

Neo4j Storage Structure

Nico-Alexei Hein
it12125@lehre.dhbw-stuttgart.de

March 11, 2015

Abstract

This is a rough explanation of Neo4j internals with a focus on disk storage and file layout.

1 Graph representation of the Data

The simple sample graph displayed in figure 1 shows a subgraph of the Neo4j example. This human readable representation is now used to explain how Neo4j stores this graph.

2 Property Records

As a first step we take a look at the data stored in this graph. In Neo4j it is possible to store properties for every node and relationship (labels are ignored ignored at the moment). To store the nodes and relationships itself in fixed size records (for faster access etc.) linked lists are used. In figure 2 the property records are displayed.

Every node and every relationship references its first property record. The property records themselves can be seen as a double linked list with a key - value store. In figure 8 the representation on byte-level is shown. Node, relationship and property records are storing pointer to the next property in the last 4 bytes (one integer in Java). Bytes 18 to 21 of property record are containing a pointer to the previous record.

3 Node Records

The nodes itself as well as relationships are stored in fixed size records. However a node needs to know all relationships it is involved in. Therefor again a linked list is used. Every node references its first relationship shown in figure 3 and in figure 8 the deep blue marked bytes. Now the complete node record was covered and before we are focusing on the double linked relationship lists the relationship record is explained.

4 Relationship Records

As could have been observed above the node record only stores a pointer to a property and a relationship but not the first adjacent. Therefore we change our perspective to the graph and are focusing on relationships. Instead of nodes referencing adjacent nodes we have nodes referencing the first relationship which references the start-node and the end-node of the relationship (Grey filled bytes in figure 8). This results in the abstract graph shown in figure 4.

The relationship type (Blue in figure 4) references a relationship type which again references a string (with the name) in the dynamic store. (This avoids redundancy in storage - the cache works in a different way.)

The four green integers of the relationship record (figure 8) are used to realize the mentioned double linked relationship lists.

Since every relationship is involved into two double linked lists (the one of the start-node and the one of the end-node) four pointers need to be stored. In figure 4. They are shown as:

- **SP** Start-Node Previous Relationship
- **EP** End-Node Previous Relationship
- **SN** Start-Node Next Relationship
- **EN** End-Node Next Relationship

In Figure 5 only the SP is set. In the following list some pointers are explained:

- 'Frisch - has written - HomoFaber' has no previous relationship since Frisch references it as its first relationship.
- 'Frisch - has written - Montauk' has 'Frisch - has written - HomoFaber' as previous relationship.
- 'Suhrkamp - has published - Steppenwolf' has 'Suhrkamp - has published - Montauk' which has 'Suhrkamp - has published - HomoFaber' which has no previous relationship.
- ...

In Figure 6 the links for the end-node previous relationship are added analogue to start-node previous relationship.

Now in figure 7 all double linked lists are complete. This is how the graph displayed in figure 1 would be stored.

5 Sources

- Slides from <http://slideshare.net/thobe/an-overview-of-neo4j-internals> by Tobias Lindaaker (assumed to be reliable since <http://www.neo4j.org/develop/internals> refers to these slides)
- <http://digitalstain.blogspot.de/2010/10/neo4j-internals-file-storage.html> by Chris Gioran

