Entity Event Knowledge Graphs for Data Centric Organizations

Jans Asman, PhD
CEO - Franz Inc.

Introduction:

In evaluating the business focus of most Enterprises one finds there is usually only one absolute core entity, with perhaps one less critical secondary entity around which all the business revolves. For example, a hospital's core entity is the patient and a secondary entity would be providers. Banks have the customer as its core entity and a B2B call center is focused on the companies they are contacting to uncover sales opportunities. In nearly all cases, this core entity is found scattered in the numerous siloed databases that have been developed by different groups, departments, and divisions due to ever changing needs.

The challenge all these enterprises fail to solve is that information about the core entity is hidden in literally thousands of separate and disconnected relational databases. Relatively few enterprises can provide a comprehensive 360 overview of their core entity and those that are close have invested in scores of IT teams to build yet another bespoke data mart on top of silos, enterprise data warehouses (EDWs), and data lakes in order to provide the answers and reports necessary for a successful business. Even worse is the new trend of quickly building new apps on top of the underlying silos, as this cements the underlying data infrastructure. No one dares to touch the data in case one of the hundreds or thousands of apps will break. For a great description of the situation being created by this approach, Dave McComb's book Software Wasteland offers insight and solutions that should be considered by data-architects.

Thought leading architects in big enterprises have always dreamt of a data centric organisation that has one semantic and perfectly unified representation of all of its core entities. This then would drive all reporting, analytics, and applications. As a result of this dream these enterprises spent hundreds of billions of dollars in the past decades on data integration, master data management, and data lakes. These efforts have proven of limited success because they fail to approach data from the entity viewpoint.

Recent market trends show the approach with momentum to solve this problem is to build "Knowledge Graphs". Gartner for the last two years has identified the validity of this technology approach by offering reports and increasing the Knowledge Graph's visibility in their hype cycle research. One of the primary reasons for this increased visibility is the fact that nearly every Silicon Valley giant and most Fortune 500 companies have begun investing in Knowledge Graphs for advanced analytics and AI.

Entity-Event Knowledge Graphs:

To support ubiquitous AI, a Knowledge Graph system will have to fuse and integrate data, not just in representation, but in context (ontologies, metadata, domain knowledge, terminology systems), and time (temporal relationships between components of data). The rich functional and contextual integration of multi-modal, predictive modeling and artificial intelligence must be considered for a modern, scalable, enterprise analytic platform. Building from entities requires a new data model approach that unifies typical enterprise data with knowledge bases such as taxonomies, ontologies, industry terms and other domain knowledge. The Entity-Event Data Model we have developed, in conjunction with several customers and partners, puts core entities of interest at the center and then collects several layers of knowledge related to the entity as 'events'. The events represent activities that transpire in a temporal context. Using this novel data model approach, organizations gain a holistic view of customers, patients, students or important entities and the ability to discover deep connections, uncover new patterns and attain explainable results.

This white paper provides a summary of the main characteristics of Franz Inc.'s approach to Entity-Event Knowledge Graphs.

1. Entities and Events Represented as Hierarchical Trees.

One useful way to think about Semantic Graph Databases is that they are very flexible object oriented databases where we have a schema (ontology) that determines per type of object the allowed attributes, datatypes and links to other objects. For this paper we do not discuss much about semantic graphs, but treat entities and events as objects.

As the title of this section states, core entities and its events are organized in a hierarchical tree that consists of two levels and an associated knowledge base ("KB"). In the following we will use some examples that come from production based Entity-Event Knowledge Graph ("EEKG") implementations in the domains of healthcare and telecommunications.

Level one of every tree is the core entity and it serves as the root of the tree. In a hospital this entity is the patient, in a telco it is a customer. Each of these objects would have the obvious and expected demographic attributes.

Level two of the tree are the events and sub-events. In a hospital, events are hospital visits with sub-events such as diagnostics, tests, medication orders, procedures, vital signs, etc. In a telco events are phone-calls, sms-s, calls to the customer care department, bill payments, phone activations, and the like.

Note that these event objects are very regular and just what you might expect: they always have a main type and a start and end-time and then a few other properties that distinguish them from other events. The first thing we want to stress is that this is where the unification and

simplification happens. Everything that happens to a patient, everything that a telco customer does, and every financial transaction for a customer of a bank has roughly the same simple event structure. The second thing we want to stress that this approach makes the EEKG design uniquely future proof: if you decide to add yet another type of event you don't have to make any changes to earlier data, you just define a new event type and you can start adding new events of that type.

