

HIISet Metadata Store

This application serves as a Proof of Concept (POC) demonstrating the practical application and benefits of utilizing HyperLogLog Sets (HIISets), as introduced in the previous post HyperLogLog based approximation for very big datasets.

The HIISet Metadata Store (HIISet-meta) is a library that consists of the following files:

1. `lisa_graph.jl`
2. `lisa_hdf5.jl`
3. `lisa_neo4j.jl`
4. `lisa_sets.jl`
5. `lisa_store.jl`

Each of these files serves a specific function within the library, as outlined below.

`lisa_store.jl`

Metadata serves as a detailed description of the primary data, encapsulating various aspects including:

- The data's origin, such as its geographical location;
- The type of data (e.g., tabular data, documents, images, streams, videos, sound, etc.);
- The data encoding;
- Access credentials (such as user ID, password, etc.);
- A log of any modifications made to the original dataset, along with details of those changes;
- Other specific details that provide additional context in unique scenarios.

In the HIISet-meta framework, metadata is structured in a graph format with nodes representing the metadata and edges indicating the connections between nodes. The `lisa_graph.jl` module is specifically designed to handle this Graph data structure. The structure is stored in two main tables within an SQLite database.

1. Nodes Table (**nodes**): This table includes several fields:
 - a. ``sha1``: A SHA1 hash serving as the node's unique identifier.
 - b. ``labels``: Graph labels assigned to the node.
 - c. ``d_sha1``: Another SHA1 hash representing the identifier for the HIISet representation of the original dataset.
 - d. ``dataset``: A compressed dump of the HIISet.
 - e. ``props``: A dictionary for storing additional attributes.
2. Edges Table (**edges**): This table captures the relationships between nodes with fields such as:
 - a. ``source``: The SHA1 hash of the source node in the relationship.
 - b. ``target``: The SHA1 hash of the target node.

- c. ``r_type``: A label denoting the type of relationship.
- d. ``props``: A dictionary for additional properties.


Data is constantly changing, and metadata is always evolving to keep up with these changes.

These modifications effectively chronicle the evolution of the original data, akin to how version control systems like Git manage changes. In Git, change history is preserved through snapshots taken at each **commit**, with the latest version of a document residing in the working directory until it is committed to the repository.

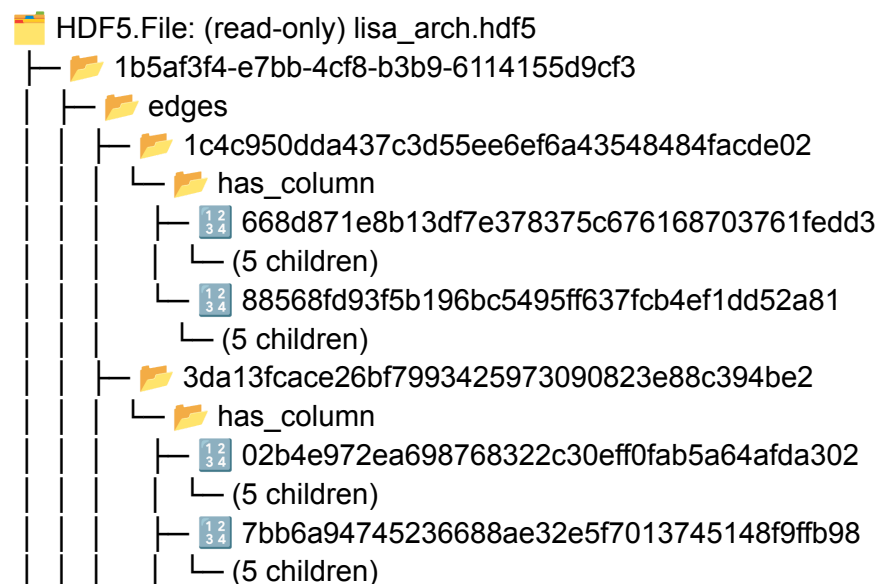
A similar strategy is employed for managing metadata in HIISet-meta. Initially, fresh metadata is gathered into temporary ``t_nodes`` and ``t_edges`` tables. Upon completing the processing, this data is committed to the permanent ``nodes`` and ``edges`` tables.