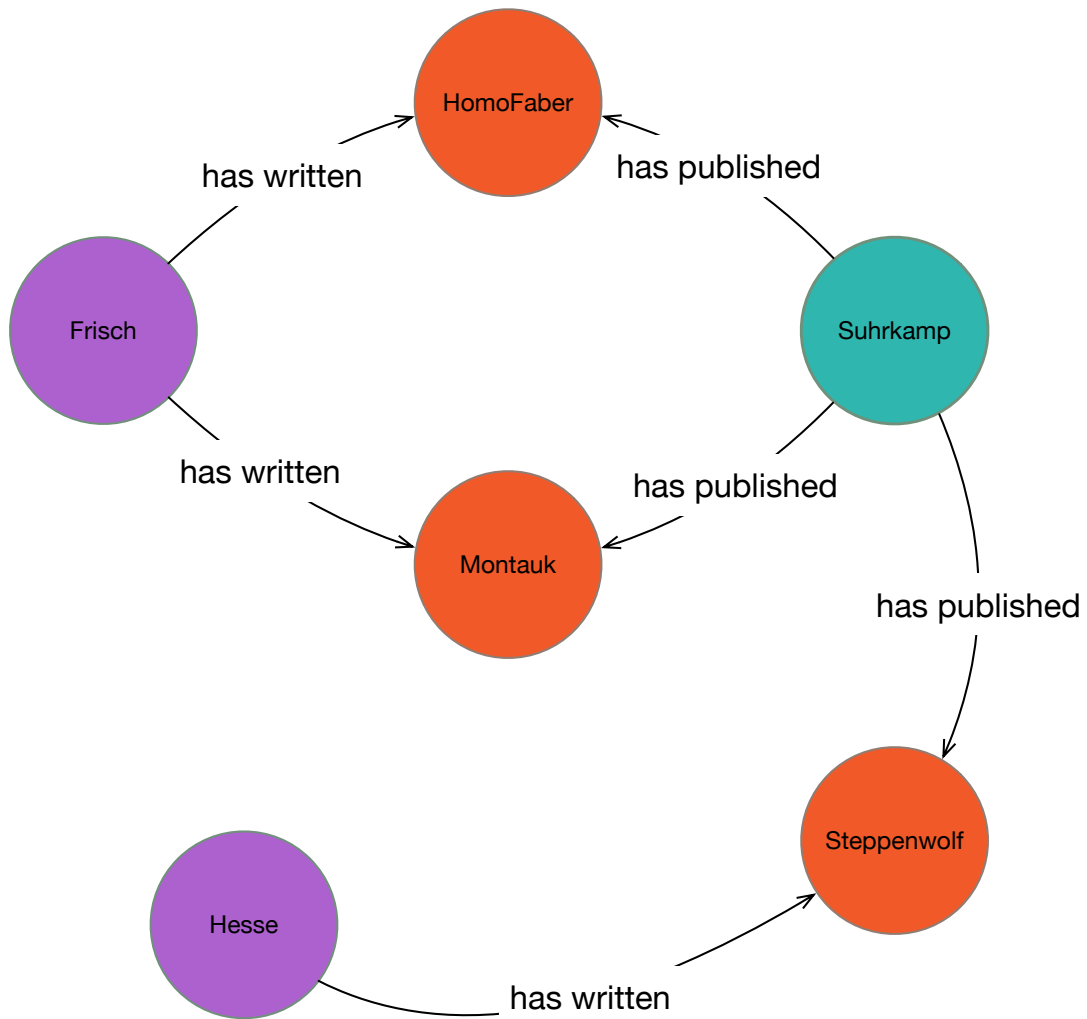


Figure 1: Simple sample Graph

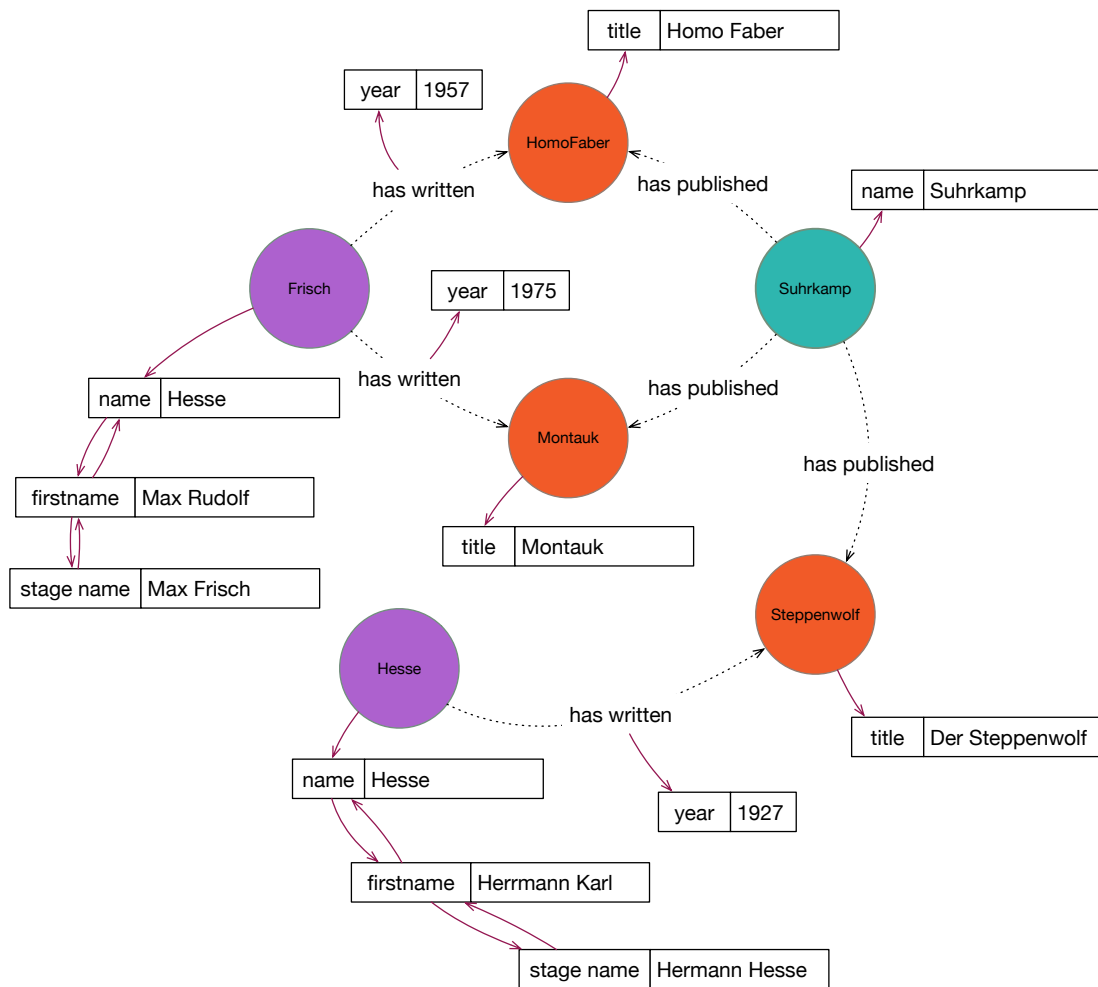


