**To do**

* CQRS
* Merge TOCs.
* Merge Contents.
* Content / Topics:
* Concepts, Design, Architecture.
* Add bibliography / tools use cases / components. Bookmarks, Lectures. Notes: Scrapbook.
* Concepts:
* Data, Information, Knowledge.
* Data, Schema, Behavior.
* Models: Layers Message IO Dataflow Bus.
* Models: Sets Contexts, Kinds, Resources Layer.
* Events Sourcing / Models Bus IO.
* Input Message Augmentations:
* Aggregation: Populate Sets.
* Alignment: Addressing / Encoding / Matching.
* Functional Activation Dataflow API:
* Resource Monad
* Kind Monad
* Context Monad
* Message Monad
* Event Monad
* API: Dataflow:
* API: Command. CQRS (CUD, R): Context (Mapping Contexts).
* API: Event: Command / Message (Context, Template: D, Mapping:  C, Transform: I);
* Dataflow: Event dispatching. Event Message / Command Context augmentation.
* Activation: Topics reacts to Events according API. Context, Kind, Resource Chain of Responsibility. Performs CUD/R and a response stream relevant to the operation performed.
* API: onEvent(Event) : Event. Order / Comparisons / Workflows.
* Activation: Statements.
* Template Data Roles (Kinds) selectors / predicates Matching Statements. Data.
* Activation: Mappings.
* Statements Matching Mappings. Schema / Context.
* Activation: Transforms.
* Mappings Matching Transforms. Behavior / Interaction.
* API: Core Model Bus Topics: Contexts, Kinds Resources.
* API: Core Model Transforms / Mappings Functions.
* API: Dynamic Model (instances) Bus Topics: Resources, Kinds, Contexts.
* API: Dynamic Model (instances) Transforms / Mappings Functions.
* Functional API (Monads Functions / Wrappers (Domain / Range): Sets Object Model:
* Example: Resource<Subject>, Subject<Resource>;
* CoSQL. Duals. Meijer.
* API Functions (domain / range: individual subjects / streams in context):
* getResource / getResources
* getKind / getKinds
* getContext / getContexts (Statement, Mapping, Transform)
* getContext / getContexts (CSPO)
* getSubject / getSubjects
* getAttribute / getAttributes
* getValue / getValues
* getMetaclass / getMetaclasses
* getClass / getClasses
* getInstance / getInstances
* getContext / getContexts
* getRole / getRoles
* getOccurrence / getOccurrences
* Models: Addressing / Encoding / Matching Layer.
* ResourceURNs Occurrences (Subjects):
* (ResourceURN, Resource, Kind, Context);
* (ResourceURN, Resource, Context, Kind);
* (ResourceURN, Context, Kind, Resource);
* (ResourceURN, Context, Resource, Kind);
* (ResourceURN, Kind, Context, Resource);
* (ResourceURN, Kind, Resource, Context);
* Addressing: Model Traversal: MapReduce
* ResourceURNs Contexts, Resources, Occurrences IDs Addressing / Encoding:
* ResourceURN : (ContextResourceURN, SubjectResourceURN, OccurrenceResourceURN);
* Matching: ResourceComparator(s).
* Models: DOM OGM / DCI / DDD / CDI: Restful objects Layer.
* RDF DOM OGM / DCI / DDD / CDI: Sesame Elmo.
* Resources / Kinds / Contexts: DOM / DCI / DDD Subjects, Concepts, Mixins, Behaviors.
* Functional Dataflow API. Model / Events Bus sync.
* DCI: Qi4j / Apache Zest (RDF / KeyValue / EAV EntityProvider).
* Dynamic Functional Contexts: Scala DSL Message Dataflows. Monadic Parser Combinators.
* Behavior / Factory: Roles. Parameterize (domain context) monadic functions applications.

**To do**

* CQRS
* Merge TOCs.
* Merge Contents.
* Content / Topics:
* Concepts, Design, Architecture.
* Add bibliography / tools use cases / components. Bookmarks, Lectures. Notes: Scrapbook.
* Concepts:
* Data, Information, Knowledge.
* Data, Schema, Behavior.
* Models: Layers Message IO Dataflow Bus.
* Models: Sets Contexts, Kinds, Resources Layer.
* Events Sourcing / Models Bus IO.
* Input Message Augmentations:
* Aggregation: Populate Sets.
* Alignment: Addressing / Encoding / Matching.
* Functional Activation Dataflow API:
* Context Monad
* Kind Monad
* Resource Monad
* Message Monad
* Event Monad
* Activation: Templates / Facets Statements Selectors / Predicates.
* Activation: Available Selectors Statements Mappings.
* Activation: Available Mappings Transforms.
* API: Core Model Bus Topics: Contexts, Kinds Resources.
* API: Core Model Transforms / Mappings Functions.
* API: Dynamic Model Transforms / Mappings Functions.
* API: Template / Facets. Selectors / Predicates. Data (Statement Context).
* API: Command. CQRS (CUD, R): Context (Mapping Context).
* API: Interaction (Behavior Context).
* API: onEvent(Event) : Event.
* Models: Addressing / Encoding / Matching Layer.
* ResourceURNs Occurrences (Subjects):
* (ResourceURN, Resource, Kind, Context);
* (ResourceURN, Resource, Context, Kind);
* (ResourceURN, Context, Kind, Resource);
* (ResourceURN, Context, Resource, Kind);
* (ResourceURN, Kind, Context, Resource);
* (ResourceURN, Kind, Resource, Context);
* Addressing: Model Traversal: MapReduce
* ResourceURNs Contexts, Resources, Occurrences IDs Addressing / Encoding:
* ResourceURN : (ContextResourceURN, SubjectResourceURN, OccurrenceResourceURN);
* Matching: ResourceComparator(s).
* Models: DOM OGM / DCI / DDD / CDI: Restful objects Layer.
* RDF DOM OGM / DCI / DDD / CDI: Sesame Elmo.
* Resources / Kinds / Contexts: DOM / DCI / DDD Subjects, Concepts, Mixins, Behaviors.
* Functional Dataflow API. Model / Events Bus sync.
* DCI: Qi4j / Apache Zest (RDF / KeyValue / EAV EntityProvider).
* Dynamic Functional Contexts: Scala DSL Message Dataflows. Monadic Parser Combinators.
* Behavior / Factory: Roles. Parameterize (domain context) monadic functions applications.

