how data management architectures are currently evolving to support the shift to data-driven applications.

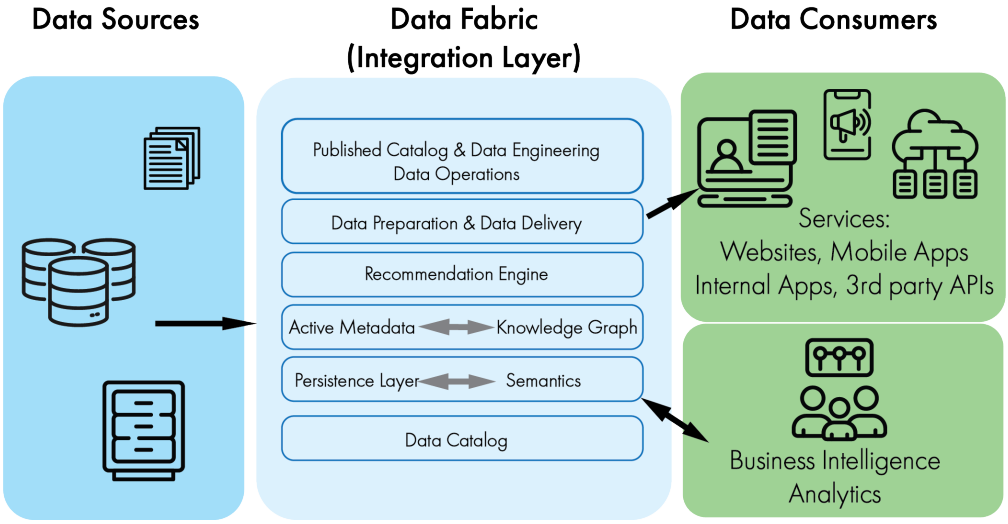
Typical legacy architecture involves data of many different types located in silos throughout an organization. Data is served to an application layer through APIs or queries, as noted in the figure below.



With this architecture, required data often resides in too many different formats and databases, which results in very slow access times. Even when collected in a single mainframe RDBMS, data lake, or data warehouse, queries may still be too slow and expensive. Furthermore, data is often inadequately defined and poorly protected.

Re-designing data architectures for self-service

To address an architecture that doesn't adequately support modern data-consuming applications, data can be accessed and stored through a well-defined operational layer or Data Fabric. A data fabric is built around centralized data warehouses, data lakes, or even knowledge graphs populated from available data sources. Maintaining source data models is not required, and typically not even allowed. Data that's saved would be sent to a new data store within the control of the data fabric. The result is a set of data sources for which extract, transform, and load (ETL) processes and access services for data-driven applications are delivered. All this can be implemented as on-premises infrastructure owned by the enterprise. Increasingly, though, it's set up through cloud computing services.



Data fabric is the number one **Gartner Technology Trend for 2022**. That's because using it has made it possible to recover nearly 70% of developer work in the data life cycle. Creating a centralized data fabric is still quite resource-intensive, but it has important advantages, particularly in supplying large amounts of clean data for machine learning or AI applications.

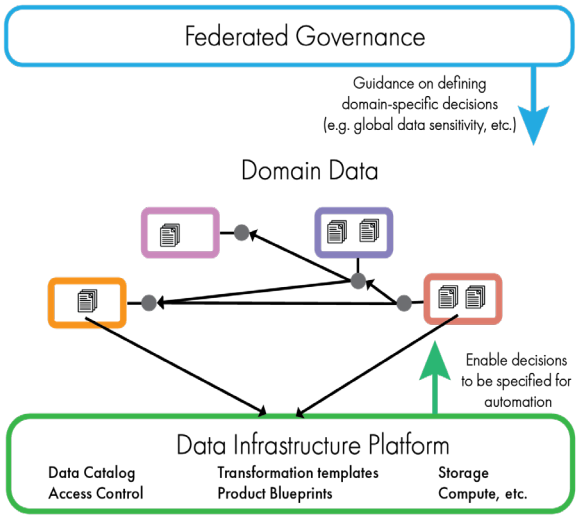
As data stores multiply in number and size, they present serious challenges in terms of interoperability, performance, and quality. To help with these issues, a knowledge base or knowledge graph architecture can be put in place. This consists of a formal collection of data and the data dictionaries, taxonomies, and semantic models that may be applied for various well-defined purposes. A knowledge graph includes full interoperability and strong governance structures, also expressed as graph data. Efficient tools and platforms to build out and extend knowledge graph architectures now exist and have been successfully deployed by many enterprises.

However, a commitment to interoperability across the enterprise is required. This can be difficult to achieve all at once, especially as new requirements continually emerge as the engagement with the data deepens. Those with experience building knowledge graphs advise starting small: building non-disruptive technology that sits beside existing systems, building just what's needed to show a particular successful outcome, and then planning to iterate and extend the system For in-depth discussion of these experiences see **The Rise of the Knowledge Graph (2021)** and **Knowledge Graphs (2021)**.

Factors such as insufficient staffing, or unfamiliarity with specific knowledge subdomains can derail the responsiveness of knowledge graphs within a data fabric. Data management headaches such as silos, duplication, and synchronization issues can persist. When that happens, users will find ways to do end-runs around a system that can't (or doesn't yet) meet their needs.