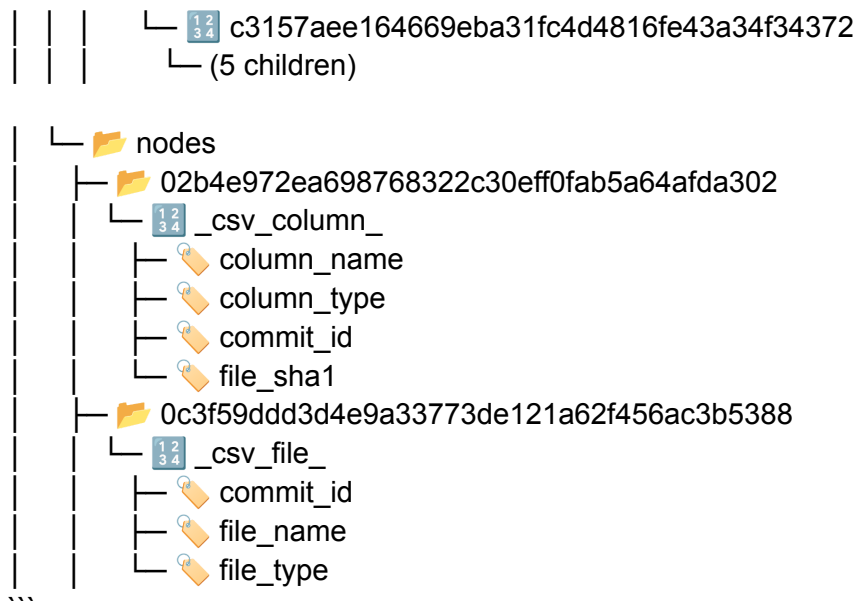
During the commit process, it may be discovered that a node or edge already exists. In such cases, the new and existing versions are compared. If they are identical, no action is taken. If they differ, the new version replaces the old, with the outdated version being archived, ensuring the metadata reflects the most current state of the data.

We are using HDF5 as an archive storage. The `lisa_hdf5.jl` is specifically designed to handle all operations related to managing HDF5 archive files. Here is an example of archived **nodes** and **edges** in the HDF5 file.

In the provided HDF5 layout the top folder  1b5af3f4-e7bb-4cf8-b3b9-6114155d9cf3 is the time based commit ID. All nested folders hold metadata for all nodes and edges submitted in this commit.

Detailed information about commit is in **commit** SQLiteDB table.





Check `lisa store.ipynb` for more details about using `lisa store.il`. [1]

lisa neo4j.jl

The file ``julia_neo4j.ipynb`` [2] serves as a practical demonstration of how to leverage the ``lisa_neo4j.jl`` file. In this example, we tackle a straightforward scenario comprising several key steps:

1. We initiate a search within the SQLiteDB table **tokens**, which functions as an inverted index. This index encompasses all the terms extracted from the processed datasets, linked back to their original nodes. The objective here is for the query to fetch all nodes containing the queried tokens.
2. Following the retrieval of nodes and edges, we embark on constructing a Neo4J Graph. This step involves organizing the collected data into a structured graph format that illustrates the relationships and connections between different nodes and edges.
3. An additional measure involves establishing relationships between similar CSV files. To achieve this, we employ the Jaccard index to gauge the similarity between two HIIsets. This approach serves as an alternative to directly comparing the original files, which may be too large or inaccessible in some instances. The Jaccard index thus provides a practical means of measuring file similarity without needing to access the files themselves.

Included below are visual snapshots of the Neo4J Graph as it appears after completing step 2, offering a graphical representation of the data relationships and structures formed as a result of these processes.



Here is a visual representation of CSV files that share similarities in the graph.



Summary

The document introduces the HIISet Metadata Store, a library that utilizes HyperLogLog Sets (HIISets) for managing metadata in a structured graph format. The library includes modules for

handling metadata, graph data structures, and archive storage. It details the process of managing metadata changes, similar to version control systems like Git, and demonstrates how to leverage the library for constructing Neo4J Graphs and measuring file similarity using the Jaccard index. The document also provides visual representations of the data relationships created in the Neo4J Graph. The references include links to detailed information on using the `lisa_store.jl` and `lisa_neo4j.jl` files.

References

1. https://github.com/alexmy21/lisa_meta/blob/main/lisa_store.ipynb
2. https://github.com/alexmy21/lisa_meta/blob/main/lisa_neo4j.ipynb
3. <https://www.thebusinessresearchcompany.com/report/metadata-management-tools-global-market-report>
4. <https://download.memgraph.com/asset/whitepapers/quick-start-guide-to-migrating-from-neo4j-to-memgraph.pdf>
- 5.