Figure 2: Property Records



Figure 3: First Relationship

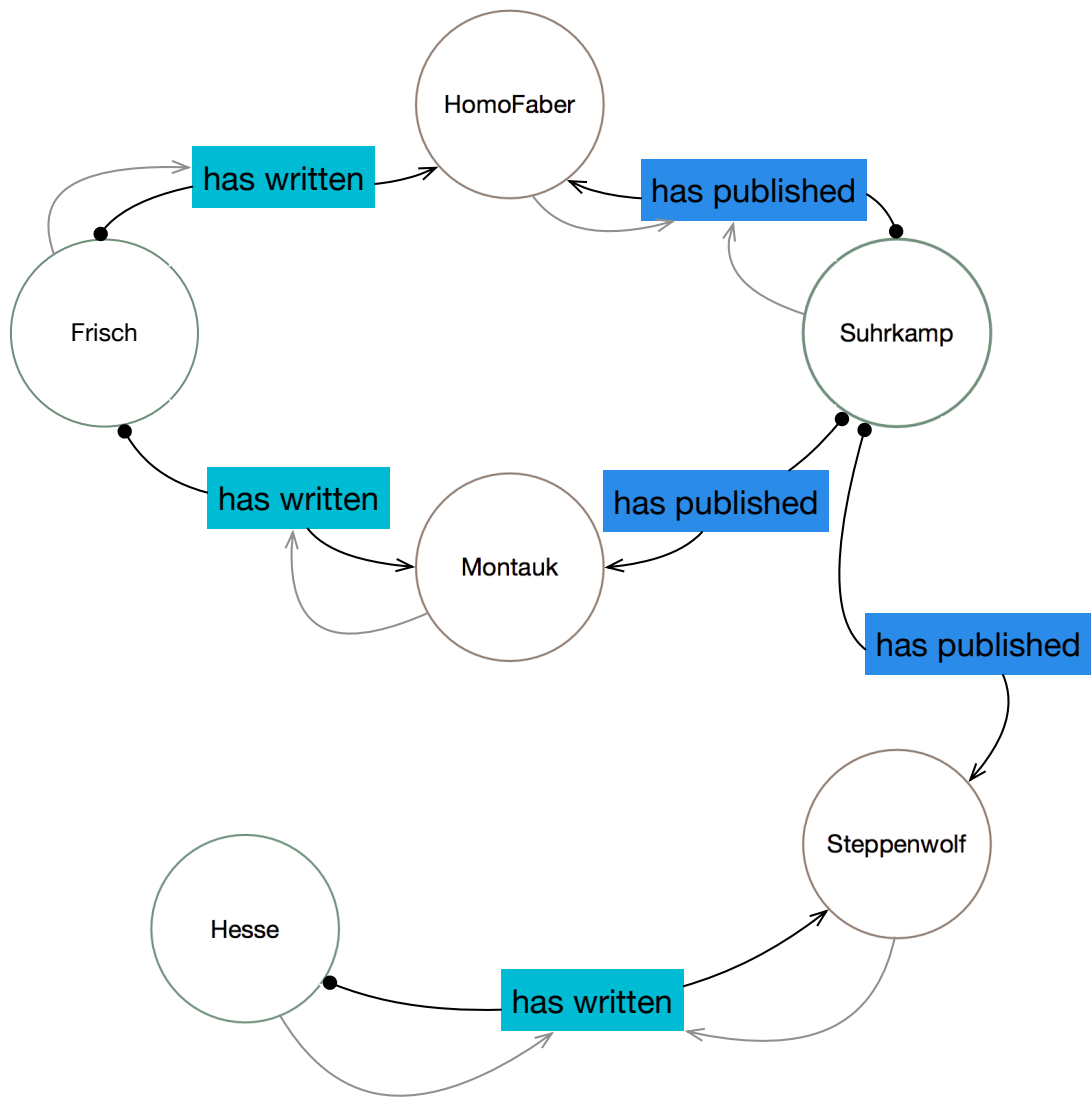