The associated KBs are not part of the tree but they are crucial to the entire operation of the EEKG. The KBs contain all the domain dependent data needed to make the data in the tree meaningful. For healthcare a knowledge base contains ontologies, taxonomies and thesauri for drugs, diseases, procedures, clinical pathways, and everything else important for medicine (like the facts that *dropsy* and *congestive heart failure* are two names for the same disease and *paracetamol* and *acetaminophen* are two names for the same drug.) For a telco the KBs would contain information about devices, location data, gis data, etc.

Here are some very simplified examples for healthcare and a telco. Note that in Figure 1 we see that the core Entity, on the left, is a patient, and Events appear in the middle. Even demographic data are events with a start time and an end-time given that people change addresses, telephone numbers, and sometimes even gender. The main events in the figure are encounters with sub-events like diagnostics, medications, procedures. On the right we have an artistic expression of the knowledge bases, in this case we show some core diagnostic concepts and their relationships in the taxonomy space.

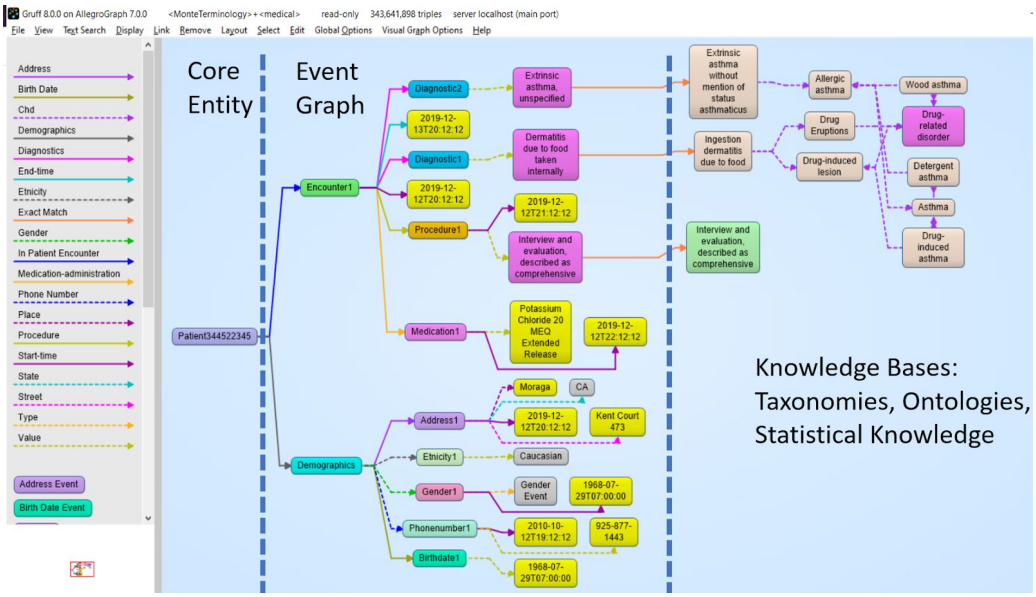


Figure 1.

If we then look at Figure 2 we see the same tri-partition: the customer is a core entity in our telecom database. Then communication events in the middle, and the knowledge base (showing place names in the USA) on the right. The place name KB is linked to the open data graph from geonames.org.

