Knowledge Base for Business Intelligence

System for population and linking of knowledge bases dealing with data, information and knowledge comming from heterogeneous data sources to provide pluggable Business Intelligence insight into applications.

Semantic Web (RDF Triples) based information manipulation and P2P node deployment architecture. Nodes clients/services provides endpointds for their sources (data sources or also another Nodes).

Augmented indexing, search and browsing facilities, analysis and mining, pluggable into existing deployments leverages the existing potential of applications to be Business Intelligence aware.

Triples (time based)

Albeit its lack of popularity in some environments, RDF triples are an exceptional serialization format. And when it comes to time related events or information, data can be serialized using the quad-extension mechanism which allows to attach a context resource to a triple, let say, time related knowledge.

One of our system’s attempts is to have things sorted by time and then arrange them in sets partitioned in base of that ordering. Time based triples may come from everywhere. It may seem difficult at first to think ordering all the events in a relational database. But think of Web related events: social media from which extract statements via NLP, page navigation data, mobile apps users behavior, IoT(s). All these are strongly time bound sources of data.

Sets arrangement

For the internal representation of triple’s knowledge we have choosen a set model oriented approach. This will hopefully allows us later to model easily Predicates upon those sets and a a graph representation mechanism based in the so called Nodes which are a definition of a Predicate for the Node itself and two other Predicates for its possible matching input/output nodes.

The set space of abstract Resource(s) is partitioned into Subjects (S), Predicates (P) and Objects (O). Resources can be anyone of these so they are represented in the form of an identifier ID = [CTX] [ID] [ID] [ID] being each part its corresponding SPO value and CTX is time data. Nesting (reification) of triples is allowable.

Sets arrangement (cont.)

Subjects intersects whit Objects to form Predicate Kinds. Objects intersects with Predicates to form Subject Kinds. Predicates intersects with Subjects to form Object Kinds. Then, all SPO intersects refer to the Triples set itselff.

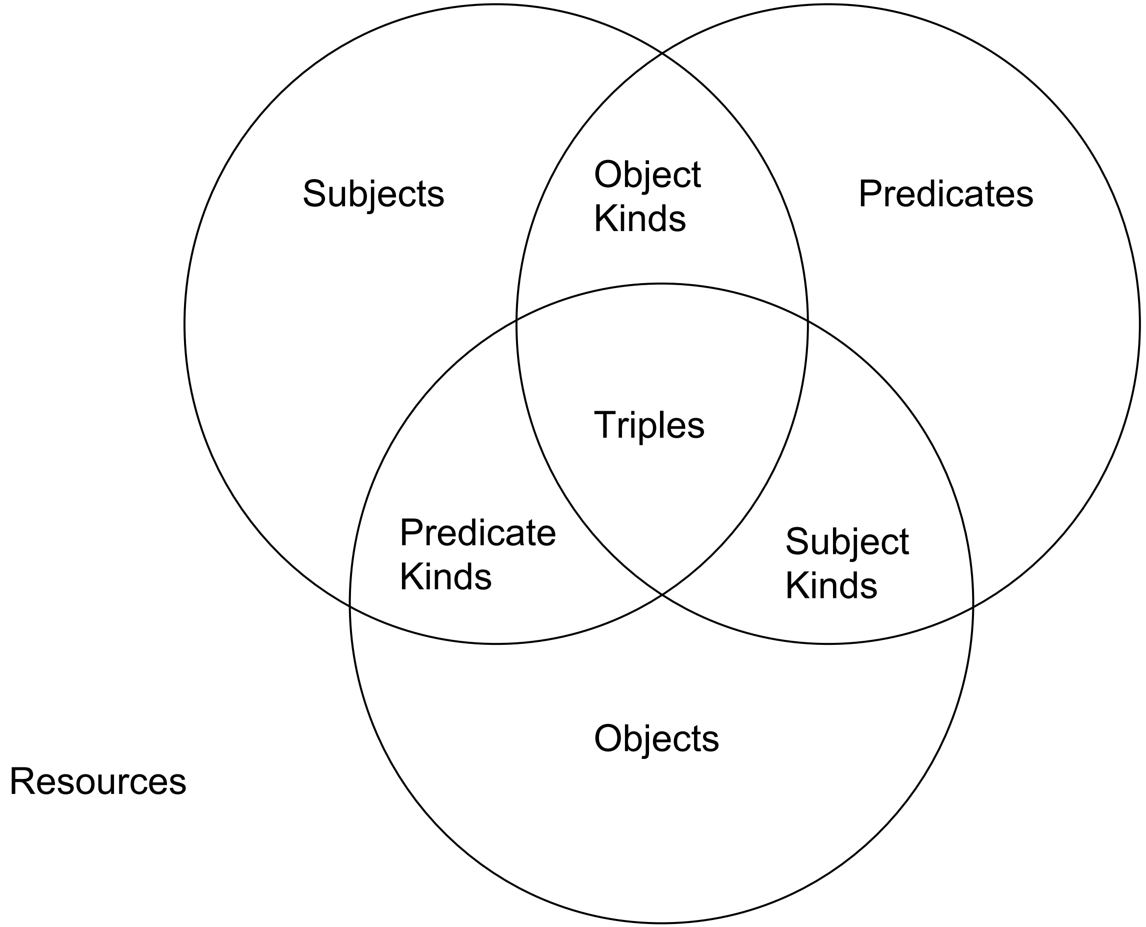
When we say ‘Kind’ for each of these SPOs we mean exactly that. A set of Predicates and its Object values characterizes a set of Subjects, given, for example, for its sex, age, etc. Idem for Predicates and Objects.