**To do:**

* Merge TOCs.
* Merge Contents.
* Content / Topics:
* Concepts, Design, Architecture.
* Add bibliography / tools use cases / components. Bookmarks, Lectures. Notes: Scrapbook.
* Concepts:
* Data, Information, Knowledge.
* Data, Schema, Behavior.
* Models: Layers Message IO Dataflow Bus.
* Models: Sets Contexts, Kinds, Resources Layer.
  + Events Sourcing / Models Bus IO.
  + Input Message Augmentations:
  + Aggregation: Populate Sets.
  + Alignment: Addressing / Encoding / Matching.
  + Activation: Dataflow Bus Subscriptions. Events Signatures: DCI Contexts (Command), Data (Templates / Facets), Interactions. Reactive IO Models Interactions.
  + Activation: Events Streams (pub / sub) producers / consumers match / dispatch by Activation bound Events dispatch (Interactions).
  + Functional Activation Dataflow API:
    - Context Monad
    - Kind Monad
    - Resource Monad
    - Message Monad
    - Event Monad
    - API: Core Model Streams (Data), Mappings (Schema), Transforms (Behavior) Functions.
    - API: Template / Facets. Data (case match).
    - API: Command. CQRS (CUD, R): Context.
    - API: Interaction.
* Models: Addressing / Encoding / Matching Layer
  + ResourceURNs Occurrences (Subjects):
  + (ResourceURN, Resource, Kind, Context);
  + (ResourceURN, Resource, Context, Kind);
  + (ResourceURN, Context, Kind, Resource);
  + (ResourceURN, Context, Resource, Kind);
  + (ResourceURN, Kind, Context, Resource);
  + (ResourceURN, Kind, Resource, Context);
  + Addressing: Model Traversal: MapReduce
  + ResourceURNs Contexts, Resources, Occurrences IDs Addressing / Encoding:
  + ResourceURN : (ContextResourceURN, SubjectResourceURN, OccurrenceResourceURN);
  + Matching: ResourceComparator(s).
* Models: DOM OGM / DCI / DDD / CDI: Restful objects Layer.
* RDF DOM OGM / DCI / DDD / CDI: Sesame Elmo.
  + Resources / Kinds / Contexts: DOM / DCI / DDD Subjects, Concepts, Mixins, Behaviors.
  + Functional Dataflow API. Model / Events Bus sync.
* DCI: Qi4j / Apache Zest (RDF / KeyValue / EAV EntityProvider).
  + Dynamic Functional Contexts: Scala DSL Message Dataflows. Monadic Parser Combinators.
  + Behavior / Factory: Roles. Parameterize (domain context) monadic functions applications.
* Augmentation: Layers (occurrences, aggregations). Quads.
* Upper Ontology Roles Aligned Object Models. Templates.
* Roles: Metaclass, Class, Instance: Resource Occurrence, Resource, Kind, Context.
* Meta Model:
* Object Model:
* Labeled Property Graph.
* Serialization (Aligned Quads):
* Occurrence: (Context, Object, Concept, Value);
* Occurring: (Object, Context, Concept, Value);
* (Context : Concept type / label, Object, Concept, Value) DOM Property Graph.
* (Object, Context : Concept type / label, Concept, Value) DOM Property Graph.
* Layers APIs: Connector Bus. Templates, Forms Meta Model Dataflow
* Encoding:
* Object Model:
* Serialization (Aligned Quads):
* (URN, Context, Kind, Resource);
* Layer APIs: Addressing, Matching, Persistence.
* Augmentation:
* Object Model:
* Sets CSPO Model.
* Serialization (Aligned Quads):
* Sets CSPO Model Statements encoding.
* Layer APIs: Aggregation, Alignment, Activation. Sets Functional Dataflow Augmentations.
* Data (Data): key / value. Column: (price: 100). Data Aggregation Augmentation.
* Information (Schema): Record (keys / values relation): (price: 100, brand: ACME). Schema Alignment Augmentation.
* Knowledge (Behavior): Records (columns values relationship). Price variation behavior example: ((price: 100, brand: ACME, date: today, priceVariation: 0), (price: 110, brand: ACME, date: yesterday, priceVariation: 10)). Activation Augmentation: materialize relationships / facts.
* DDD REST HATEOAS DOM:
* Object Model:
* ID: Object Occurrence.
* Object (ID, Type, Member\*); Node.
* Type : Object;
* Member : Object; Arc (Property Graph).
* Serialization (Aligned Quads):
* (Object, ID, Type, Member\*);
* Layer APIs: Naming, Registry, Index. DOM Functional Dataflow.
* Messages: Dynamic Object Model Functional Monads bound (kinds signatures subscriptions) Functions. Contexts (Data, Schema, Behavior) browse traversal / transform. Resource aggregates Messages.
* (Resource, Transform, Mapping, Statement);
* Connector Bus API. Messages.
* Layers Dataflow Layout. Messages.
* Templates: Activation. Messages.