Application Service

Models Architecture

The idea is to enable model representations being equivalent (containing the same data) in various layers to be switched back an forth between each layer representation to be used in the most appropriate task for a given representation.

Reference Model

(Aggregation / Grammar)

ID
- primeID : long
- urn : string
- occurrences : IDOccurrence[]
- embedding : double[]

IDOccurrence : ID
- occurringId : ID
- context : IDOccurrence
- embedding : double[]

Statement : IDOcurrence (Property Graphs)
- context : ID
- subject : ID
- predicate : ID
- object : ID

Statements

Data: (IDOccurrence(ID), IDOccurrence(ID), IDOccurrence(ID))
Schema: (ID(IDOccurrence), ID(IDOccurrence), ID(IDOccurrence)

FCA Contexts. Prime IDs. Embeddings

FCA Prime IDs (Embeddings):

Each ID is assigned a unique prime number ID at creation time. FCA Context / Lattices built upon, for example for a given Data / Schema predicate / arc occurrence role, having the context objects being the statement occurrence subjects and the context attributes the statement occurrence objects, Predicate FCA Context: (Subjects x Objects). For a subject

statement occurrence the context is: Subject FCA Context: (Predicates x Objects and for an object statement occurrence role the context is: Object FCA Context (Subjectx x Predicates).

Embeddings: For an ID, its prime ID number plus all ID's occurrences embeddings. For an IDOccurrence, its ID class embeddings, its occurring ID embeddings and its context embeddings.

Embeddings similarity: IDs, IDOccurrences sharing the same primes for their embeddings in a given context. FCA Concept Lattice Clustering. (TODO).

Statements:
(Context, Attribute, Value)

TODO:
FCA / Multidimensional features (OLAP like):

Dimensions: Time, Product, Region
Units: Month / Year, Category / Item, State / City

Context : (Context, Attribute, Value)

Examples:
(soldDate, aProduct, aDate)
((soldDate, aProduct, aDate), Product, aProduct)
(((soldDate, aProduct, aDate), Product, aProduct), Region, aRegion)

Graph Model

(Alignment, Semantics, Sets / Kinds)

Context : IDOccurrence (Set)

Subject : IDOccurrence (Set)

Predicate : IDOccurrence (Set)

Object : IDOccurrence (Set)

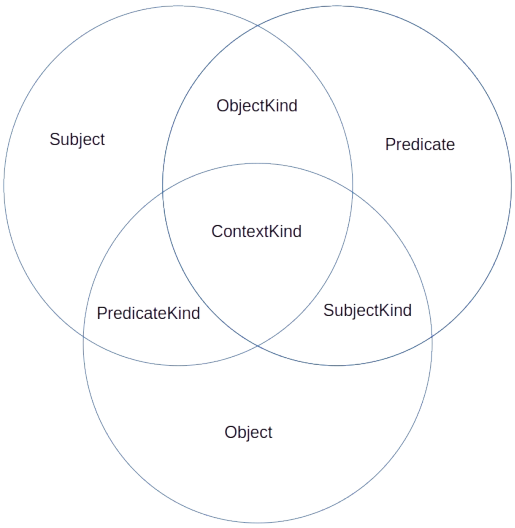
Kind<AttributeType, ValueType> : Interface
- superKind : Kind
- attributeValues : Tuple<AttributeType, ValueType>[]

Reification: Kind implementations extends / plays Subject, Predicate and Object roles in statement.

SubjectKind : extends Subject, implements Kind<Predicate, Object> (Predicates intersection Objects)
- occurrences : Subject[]

PredicateKind : extends Predicate, implements Kind<Subject, Object> (Subjects intersection Objects)
- occurrences : Predicate[]

ObjectKind : extends Object, implements Kind<Predicate, Subject> (Predicates intersection Subjects)
- occurrences : Object[]



Statements

Data: Context(Subject, Predicate, Object)
Schema: Context(SubjectKind, PredicateKind, ObjectKind)

Activation Model

(Activation, DOM / DCI / Actor, Role. Pragmatics)

Instance : IDOccurrence
- id : ID
- label : string
- class : Class
- attributes : Map<string, Instance>

Class : Instance
- id : ID
- label : string
- fields : Map<string, Class>

Context

- roles : Role[]

Role : Class
- previous : Map<Context, Dataflow>
- current : Map<Context, Dataflow>
- next : Map<Context, Dataflow>

Dataflow : Context
- role : Role
- rule : Rule (TODO)

Interaction
- actors : Actor[]

Actor : Instance
- previous : Map<Context, Transform>
- current : Map<Context, Transform>
- next : Map<Context, Transform>

Transform
- actor : Actor
- production : Production (TODO)

COST (Conversational State Transfer)

REST API is in initial state for a given context. The client retrieves the ‘current’ role context dataflow representation instance (Interaction, Actor, Transform), process it (DSL, ‘Activates’ and invokes API for the given representation Transform) and posts back the activated representation. The service then is able to determine the next Dataflow Role representation instance in a given use case (Context). TODO: Populate (infer) Dataflow Roles rules (state flows), Populate (infer / execute) Transform Actors productions using data encoded in the proposed models.