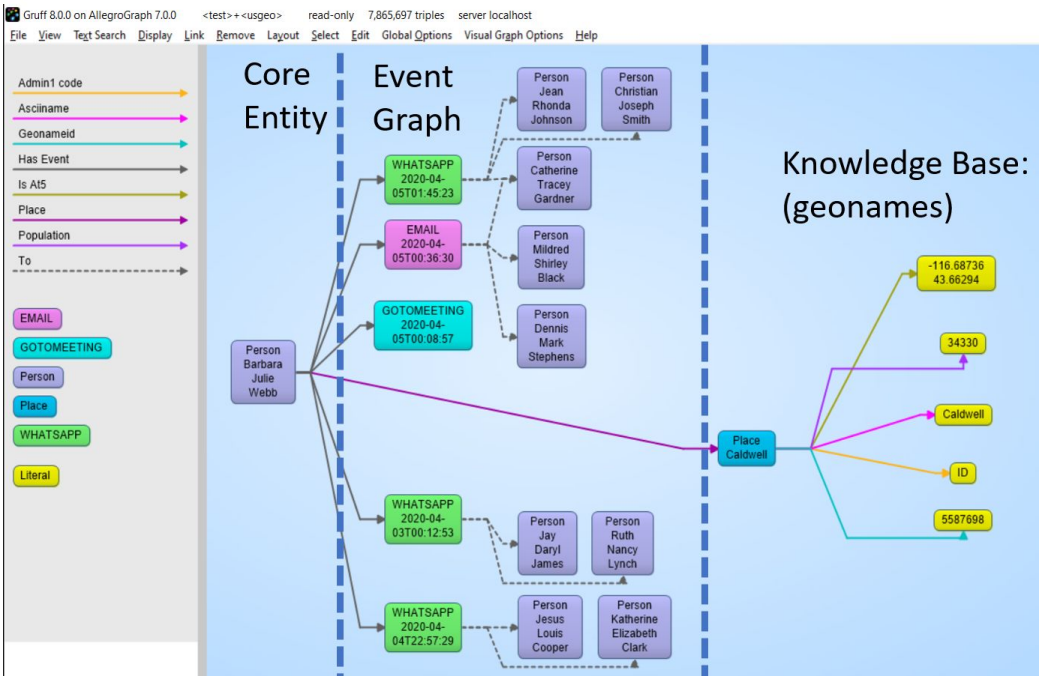


Figure 2.

2. Entity-Event Trees Logically and Physically Grouped Together.

One of the important properties of Semantic Knowledge Graphs is the ability to easily group elements of the graph into sub-graphs. So even if a patient or a customer might be a wide and deep tree with thousands of events and subevents, every fact in the tree has a group tag (for readers well versed in semantic graph databases: every entity lives in it's own graph, that is, the fourth element of every triple in the patient or telco tree has the entity as its own graph). Because of AllegroGraph's indexing technology every fact about a customer or patient can be found with a single physical disk access as a graph or as a JSON object that is easy to process by application and UI designers.

Benefits of the EEKG Model

Having described the main characteristics we can describe some most important benefits of this approach.

A. Easy to Understand the Data.

This is an important benefit for the business, for data scientists, and application developers. As we describe above, the entity-event approach makes our Knowledge Graph extremely uniform. To someone already familiar with graph databases we can explain the shape and contents of a typical knowledge graph in half an hour and then they can start writing queries. If you don't know graph databases it may take a little bit longer but not dramatically so. Compare that with the

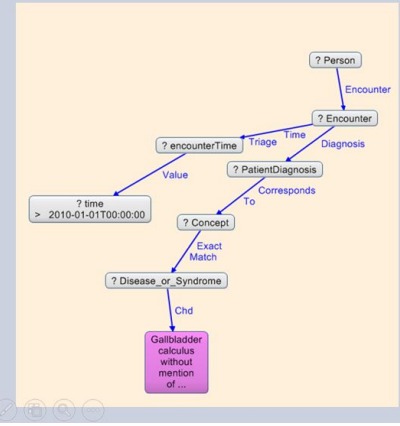
weeks and weeks you would need to learn to understand even the basics in your data lake, silos, or EDW.

B. Easy to Write and Execute Complex Queries.

Because the data is so uniform, queries that are very complex in a relational database become very easy to write in an entity-event knowledge graph. The following figure shows how we can do a query to find all the people that had ‘gallbladder calculus’ in 2010 and later. On the right part of the picture we see part 1 of 3 of a real SQL query to get that out of the database. On the left we see the graphical query we can write in our visual tool Gruff to create the SPARQL query on the top.

SPARQL

```
Select (count(distinct  ?pt) as ?count) where {
  ?pt cdm:encounter ?enc .
  ?enc timew3c:begins/upper:value ?date ;
   cdm:diagnosis/upper:correspondsTo/skos:exactMatch ?cui.
 filter (?date > '2010-01-01T00:00:00'^^xsd:dateTime)
 ?cui mth:chd*  mth:C0497327.}
```



SQL

(Part 1 of 3)

```
Select dx.dxCode, dx.dxCount, cl.dxClassName, cl.ClassCount
From (select distinct c.person_source_value as person,
c.condition_source_concept_id as dxCode, f.concept_name as
dxName,
a.concept_id_2 as dxClass, e.concept_name as dxClassName from
omopv5.concept_relationship a
join cdrn.condition_occurrence c on
a.concept_id_1=c.condition_source_concept_id join
omopv5.concept_ancestor d
on d.descendant_concept_id=a.concept_id_2 join omopv5.concept e
on e.concept_id=d.ancestor_concept_id
join omopv5.concept f on f.concept_id=d.descendant_concept_id
where a.relationship_id='Maps to' and
d.min_levels_of_separation=2 and e.standard_concept='S') dx JOIN
(select distinct c.person_source_value as person,
c.condition_source_concept_id as dxCode, f.concept_name as
dxName, a.concept_id_2 as dxClass, e.concept_name as
dxClassName from omopv5.concept_relationship a join
cdrn.condition_occurrence c on
a.concept_id_1=c.condition_source_concept_id join
omopv5.concept_ancestor d on
d.descendant_concept_id=a.concept_id_2
join omopv5.concept e on e.concept_id=d.ancestor_concept_id join
omopv5.concept f on f.concept_id=d.descendant_concept_id
where a.relationship_id='Maps to' and d.min_levels_of_separation=2
and e.standard_concept='S') cl on dx.dxClass=cl.dxClass
order by cl.dxClass desc
```

