Aggregation:

(Class, Instance, Attribute, Value);

* Class (Employee, Married) : Table
* Instance (anEmployee, aMarried) : PK (row)
* Attribute (Employer) : Column
* Value (anEmployer) : Value

Activation:

(Metaclass, Context, Role, Occurrence);

* Metaclass (Employment, Marriage)
* Context (anEmployment, aMarriage)
* Role (Employee, Employer, Wife, Husband)
* Occurrence (anEmployee, anEmployer, aWife, aHusband)

Mappings (Alignment):

(Dimension, Measure, Unit, Value);

* Dimension (Age, Distance)
* Measure (DoB, elapsedTime)
* Unit (Years, Km/h)
* Value (anAgeYears, aDistanceKmts)

Augmentations:

* Aggregation: sort / train. Clustering. N-ary relationships (marriages, employments). FCA: P(S, O) Contexts.
* Activation: predict. Classification. Age / DoB, married / marriage (wife, husband, dateMarried). Sets (CSPO, Kinds, Statements).
* Alignment: complete / map. Regression. State (single / married). Dimensions: compose / translate. Sort: current Value, next Measure. Tag: Classifications, Clusterings.
* Merge Graphs: Ontology Matching. Dimensional Alignment.

Image Recognition:

* Context (Class, Metaclass, Dimension): Class
* Subject (Instance, Context, Measure): Blob
* Predicate (Attribute, Role, Unit): x1, y1, x2, y2, props / tags
* Object (Value, Occurrence, Value): x1, y1, x2, y2, props / tags
* Context: Subject Classification Class (Person).
* Subject (train / predict): example.org?aPerson&data=Base64Data.
* Attributes / Values: Metadata.

Train / Predict / Matching:

Aggregation, Activation, Alignment: Statements, Kinds, CSPO (URNs).

CSPOs: URNs, Reified Kinds, Reified Statements.

Kinds. Inferred SPO Types (Sets):

SubjectKind: S(P, O)\*. Aggregated Subject Predicates / Objects.

PredicateKind: P(S, O)\*. Aggregated Predicate Subjects / Objects.

ObjectKind: O(P, S)\*. Aggregated Object Predicates / Subjects.

Augmentation Statements:

Aggregation / Reification:

Statements / Kinds Reified as URNs.

Aggregation:

* Context (Class): SubjectKind. Employer
* Subject (Instance): URN. anEmployer
* Predicate (Attribute): URN. Employee
* Object (Value): URN. anEmployee
* Context (Class): SubjectKind. Employee
* Subject (Instance): URN. anEmployee
* Predicate (Attribute): URN. Employer
* Object (Value): URN. anEmployer
* Context (Class): SubjectKind. Position
* Subject (Instance): URN. aPosition
* Predicate (Attribute): URN. Employee
* Object (Value): URN. anEmployee
* Context (Class): SubjectKind. Employee
* Subject (Instance): URN. anEmployee
* Predicate (Attribute): URN. Position
* Object (Value): URN. aPosition

Activation:

* Context (Metaclass): SubjectKind. Works, Employs, Performs
* Subject (Context): AggregatedReifiedAggregation SKs. HasEmployer, HasEmployee, HasPosition
* Predicate (Role): AggregationSubjectKinds. Employer / Employer, Employee / Position
* Object (Occurrence): AggregatedReifiedAggregation OKs. EmployerOf, EmployeeOf, PositionOf

Alignment:

* Context (Dimension): SubjectKind. Employment
* Subject (Measure): AggregatedReifiedActivation SKs. Employed, Employing, Performing
* Predicate (Unit): ActivationSubjectKinds. Works, Employs, Performs
* Object (Value): AggregatedReifiedActivation OKs. EmployeedAt, EmployingWho, PerformingPosition

Aggregated Reified Statements:

Kinds Layered Predicate Matching. PK: P(S, O). Layered Pattern Matching:

Activation (S, O) from Activation Predicate / Aggregation Subject Kind Attributes (PKs SKs / OKs).

Dimensional (S, O) from Alignment Predicate / Activation Subject Kind Attributes (PKs SKs / OKs).

Functional Composition:

C: Transform Context. Mapping State.

P: Transforms, Mappings.

S, O: Monads, Functors.

Graph Navigation. Kinds Pattern Matching:

SK as P / O. PK as S / O. OK as P / S.

Reified Kinds. SK / S, PK / P, OK / O.

S.flatMap(P) : O;

O.flatMap(P) : S;

Aggregation Functional Statements (data):

(anEmployee(Employer) : anEmployer);

(anEmployer(Employee) : anEmployee);

(anEmployee(Position) : aPosition);

Activation Functional Statements (schema):

(HasEmployer(Employer) : EmployerOf);

(HasEmployee(Employee) : EmployeeOf);

(HasPosition(Position) : PositionOf);

Alignment Functional Statements (behavior):

(Employed(Works) : EmployedAt);

(Employing(Employs) : EmployingWho);

(Performing(Performs) : PerformingPosition);