KFG Notes

What is a Distributed Knowledge Graph?

Joel Gustafson

Published on: Oct 15, 2020

Introduction

The overarching goal of the Underlay project is to *create a distributed public knowledge graph.*

What does that mean? Lots of graph databases are distributed! Distributing things is a classic scaling strategy; it's basically expected that a large-scale system like a knowledge graph would be physically distributed.

But "knowledge graph" means more than "graph database", and in our context, "distributed" means something more than physical decentralization.

The first section is an introduction to the term "knowledge graph" and its general connotations. The second explores different ways that knowledge graphs degrade with scale, and the last section outlines the Underlay's alternative approach to large-scale data systems.

Knowledge Graphs

Knowledge graph is still a contested term with no clear or precise definition. Hogan et al. give an excellent and comprehensive introduction in *Knowledge Graphs*, from which we will quote extensively. They say that although the term is applied to a wide variety of only roughly similar projects,

> ... underlying all such developments is the core idea of using graphs to represent data... The result is most often used in application scenarios that involve integrating, managing and extracting value from diverse sources of data at large scale ... we view a knowledge graph as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities

Structurally, knowledge graphs use graph-based data models that highlight entities and relationships. These entities and relationships may have varying degrees of structure on top of them:

> Graphs allow maintainers to postpone the definition of a schema, allowing the data – and its scope – to evolve in a more flexible manner than typically possible in a relational setting

Some of the difficulty in defining knowledge graphs can be traced to this general approach to schemas. "Schemas", if they exist, may be ad-hoc and unenforced, which often results in data that feels "half structured". This is not a sign of poor quality or problematic incompleteness, but rather a core distinguishing characteristic of knowledge graphs.

> Knowledge graphs are often assembled from numerous sources, and as a result, can be highly diverse in terms of structure and granularity. To address this diversity, representations of

> *schema*, *identity*, and *context* often play a key role, where a *schema* defines a high-level structure for the knowledge graph, *identity* denotes which nodes in the graph (or in external sources) refer to the same real-world entity, while *context* may indicate a specific setting in which some unit of knowledge is held true

Simply describing the technical structure of knowledge graphs misses a key aspect, which is *where the data comes from* and *how the database is maintained*. Knowledge graphs are graph databases *that come from* the integration of data from many sources and in many domains. In other words, you make a knowledge graph by pulling data about lots of different kinds of things together in one place. This is part of what might justify calling them *knowledge* graphs, since it involves parsing out latent common structure behind "mere data" as it exists already, and aligning that common structure in a single space.[1] A knowledge graph is a graph that has somehow unified or reconciled the entities and relationships shared between (previously) separate data sources. Obviously not all entities and relationships are shared, but intuitively we feel that many are: an employee in a company directory can be treated as the same class of thing as a city resident in a census. Within each original dataset, the two will have different set of properties, but an external human curator can reasonably infer that they refer to the same type of real-world entity.

The phrase "real world" shows up a lot when discussing knowledge graphs or semantic data, where it serves to ground the referents of references, and especially to ground when two things are "actually" the same. We will return to this point in a little while.

Lastly, there's a continuous, living, process-oriented aspect to knowledge graphs:

> ... effective methods for *extraction*, *enrichment*, *quality assessment*, and *refinement* are required for a knowledge graph to grow and improve over time

The *practice* of integrating, reconciling, and curating multi-domain data sources is the real thing that people are naming. The graph data itself is, like a mushroom, only a static artifact of a complex and evolving underlying system.

To summarize, a knowledge graph is not just data in a graph database; it's a loose collection of ongoing processes. It's easy to overlook this and only focus on the data because these processes are often implicit, ad-hoc, done by people, embedded in organization-specific workflows, and so on.

Degrading with scale

Given this broader framing, we might start to wonder how the process-oriented aspects scale as a knowledge graph tries to integrate data from more and more domains.

Aligning ontologies

We described integration as a sort of noble exercise: reconciling the entities and relationships shared between separate sources. But it is also inherently an exercise in compromise. Different sources inevitably represent the same things in different ways, but worse, the things they represent are never actually the "same", and only ever partially correspond to each other.

This isn't about schemas in the technical database sense. Two databases might have the exact same schema, but still represent slightly (or wildly) different things. The real thing that integration accomplishes is more like ontology alignment, an operation on abstract structures that just happens to be expressed as a mapping from one database into another. Typically we don't work with or even think about explicit ontologies - we just rely on natural-language property names and general context instead - but it's useful to imagine they exist because it helps explain why integration is so notoriously painful. The complexity and similarity of the actual schemas involved isn't even that relevant; what matters is how well the underlying concepts align.