C. Provides a 360 Overview in Milliseconds

We described earlier that every fact in an entity-event tree is tagged with the id of the entity. This gives us a powerful advantage: with a one line query we can get all the data about an entity in just a few milliseconds. To make life easier for application developers we can also return the data about the patient in a variety of formats including as JSON or JSON-LD objects.

D. One-time Mapping and ETL Process

All the data about our example patient or a telco user resides in literally thousands of siloed databases, or if you are lucky, in one or more enterprise data warehouses (EDWs). For example: we have seen at our hospital partner that just one EDW might have 4,000 tables and more than 20,000 columns. Any single patient can be theoretically in any of these tables and columns so how would you ever retrieve all the data for just one patient?

With our entity-event approach extracting data from this EDW became a one time mapping process. Potentially every table/column is mapped to a particular event-type and event-property

and this mapping is stored declaratively. From that moment on this mapping can be used to populate the entity-event tree.

The most important insight here is that this needs to be done only once. All future reporting, analytics, and feature extraction for machine learning can be done on top of this knowledge graph infrastructure without ever creating separate data marts.

E. Entity Event Knowledge Graphs can be Built Incrementally

When we explain this model to potential users one major concern is whether or not all the data in all data lakes needs to be copied into the EEKG. Fortunately, the answer to this question is no. First: we have seen that in most cases we only need 20 % of the relevant data available in data lakes and EDW, as the remaining data is for relational database bookkeeping not required for an EEKG. Second: our approach lends itself to starting with only a few types of events. Yes it would of course be fantastic if you could literally take all events that happened to your patient, customer, or product and put it straight into the EEKG from day one, but in reality we see that with just a few types of events you can already achieve value sufficient to convince management in larger organizations. Note that we said above that the shape of the data doesn't change when you add new types of events, so this approach is built for incremental development

F. Provenance at the Event Level.

Another concern that people raise is how you provide provenance to the data. The answer is simple: in the first two large EEKGs built in this style provided provenance as first class annotations of events. In both EEKGs the events had annotations about when the data was added, by what procedure and from what database/table/column. Note that in the hospital use case about 60% of the data is actual provenance data. Note that the annotations have the same shape for every conceivable type of event, whether it is a diagnostic or a billing event. So again no unnecessary explosion of types of provenance, just one shape and therefore, no extra complexity.

G. Security

In some use cases, especially in healthcare and finance it is very important to make sure that only the right people can see the right data. AllegroGraph has a mechanism for ensuring that, which was built according to requirements of the DOD and Intelligence agencies. It is called *triple-attributes* and is described in this tutorial. With this mechanism we can protect every fact in the Knowledge Graph. AllegroGraph and thereby Knowledge Graphs built in AllegroGraph are the only one that can provide this level of security.

H. Horizontal Scaling through Sharding

The entity-event model also lends itself easily and naturally to sharding, and thereby to unlimited horizontal scaling. Note that we described above how every fact in the entity-event

tree graph is tagged with the entity-id. We can use this entity-id for sharding purposes and we will describe in another paper how we can use this sharding strategy with AllegroGraph.

Concluding

AI applications and complex reasoning analytics require information from both databases and knowledge bases that contain domain information, taxonomies and ontologies in order to conduct queries. Some large-scale knowledge bases cannot be sharded because they contain highly interconnected data. AllegroGraph, with FedShard™, federates any shard with any large-scale knowledge base – providing a novel way to shard knowledge bases without duplicating knowledge bases in every shard. This approach creates a modern analytic system that integrates data in context (ontologies, metadata, domain knowledge, terminology systems) and time (temporal relationships between components of data). The result is a rich functional and contextual integration of data suitable for large scale analytics, predictive modeling, and artificial intelligence.

Financial institutions, healthcare providers, contact centers, manufacturing firms, government agencies and other large enterprises that use AllegroGraph gain a holistic, future-proofed Knowledge Graph architecture for big data predictive analytics and machine learning across complex knowledge bases.

This Entity-Event approach to simplifying enterprise data is the future for horizontally scalable Knowledge Graphs. We are happy to discuss with the reader how the approach discussed in this article can provide a solution for your enterprise application.

Email us: info@franz.com