What’s the Difference Between an Ontology and a Knowledge Graph?

Bess Schrader

As semantic applications become increasingly hot topics in the industry, clients often come to EK asking about ontologies and knowledge graphs. Specifically, they want to know the differences between the two. Are ontologies and knowledge graphs the same thing? If not, how are they different? What is the relationship between the two?

In this blog, I'll walk you through both ontologies and knowledge graphs, describing how they’re different and how they work together to organize large amounts of data and information.

What is an ontology?

Ontologies are semantic data models that define the types of things that exist in our domain and the properties that can be used to describe them. Ontologies are generalized data models, meaning that they only model general types of things that share certain properties, but don’t include information about specific individuals in our domain. For example, instead of describing your dog, Spot, and all of his individual characteristics, an ontology should focus on the general concept of dogs, trying to capture characteristics that most/many dogs might have. Doing this allows us to reuse the ontology to describe additional dogs in the future.

There are three main components to an ontology, which are usually described as follows:

- **Classes**: the distinct types of things exist in our data (e.g. ….)
- **Relationships**: properties that connect two classes
- **Attributes**: properties that describe an individual class
For example, imagine we have the following information on books, authors, and publishers:

### Books

<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Publisher</th>
<th>Year Published</th>
<th>Followed By</th>
</tr>
</thead>
<tbody>
<tr>
<td>To Kill a Mockingbird</td>
<td>Harper Lee</td>
<td>J. B. Lippincott Company</td>
<td>1960</td>
<td>Go Set a Watchman</td>
</tr>
<tr>
<td>The Picture of Dorian Gray</td>
<td>Oscar Wilde</td>
<td>J. B. Lippincott &amp; Co.</td>
<td>1890</td>
<td></td>
</tr>
</tbody>
</table>

### Publishers

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. B. Lippincott &amp; Company</td>
<td>Philadelphia</td>
<td>United States</td>
</tr>
<tr>
<td>HarperCollins, LLC</td>
<td>New York City</td>
<td>United States</td>
</tr>
<tr>
<td>Heinemann</td>
<td>Portsmouth</td>
<td>United States</td>
</tr>
<tr>
<td>New American Library</td>
<td>New York City</td>
<td>United States</td>
</tr>
<tr>
<td>Hutchinson</td>
<td>London</td>
<td>United Kingdom</td>
</tr>
</tbody>
</table>

### Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Country of Birth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harper Lee</td>
<td>United States</td>
</tr>
<tr>
<td>Oscar Wilde</td>
<td>Ireland</td>
</tr>
<tr>
<td>Arthur C. Clarke</td>
<td>United Kingdom</td>
</tr>
</tbody>
</table>
First, we want to identify our classes (the unique types of things that are in the data). This sample data appears to capture information about **books**, so that's a good candidate for a class. Specifically, the sample data captures certain types of things about books, such as **authors** and **publishers**. Digging a little deeper, we can see our data also captures information about **publishers** and **authors**, such as their locations. This leaves us with four classes for this example:

- Books
- Authors
- Publishers
- Locations

Next, we need to identify relationships and attributes (for simplicity, we can consider both relationships and attributes as properties). Using the classes that we identified above, we can look at the data and start to list all of the properties we see for each class. For example, looking at the **book** class, some properties might be:

- Books have authors
- Books have publishers
- Books are published on a date
- Books are followed by sequels (other books)

Some of these properties are **relationships** that connect two of our classes. For example, the property **“books have authors”** is a relationship that connects our **book** class and our **author** class. Other properties, such as **“books are published on a date,”** are attributes, describing only one class, instead of connecting two classes together.

It’s important to note that these properties **might** apply to any given book, but they don’t necessarily **have** to apply to every book. For example, many books don’t have sequels. That’s fine in our ontology because we just want to make sure we capture possible properties that could apply to many, but not necessarily all, books.

While the above list of properties is easy to read, it can be helpful to rewrite these properties to more clearly identify our classes and properties. For example, **“books have authors”** can be written as:

```
Book → has author → Author
```

Although there are many more properties that you could include, depending on your use case, for this blog, I’ve identified the following properties:

- Book → has author → Author
• Book → has publisher→ Publisher
• Book → published on → Publication date
• Book → is followed by → Book
• Author → works with → Publisher
• Publisher → located in → Location
• Location → located in → Location

Remember that our ontology is a general data model, meaning that we don’t want to include information about specific books in our ontology. Instead, we want to create a reusable framework we could use to describe additional books in the future.

When we combine our classes and relationships, we can view our ontology in a graph format:

What is a knowledge graph?

Using our ontology as a framework, we can add in real data about individual books, authors, publishers, and locations to create a knowledge graph. With the information in our tables above, as well as our ontology, we can create specific instances of each of our ontological relationships. For example, if we have the relationship Book → has author → Author in our ontology, an individual instance of this relationship looks like:
If we add in all of the individual information that we have about one of our books, *To Kill a Mockingbird*, we can start to see the beginnings of our knowledge graph:

If we do this with all of our data, we will eventually wind up with a graph that has our data encoded using our ontology. Using this knowledge graph, we can view our data as a web of relationships, instead of as separate tables, drawing new connections between data points that we would otherwise be unable to understand. Specifically, using SPARQL, we can query this data, using inferencing, letting our knowledge graph make connections for us that weren’t previously defined.
So how are ontologies and knowledge graphs different?

As you can see from the example above, a knowledge graph is created when you apply an ontology (our data model) to a dataset of individual data points (our book, author, and publisher data). In other words:

\[
\text{ontology + data = knowledge graph}
\]

Ready to get started? Check our ontology design and knowledge graph design best practices, and contact us if you need help beginning your journey with advanced semantic data models.

Enterprise Knowledge (EK) is a services firm that integrates Knowledge Management, Information Management, Information Technology, and Agile Approaches to deliver comprehensive solutions. Our mission is to form true partnerships with our clients, listening and collaborating to create tailored, practical, and results-oriented solutions that enable them to thrive and adapt to changing needs.

Our core services include strategy, design, and development of Knowledge and Information Management systems, with proven approaches for Taxonomy Design, Project Strategy and Road Mapping, Brand and Content Strategy, Change Management and Communication, Agile Transformation and Facilitation, and Enterprise AI.

info@enterprise-knowledge.com | 571-403-1109 | @EKConsulting