To address these issues, a distributed **Data Mesh** platform architecture has emerged, and is now gaining in popularity. It was first introduced by **Zhamak Dehghani** in 2019, who defined it as:

> ***"... an intentionally designed distributed data architecture, under centralized governance and standardization for interoperability, enabled by a shared and harmonized self-serve data infrastructure."***



The distributed model attempts to pre-empt points of failure that are repeatedly experienced with centralized architectures. Global governance is still put in place to support the overall experience of the enterprise's data consumers, but data remains in independent domains that can be accessed through a shared data infrastructure platform. The design is intended to promote domain data ownership and data as a product, with the additional goal of creating a self-serve data platform that includes federated data governance. A data mesh does support rapid iteration of exploration, analysis, and insight because it encourages domain experts and cross-functional teams to take the lead in creating data products that can be shared and re-used in specific, controlled ways. This has already lead to profound creativity and collaborative problem solving. The current trend is summed up in a recent **blog post**:

> ***"Every data-first company strives to or is already in the process of adopting a self-service business intelligence model."***

However, it's important to recognize that there is usually a continuum from more ad hoc to more controlled data provisioning. In practice, a mix of architectures are often in place operating side-by-side, either by design, or as temporary structures as a new architecture is adopted. Ultimately, implementing a data mesh is seen as a **three-part journey** involving 1) a shift of mindset, 2) building the platform support, and 3) implementing federated governance.

Graph visualization in a self-service environment

In a self-service data ecosystem, exploratory data visualization becomes especially useful. Domain-centered teams need to be able to explore and evaluate available data sources, build graph models according to their specific needs, and then federate data products for use across the enterprise. The data products being delivered must include not just the required metadata on provenance, quality and ownership, but ideally visual examples and interpretation as well.

Because visualizing patterns speeds up insight at every point in a data workflow, an infrastructure platform must include no-code or low-code collaborative software tools for data access, modeling, visualization, analysis, and communication. Teams focused around the interests of domain experts need their tools for visualizing data to be democratized, easy to understand, and seamless to use.

GraphXR stands out in this regard because it can pull data from almost any structured source, which can then be collaboratively accessed, modeled, explored, and transformed. If desired, data can be saved to a graph database or as JSON documents readable by many other applications. So, while a separate graph database repository can be used, it is not actually required. Integrated 2D and 3D visualization lets users find the best ways to work with data and communicate those insights to a larger team. Finally, an integrated notebook extension is available that provides support for reproducible data pipelines into the graph environment, along with dashboard-like visualization for exploration and analysis.

Ease of visualization will continue to be a focus of graph-aware products. Cloud computing platforms provide access to visualization dashboards as part of their data management offerings. Examples include **AWS QuickSight**, Microsoft **Data Explorer**, ElasticSearch **Kibana**, Google **Data Studio** or **BI Engine** for use with BigQuery, and many others. These provide prepackaged visualization options similar to what can be developed in Python, Observable, or CoLab notebooks: 2-dimensional charts (line, pie, and histogram), geographic mapping, and 2D representation of graph data.

Graph databases such as Neo4j, TigerGraph, NebulaGraph, MemGraph, ArangoDB, and many others include at least 2D visualization as part of their basic tool sets. Graph database purveyors are racing to offer large-scale analytic and exploratory connectivity, to respond to the continued interest in integrated, large-scale knowledge graph architectures.

Conclusion

Graph data, with its emphasis on patterns and connections, enables rapid visualization of high-dimensional data in terms of how humans naturally think. Graph data is relatively easy to build from data stored in other structured forms, meaning that many patterns implied in those data can now be investigated and utilized.

With the explosion in the amount of data available, the need for agile data-driven insight has led to the adoption of data management architectures that empower domain experts to contribute data products to the enterprise. A key challenge is to build an organization-wide architecture that can respond rapidly not just to changing data but also to the changing questions that arise during iterative exploration and analysis. This raises the need for full-featured, easy-to-use visualization and analytic tools to engage with the self-service model of data discovery, exploration, analysis and communication of results.

References and Further Reading:

Knowledge Graph and Data Architectures:

Barrassa, J., Hodler, A., and Webber, J. Knowledge Graphs. Data in Context for Responsive Businesses. July 2021.

Dehghani, Zhamak. 20 May, 2019. How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh.

Dehghani, Zhamak. March, 2022. Data Mesh. Delivering Data-driven Value at Scale.

Gartner Group. Top Strategic Technology Trends for 2022.

Martin, S., Skekely, B., and Allemang, D. March 2021. The Rise of the Knowledge Graph. Toward Modern Data integration and the data Fabric architecture.

Mehta, Yash. Jan 4, 2022. Business leaders Must Make Data Fabrics a Priority in 2022.

Query / Modeling languages:

Arango DB AQL

Cypher, CSV to graph via Cypher

SPARQL, SPARQL Basics

SHACL, Why I use SHACL for defining ontology models

TigerGraph GSQL

OWL / RDFS, OWL and RDF, Why I don't Use OWL any more

Data Interchange and Standardization

Gavin Mendel Gleason. The Semantic Web is Dead - Long Live the Semantic Web!, Github Blog Post, August 2022.

Web Standardization for Graph Data: 2019 Working Group Announcement.

Weinberger, D. August 10, 2016. SOLID (social Linked data) and the Interplanetary File System (IPFS)

API/ Access Platforms

Phil Sturgeon. 23 Jan 2017. GraphQL vs REST: Overview

AWS Athena / QuickSight

MS Azure / Data Explorer

ElasticSearch / Kibana / Elastic Stack (Elasticsearch, Kibana, Beats, Logstash)

Google BigQuery - SQL data warehouse / BI Data Studio

Kineviz: How to visualize Google BigQuery on GraphXR