To illustrate this, suppose that we have a general-purpose knowledge graph full of entities like people, places, and things. Let's say we also scrape GitHub to build a collaboration graph of GitHub accounts, and now we'd like to integrate this into our knowledge graph so that we can query for collaborator relationships between people. We're immediately faced with the distinction between people and user accounts, which are not quite the same thing. People might have several accounts - maybe different ones for personal and work projects - and any account might be used by more than one person.

There are two basic strategies that we can apply to align these ontologies.

One option is to just ignore the difference and merge the two classes based on email, full name, or whatever we have, and add the new "collaborator" relationships directly between people in the knowledge graph. This keeps the graph as simple as possible - we're just adding data and not reorganizing anything - but the sacrifice we make in doing so is that we distort the data we're integrating. People and user accounts are simply different kinds of things, and treating them as the same will give us extra or missing links due to the mismatch between the underlying ontologies. This effect could be called *dilution* or *compression* or *erosion*.

Our other option is to make separate entities for each user account, and then relate them to the associated people whenever we can. Maybe we'd even introduce a new "Agent" class, with both Person and User Account as subclasses, plus a "represents" relationship between Agents. Then maybe we add some inference rule about "if Agent A represents Agent B, Agent C represents Agent D, and Agent A and Agent C are collaborators, then Agent B and Agent D are collaborators too", so that we can query for collaboration relationships between people like we wanted. Here, we've been more faithful with our alignment, but only at the expense of accumulating complexity in the process.

In practice, we usually end up using a combination of the two when integrating new data, leaving us with a result that is (oddly enough) both slightly simplified and slightly more complex than the data that we started with. This trade-off is unavoidable; the real value proposition of knowledge graphs is in the balancing of these extremes. If we apply too many simplifying assumptions, we'll wind up with data that bears so little resemblance to its sources that it is effectively useless. On the other hand, if we accumulate too much complexity, we'll get a knowledge graph that is so convoluted that it becomes effectively unusable. The best we can do is shoot for a middle ground that's reasonably expressive and reasonably manageable.

You can sense this trade-off in the class hierarchy of any large ontology, like schema.org or Wikidata. They straddle the fine line between surreal, hopelessly abstract semantic soup and practical, best-effort compromises. Given an impossible question like "what kinds of actions are there", the answers are about as good as you could hope for.

- Thing
  - Action
    - AchieveAction
      - LoseAction
      - TieAction
      - WinAction
    - AssessAction
      - ChooseAction
        - VoteAction
      - IgnoreAction
      - ReactAction
        - AgreeAction
        - DisagreeAction
        - DislikeAction
        - EndorseAction

https://schema.org/docs/full.html

Ignoring ontologies

Ontologies are typically associated with formal class hierarchies, but the effect is the same for unstructured "schema-less" graph data as well. This is because even schema-less data models have entities that need to be reconciled and joined with new data, and this reconciliation process is a kind of alignment whether we like it or not.

This perspective is easy to miss because of the way that ontologies are implicitly encoded in schema-less graph data. When we imagine integrating new data into the graph, we usually only picture adding some new nodes and edges. That doesn't seem like it involves "changing the ontology", and it definitely doesn't feel lossy in any concerning way.

But every decision to reconcile two entities dilutes their specificity, usually without leaving any way to tell that the entity is now the rough composite of several things from different sources. Subtleties will inevitably get lost, like the distinction between a person and a user account, a website and its publisher, a president and their administration, and so on. As we say more and more kinds of things about an entity, we're left with fuzzier and fuzzier answers to the question "what does this entity actually represent?"

Put more generally, schema-less data models don't accumulate the visible complexity of a class hierarchy, they just ignore the complexity entirely. We're still left with a diluted ontology, we just don't

acknowledge it.

All models are wrong; some get worse and worse

This is probably a good place to step back and check in with the rest of the world. Problems related to "models distorting the underlying reality" are not unique to knowledge graphs or even computing - they're as old as civilization itself. *Seeing Like a State* by James C. Scott traces this tension throughout history: standardized units, first and last names, and many significantly more violent and brutal interventions are all the result of imposing "state simplifications" that increased outward legibility (often for good reason) at the expense of interfering with local practice (or, for our purposes, *local ontology*). Some things are different to different people:

> Telling a farmer only that he is leasing twenty acres of land is about as helpful as telling a scholar that he has bought six kilograms of books

And there are plenty of ontological differences that are even more thorny. Scott details how some communities didn't have our conception of land ownership at all, and instead operated their farms on amazingly complex shared-use policies tacitly developed and passed down over generations. When the state wanted to tax the land and asked who owned it, there simply wasn't an answer.

