Semantic Web & BI

Triples (Quads)

Sources of contextualized triple graphs

Analysis. & Discovery

Declarative application modelling

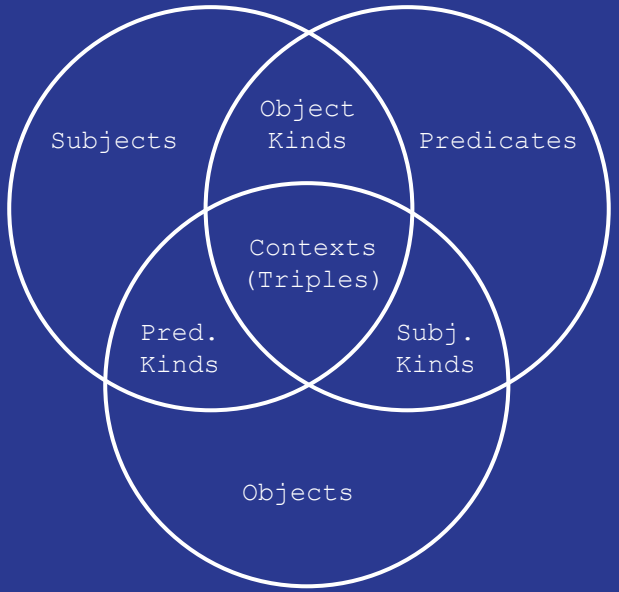
Sets API & Functional Model

Triples and their contexts are arranged in a three sets elements dispositions so each set represents one part of the triple and their intersections between two of the sets represents a ‘Kind’ corresponding for the third set. Each Kind aggregates ‘attributes’ and ‘values’ which accounts for the classes and metaclasses of the Resources of the third set themselves.

Element, Set, ElementType, Resource, Predicate. Hierarchies. Functional features. Type tree relation with hashed keys. All sets elements are subclasses of Resource and have all four triple/quad resource components (ctx, subject, predicate, object). The kind of Resource each part will holds depends on which Resource subclass is used.

Every set element is an instance of Resource, class which accounts for an individual or reifyied Resource (individual URI or another triple Resource treated as a Resource). Traversal over the sets structure is achieved performing joins between corresponding set resources components and using element types as functors in a functional environment.

SPO Model



Occurrence	Attribute	Value
Subject	Predicate	Object
Predicate	Subject	Object
Object	Predicate	Subject

Triples:

```
Occurrences:
[context] [occurrenceURI] [classID] [metaClassID]

Kinds:
[metaClassID] [classID] [attribute] [value]

Contexts:
[context] [Subject] [Predicate] [Object]
```

SPO Model

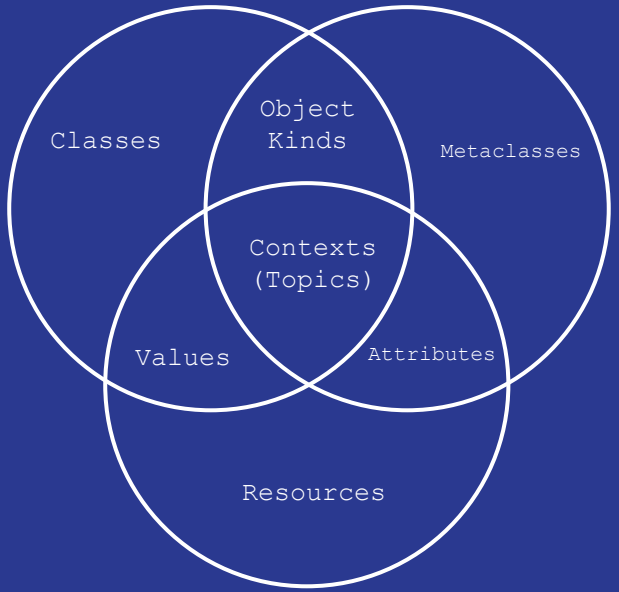
SPO Model is the basic abstraction for source triples/graphs which will be aligned/merged/augmented via the other set models.

Kinds aggregate classes/metaclasses hierarchies encoded in class/metaclass resources. Reifyied Kinds accounts for type hierarchy trees into each SPO set. A class is determined by the intersection of Subjects having the same attributes. A metaclass is the set of same attribute values for a given class.

Sets api manipulation. Sets hold hashed keys relating an element’s position into set types tree to an elements map. Functional algorithms.

Abstract Predi:cates, possible kind individuals: query.

Similarity Model



Occurrence	Attribute	Value
Class	Resource	Metaclass
Metaclass	Resource	Class
Resource	Metaclass	Class

Triples:

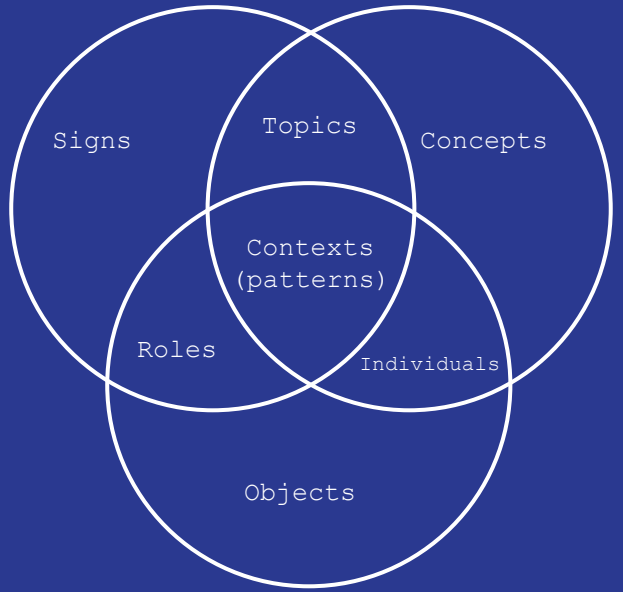
```
Occurrences:
[context] [occurrenceURI] [classID] [metaClassID]

Kinds:
[metaClassID] [classID] [attribute] [value]

Contexts:
[Topic] [Resource] [Class] [Metaclass]
```

Similarity Model

Topics inference / aggregation.
Resolve Resources class / metaclass. Merge.

Topics as a pattern to “reacts” to input triples. Merge topics. Update sets tree hash map keys. Identify similarity, later in Semiotic Model identify equivalence.

Semiotic Model



Occurrence	Attribute	Value
Sign	Concept	Object
Concept	Object	Sign
Object	Concept	Sign

Triples:

```
Occurrences:
[context] [occurrenceURI] [classID] [metaClassID]

Kinds:
[metaClassID] [classID] [attribute] [value]

Contexts:
[context] [Object] [Concept] [Sign]
```

Semiotic Model

Identify equivalent Resources and merge. Similar object graphs structures.

Signs: all SPOs.
Concepts: all SPO Kinds.
Objects: Reifyied sets of triples describing an entity. Topics aggregates object kinds.
Syntax analisys: Primitives. Merge different URIs, same meaning..

Contexts (patterns) as Purpose (Predicate) definitions.
Patterns “react” to input triples. Purpose driven declarative apps.

Applications

DOM model. Peers / Node applications.
Merge (models). Mapred / DCI / monads.

Nodes: Topics. Purpose. Transforms: rules flows events declaratively stated.
Possible individuals. Node nav hierarchies. Analysis.

Linked Topics Purpose driven apps. Declarative, domain driven development.

Backend: p2p nodes running services/clients (apps) declarative clients / agents.

Triple encoding and peers resolution. Registry, dataflow.

Applications and client agents evolve from domain data (and schema) specifications.