Info. Mgt. articles – How Semantic Graph Techniques Ease Data Integration

Big data benefits many industries, healthcare in particular, where access to data can result in earlier and more accurate diagnoses, better disease management, enhanced efficiencies and improved outcomes. The possibilities are even greater when truly large stores of data are available for analysis. A wide and deep data pool fuels real-time analysis, trustworthy predictive analytics, and profound learning—smaller data sets simply can't reveal the big picture.

The flood of information captured by diagnostic imaging systems, health monitoring devices, and electronic medical records have ensured that we don't have to worry about a healthcare data drought. IDC predicts that U.S. healthcare data alone will grow to 2,314 exabytes by 2020 (one exabyte is equal to 1 billion gigabytes), up from a mere 150 exabytes in 2010. A significant amount (about 75%- 80%, according to studies by Gartner, IDC and IBM) of healthcare data is unstructured images, graphics, videos, text messages, notes, recordings of conversations, transmissions from sensors and medical monitoring devices, and more – and can't be neatly organized into a relational database.

Like other industries, the healthcare industry needs to be able to work with structured, semi-structured and unstructured historical, near-time, and real-time data. The big data challenge is finding the most efficient, trustworthy ways to integrate unstructured and semi-structured data with structured data. The difficulty of this process largely depends on the type of database in use – Relational, NoSQL, Property Graph, or a Semantic Graph database.

Know Your Database Architecture

All databases house collections of data sets. The architecture of a database defines its suitability for various use cases. Let's dive into the key differences between relational, NoSQL, Property Graph and Semantic Graph databases as they relate to data integration.

Relational

A relational (SQL) database works well for storing the types of information that can be easily broken into defined sets. In a relational database, data is organized into "tables" (or "relations") in accordance with how a piece of data relates to all of the other data in that collection. Tables are comprised of rows and columns, with rows representing a single entry (instance) with a key that can be linked to rows in other tables. Each column ("attribute") represents a value that further describes the row.

As an example, in a table that contains hospital patient data, each row would represent a particular patient. Each column would contain information about that patient: name, address, contact information, insurance information, admittance date, release date, and so on.

A relational database has long been a satisfactory choice for the capture and management of structured data, along with enabling decision support analysis. But the limits of relational databases quickly become clear as data stores grow. Relational databases simply don't scale well, as finding approximate patterns across numerous fields in a large database is a demanding process that can easily become bogged down. Additionally, while schemas can be defined to enable analysts to work with unstructured data, preparing the data for consumption significantly impacts the availability of data.

NoSQL

NoSQL database technologies were developed in response to the limitations of relational databases and are well suited for applications with diverse data types with very large volumes of both structured and unstructured data. NoSQL databases do not require schemas, which makes data entry and exploration noticeably more nimble than a relational database.

Graphs

Part of the NoSQL ecosystem, Graph databases have become the fastest growing segment of the NoSQL Market. A Property Graph database is essentially a collection of nodes and edges where each node represents an entity (such as a person or business), and each edge represents a connection or relationship between two nodes. Nodes can have any number or type of associated relationships. This architecture makes a graph database particularly well suited for exploring highly connected data to find commonalities and anomalies.

Semantic Graphs

An advanced Graph Database approach, a Semantic Graph Database or Triplestore" makes the addition of new information, not anticipated in the original database design, far more straightforward. In fact, Semantic Graph Databases are so flexible that database designers do not have to create a schema up front but can build an ontology based on the data they need to include, editing it as they go. Although nothing prevents the architect from creating an initial ontology.

Semantic Graph Databases are most valuable for complex metadata applications where the number of classes (i.e. types of objects) change daily, properties within classes change on-the-fly, and it is critical have self-descriptions of data. Grounded in formal logic, semantic analytics can easily encompass associative and contextual concepts for richer data analysis, which provide a more expansive, exploratory querying experience. As noted in David S. Frankel's article, "*How*

Semantics Can Take Graph Databases to New Levels," querying a database using formal semantics provides the ability to "infer logical consequences from a set of asserted facts or axioms Reasoners grounded in formal semantics can be potent tools when managing large graph databases."

Integrating Databases

Given the structural flexibility of Semantic Graph Database, integrating databases is a virtually effortless, "bottom-up" process, rather than the traditional and time-consuming "top-down" masterdata approach. Using the latter approach, with assumption that the various data sources being integrated will contain some level of conflicting information, requires that you develop a probability scheme capable of ascertaining the entire "truth" for all the data that you will integrate. As you might imagine, this process will almost always take significant amounts of time and effort.

But with the Triplestore architecture, you can keep (most) of the data in its original databases as you build out a set of triples and rules to integrate it. Triples are easy to define, as they use a semantics data model (subject, predicate, object) to link two entities (which can be people, places, or things) – the relationship between them subsequently forms the "triple."

Example: **Subject:** JohnSmith **Predicate:** worksIn
Object: Miami

When linked together, triples form a graph. The graph for our example could extend out to all the John Smiths working in Miami, all John Smiths in the Southern States, all Smiths anywhere, or anyone who "worksin" or, for that matter, "livesin" Miami, or Florida. The standard language for writing triples is RDF (Resource Description Framework), and the standard query language is SPARQL.

Writing a triple is straightforward – it's easy to correctly

integrate databases with no loss of “truth.” And, since triples form graphs (interconnected networks comprised of data relationships), they also accurately describe the most complex information structures and connections.

Making the move to a more robust, but relatively new database architecture may seem daunting. But consider this: Triplestores offer the rigor of relational databases, the scalability of big data systems, and complete support for big, complicated joins. Semantic Graph Databases are simply the best option for integrating disparate data types, and the architecture provides significant adaptability for future use cases.

Dr. Jans Aasman is CEO of Franz Inc., which specializes in Graph database t