It's no surprise that we're still struggling with these. Marriages in databases, sex and gender in web forms, or even gerrymandering can be seen as an extension of the same fraught relationship that models induce between the modeled and the modellers. So what's the big deal? Why fret over ontology misalignment in knowledge graphs when we're already swimming in misaligned ontologies of all kinds?

Well, there are two related phenomena that *are* specific to knowledge graphs.

The first is that, like all data models, knowledge graphs provide value by balancing different types of sacrifices, but in knowledge graphs these sacrifices accumulate. In most applications the ontological loss is relatively fixed - you set the data model once, or refine it over time, but the scope is constant. But knowledge graphs mostly grow horizontally to encompass more *kinds* of things; the ontology is actually the *primary* scaling dimension.[2] As they grow, knowledge graphs get unavoidably worse and worse with respect to the degree that they represent or correspond to their sources; they degrade with scale. Again, this is easy to miss if all that we think of is just "adding nodes and edges".

The second is that the ontology we get from iterated compromise is, to abuse the journalism term, a "view from nowhere". Unlike data models in most applications, which are grounded with a specific purpose and oriented in a specific context, knowledge graphs tend to be open-ended, projecting anything and everything into one space. Again, this is in the essence of their value proposition, but it

means that the ontology that emerges is one that nobody actually believes in, for lack of a better phrase. Knowledge graphs are supposed to be general-purpose, but that means they are for *no-purpose*. The general purpose does not exist in the same way that the average man does not exist, and designing for and around it invites the same kind of unintended consequences. These unintended consequences can be benign or oppressive - weird quirks or systemic bias. And this is all especially sobering given that information is an increasingly relevant form of power, and that representation (ie ontology) is relatively subtle and difficult to challenge.

Knowledge graphs don't just get worse and worse: they get worse and worse *at being something that's already bad*. And the negatives here do not cancel out.

Decentralizing context

What's the common theme there? To avoid running the word "ontology" even further into the ground, let's call it *context*. Every piece of data is only meaningful relative to its context, which is usually only implicit. Integrating data into a knowledge graph has a destructive effect on its context, and the general-purpose context of the knowledge graph is not always particularly useful and has to the potential to be catastrophically bad.

But we'd still like to be able to do the things that knowledge graphs let us do. Lots of questions do end up spanning several contexts, and it'd be nice to have a unified way to answer them. In fact, we can even think of every question as having its own context, which usually doesn't correspond perfectly to contexts of any of the data that we have available. From this perspective, we see that the knowledge graph approach is to have all the questions *and* data rendezvous in a single global context, which suggests the natural alternative: decentralization.

We shouldn't be too glib about this - it sounds nice, but what does it really mean? A first-level breakdown involves investing in two new kinds of things:

1. Explicit representations of a dataset's context. Where did it come from? What is it supposed to apply to? When is it supposed to apply?
2. Explicit representations of *qualified* relationships between datasets that preserve that context.

Neither of these have to even be machine-readable to work! Knowledge graphs are human processes and will remain so until we have a binary serialization for meaning. So in practice this actually begins with extremely human-centric tools for data curation - tools for creating a schema for the user's specific domain, but that also encourage documenting the provenance of the data, in a format as structured as appropriate. The goal is to create a system that entirely obviates the need for the centralized general-purpose knowledge graphs that we have now, by connecting the contexts of

questions directly to the contexts of the answers. We choose to still call this a knowledge graph, but it's more akin to the World Wide Web than AOL.

Does this idea have any real technical foundation? Why should we expect this to even be possible? Well, maybe we shouldn't *expect* it to be possible, but we certainly shouldn't assume that we've even scratched the surface of understanding *how to work with data well*. Computing is still in its infancy; we barely know what we're doing as it stands.

And there is some theory that we can try to apply. Category theory in particular is rich with many of these themes, like qualified relationships and compositional mappings. And sure enough, lots of applied category theory work revolves around databases, schemas, integration, ontologies, and knowledge representation - sometimes very abstract, sometimes shockingly practical. Things are only theoretical until they're not.

Humans are so adept at context-switching that we give ourselves the illusion of having a single big ontology. Our goal is to build a large-scale data system that is so adept at context-switching that it gives the illusion of being a knowledge graph.

Footnotes

1. What deserves to be called "knowledge" and whether knowledge graphs as we know them qualify is a much larger debate than we're equipped to tackle. ↩

2. This is a little exaggerated - it would probably be more accurate to say that knowledge graphs grow *diagonally*. ↩