Statements

Data: (Interaction, Actor, Transform)
Schema: (Context, Role, Dataflow)

Components (Services) Architecture

Services Layout

Services: Reactive Endpoints. Consumers / Producers <MessageType>. WebFlux: onMessage (post), nextMessage (subscribe). Reactive Consumers / Producers with Message IO in context (session / dialog history) #EAP.

Application Service

Augmentation Service

Handles Services Subscriptions. Dispatch Kafka (Reactor) Streams. Saga Pattern (MessageType logs). Reactive Consumer / Producer with Message IO in context (session / dialog history) #EAP.

Aggregation Service

Consumes: DataSources (CSPO) URI Strings Statements.
Produces: Reference Model (ID, ID, ID, ID) Statements.
Features:

- IDs / Embeddings.
- FCA Contexts (Clustering).
- Basic types / attributes inference (FCA).

Alignment Service

Consumes: Reference Model (ID, ID, ID, ID) Statements.
Produces: Graph Model (CSPO) Statements.
Features:

- Upper (inferred) ontology alignment.
- Links completion, Ontology Matching.
- Types / Kinds Inference. Order (hierarchies / dimensional).

Activation Service

Consumes: Graph Model (CSPO) Statements.
Produces: Activation Model DCI (Context, Interaction, Role, Actor) Statements.
Features: Use case inference / execution. Browse / Run transactions across integrated applications.

Services Dataflow (see Protocols)

DataSource ↔️ ApplicationService ↔️ Augmentation ↔️ Aggregation ↔️ Alignment ↔️ Activation ↔️ Augmentation ↔️ ApplicationService ↔️ Producer

Helper Services

MCP Host. Servers.
DIDs / FCA Contexts / PrimeIDs.
Index / Embeddings / Similarity.
Shared State (representations aware backend).

Index Service

TODO

Naming Service

TODO

Registry Service

TODO

Protocol (Message Types) Architecture

Context / History (session) aware Dialogs #EAP

TODO

CSPO URI Strings Statements

Statement<String, String, String, String>.

Reference Model Statements

Statement<ID, ID, ID, ID>.

Graph (Sets) CSPO Statements

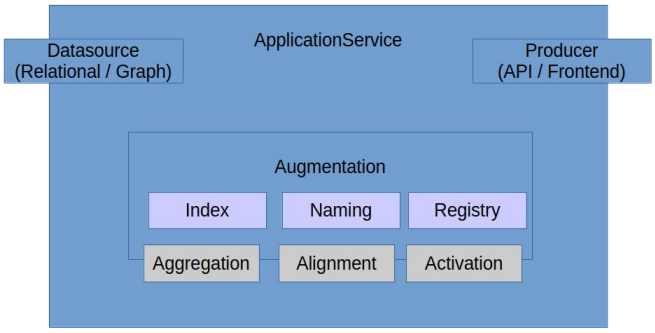
Statement<Context, Subject, Predicate, Object>.

Activation (DCI) Statements

Statement<Context, Interaction, Role, Actor>.

Interactions (Reactive Services Messages Dataflow) Architecture

Functional (Use Case) Architecture



Integration Example

Configured Datasources of Applications to integrate from (examples). Produces (CSPO) URI Strings Statements Streams.

Aggregation: ETL Consumes previous Statements, Produces Reference Model IDs / IDOccurrences Statements Streams.

Prime IDs / DIDs Assignation. DCI Contexts Concept Lattices Tables population.

Alignment matches sources equivalences.

Activation discovers available Contexts Interactions and past Interactions Transactions.

Producer UI selects next step from current state (available transaction or next if previous state). Submits Context Interaction Role Actor Form: Browses previous or perform next.

Activation consumes Producer statements and forwards to Alignment for backend sync processing.

Alignment matches Activation inferred Contexts into DataSources source data.

Aggregation converts Alignment source aligned data into Datasource quads.

Augmentation submits stream for updating and syncing to Datasource.

TODO

Tools