LivCos

Cosmos Requirements

Table of Contents

1 Structures

The Cosmos should be able to store all kind of data and it should allow to manage this data over multiple interfaces.

The data should be manageable in it's best suitable structure.

1.1 List

Arrays, (double) linked lists, unordered lists, sorted lists, sets, sorted sets, maps.

Addressable by index, pointer (memory) or key. Iterators.

1.2 Table

Tables, matrices, cubes, n-cubes.

Addressable by coordinates.

1.3 Hierarchy

Hierarchical trees, binary trees.

Addressable by ancestor path. Tree walker.

1.4 Network

Networks, graphs.

Addressable by routes.

1.5 Triplets

RDF subject-predicate-object triplets.

Addressable by subject URIs (groups of triplets) or separate ID.

2 Search

We should be able to instantly search the Cosmos in various contexts.

2.1 Outline Search

Search the cosmos objects.

2.2 Full Text Content Search

Search words in every peace of content. When only word fragments are provided, offer a word-auto-complete feature or search with several words, closely matching the fragment.

2.3 Entity Type Search

Filter search results by entity type. Also allow to search for entity type definitions.

2.4 History Search

Also search the data history. Fit her the results for requests of a specific history time range.

3 Cache

Some data is often read and rarely written. To improve read access performance the data can be held in fast-accessible memory. This cache memory can be filled with the first read or by a separate process. Changes to the original data could auto-propagate into the cache or simply invalidate the cache.

3.1 Search Index

Search indexes are special "caches" of the data to improve search performance. These caches contain an incomplete snapshot of the data, optimized for certain search scenarios.

A search index could be seen as a special collection of triplets, sharing all the same predicate.