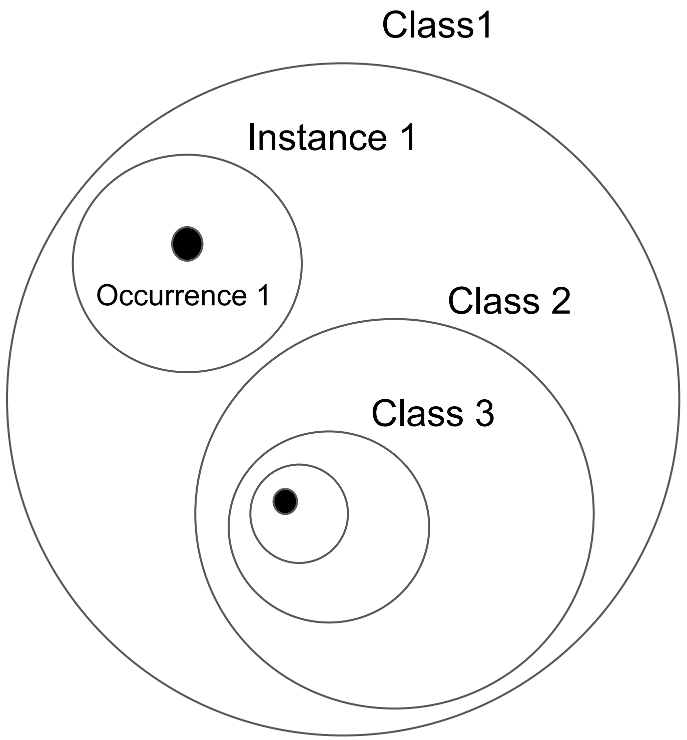
Due to Kinds being the definition of sort of types we need to partition this sets temporally and use this knowledge to partition SPO sets. With ‘temporally’ we mean that, for example, some Predicates usually occurs before/after anothers.

Sets arrangement (cont)

Sets arrangement consists of the partition of sets in the relative/abstract concept of classes, instances and occurrences. The more external a set is, the more before temporally is of all its contents. So, for example, one is first a son of, then a father of and then a grandfather of (Kinds). A Kind doesn't have occurrences.
So classes enclose other classes and instances. Instances encloses other instances and occurrences. Occurrences may enclose what may be interlinked with others SPO partitions. For each SPO set its occurrences are of its set type.

S class/inst: Predicate / Object
P class/inst: Object / Subject
O class/inst: Subject / Predicate


Subjects
Object Kinds
Predicates
Triples
Predicate Kinds
Subject Kinds
Objects
Resources


Class1
Instance 1
Occurrence 1
Class 2
Class 3

Inference

Temporal logic may be implemented using quad context in triples by retrieving / adding a source timestamp to triples, an ordering / offset context to Kinds and by logical operators over the set of SPOs by using three bits to tag time to occurrences in its context: each one regarding something happening in the past, present, future or their combinations.

A possible implementation of the set partitions could be a two-keyed hash map in which one can traverse classes, instances and occurrences, query for possible keys / paths and retrieve values based on the closure of the sets hierarchy. Multiple occurrences of the same SPO(s) are different set elements because of their instances / classes.

Inference (cont)

Predicates: predicates are triples of the form pred: set(pred, pred, pred), where pred may be any Resource type (SPO, Kind, Triple) or even another reified pred. Sets API based resolution. Reified response:

((person, leads, organization), bornIn, (place, country, ‘AR’))

Type subsumption: organization: company, club, etc.
S/O(s) as predicates: use Kind predicates.

Nodes: detect Predicate patterns (Node types). Aggregate Node predicates and their parent / child (graph in/out) Nodes (predicates). Example: Movie database: categories, films, roles and actors, ratings.

Application Object Model

Application Object Model for ease of translation of the underlying core components interfaces into a graph / facade and to provide a foundation in which to build upon the proposed P2P client / service abstractions (protocols)

Model
- types : Type[]
- entities : Entity[]

Type
- propertyTypes : Type[] (Map<String, Type[]>)
- entities : Entity[]

Entity
- type : Type
- properties : Property[]
- payload : Object

Property (extends Entity)
- value : Entity