Graph Databases & RDF

Graph databases use graph structures with nodes, edges, and properties to represent and store data. If this rings a bell from network modeling, you're absolutely right. Just like network models, graph databases are based on graph theory. The main constructs for representing graph data are nodes, which correspond to entities such as a particular person or a disease, edges that connect nodes to each other and which correspond to relationships, and properties, which are attributes of a node or an edge. Here's a simple example of a visualized graph dataset. According to this dataset, Lucy lives in Edinburgh, and she knows James, who knows peter. Querying graph data is nothing more than traversing the graph. For instance, to find all of Lucy's friends or friends. What we've described so far is the general idea of graph databases. In practice, there are three main graph data models, the property graph, RDF, and hypergraphs. Let's have a closer look at RDF. In RDF, we can represent data in the form of triples. That is, statements consisting of a subject, a predicate, and an object. For instance, Lucy lives in Edinburgh, is a triple, where Lucy is a subject, lives in, is a predicate, and Edinburgh is the object. This is visualized like this. The statement 'Peter is 22 years old' can also be expressed as a triple, where Peter is a subject, age is a predicate, and 22 is the object. Visualized like this. Note the difference in the graphical notation for 22, and Peter, or Edinburgh. This is because 22 is an integer. So it is considered to be a literal value, while Peter and Edinburgh are not literals but resources. What do we mean by resources? In RDF, resources can be anything, including documents, people, physical objects, and abstract concepts. They denote something in the world. So, the resource Peter refers to the actual person called Peter. The idea of resources is so central to RDF that its name has to do with resources. RDF stands for Resource Description Framework. RDF is also based on the idea of identifying things using web identifiers called Uniform Resource Identifier or URIs. URIs can be used for both resources and properties. So, in the previous example, we could use URIs for the resources Lucy, Edinburgh, and Peter, as well as for the properties lives in and age. By using URIs, we can uniquely identify these things. And thus, be sure that when we say, for instance, Edinburgh in our dataset, we're referring to the city of Edinburgh, which is the capital of Scotland. And where did we get this URIs from? Even though you can make up your own URIs for the things that you want to refer to, it is best practice to use existing URIs. These are typically created by organizations and working groups in the RDF community and they're published on the web, so you can look them up. In this example, I've reused the Edinburgh URI created by dbpedia, and I have also used existing terms from the friend of a friend vocabulary for lives and age. Again, specified through URIs. However, I've made up my own URIs for Lucy and Peter, as these are fictitious people. As we've already mentioned, URIs allow us to uniquely identify resources, thus avoiding any ambiguity. What is more important is that by uniquely identifying resources, we can easily link data about the same resource. Suppose that, in addition to the previous RDF dataset about Lucy and Peter, we had a separate RDF dataset according to which Lucy knows James and James knows Peter. Since the two datasets used the same URIs for Lucy and Peter, there's no need to do any manual linking. We simply reuse the existing nodes for Lucy and Peter. As you can see, merging different RDF datasets is simply a matter of bringing the two sets of RDF statements together. In other words, all we do is form the graph of all triples from each individual graph. This is considerably easier compared to merging data that sit in separate relational databases. So far, we've been looking at RDF graphs, which visualize RDF data. But how do we write down RDF statements? There are various forms of syntax for expressing RDF. One of the most popular ones is Turtle, which stands for Terse RDF Triple Language. In its simplest form, an RDF triple statement is a sequence of subject, predicate, object terms separated by white space and terminated full stop after each triple. So, our first RDF graph can be written down like this. The first line corresponds to this part of the graph and the second to this one. As you can see in this example, full URIs, should be enclosed in angle brackets. We've now covered the fundamentals of RDF. Even though we've mostly looked at simple examples, The same ideas hold for complex biomedical datasets. We can express biomedical data in the form of RDF triples such as protein P29474 is associated with asthma. We use URIs to refer to biomedical terms such as asthma, so that there's no ambiguity when sharing the data. And more importantly, we can easily merge different biomedical RDF datasets, allowing us to explore indirect relationships between different concepts, whih can have a significant impact on hypothesis generation.