Figure 4: Start Node - End Node

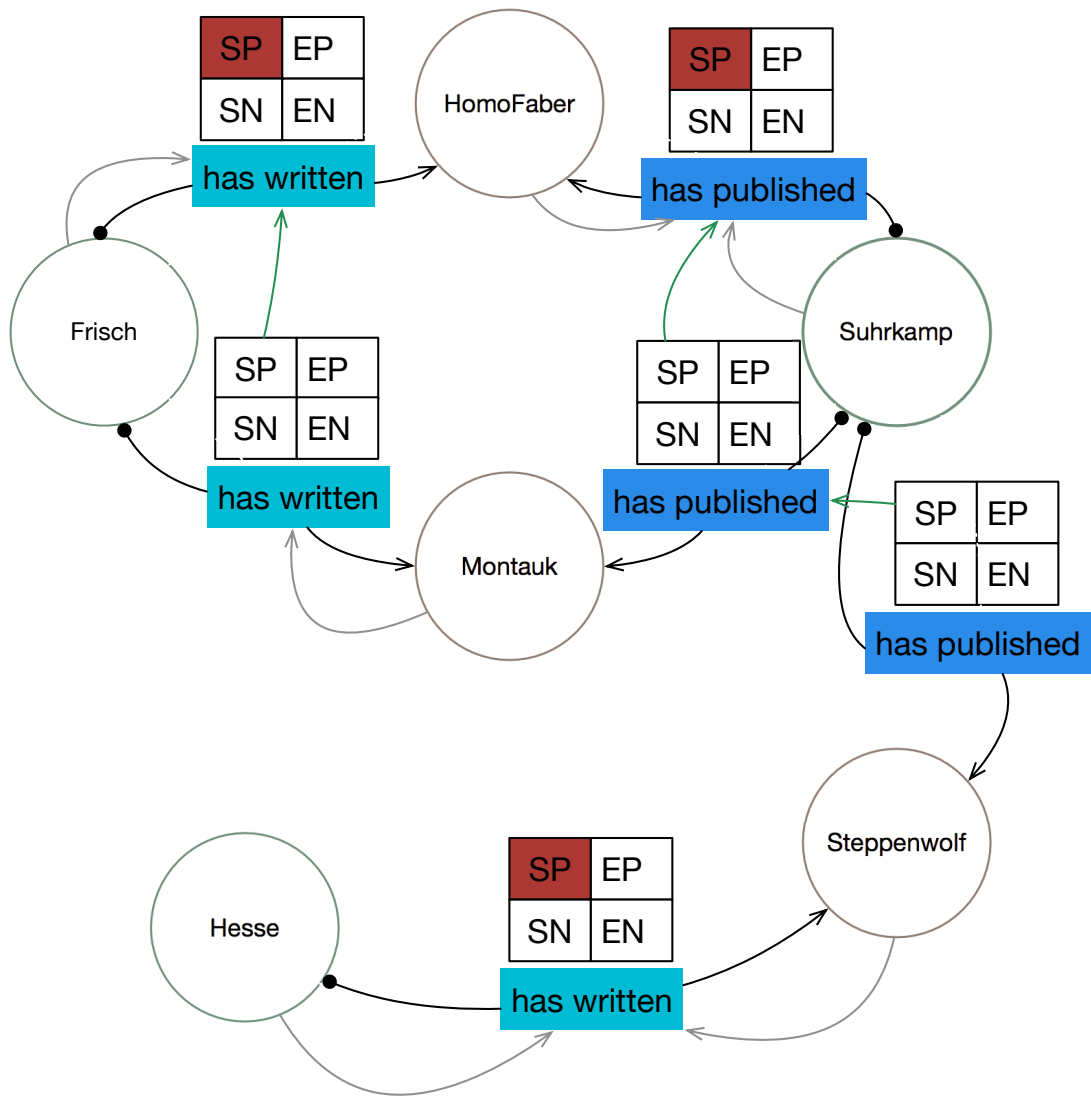


Figure 5: Start Node Previous Relationship

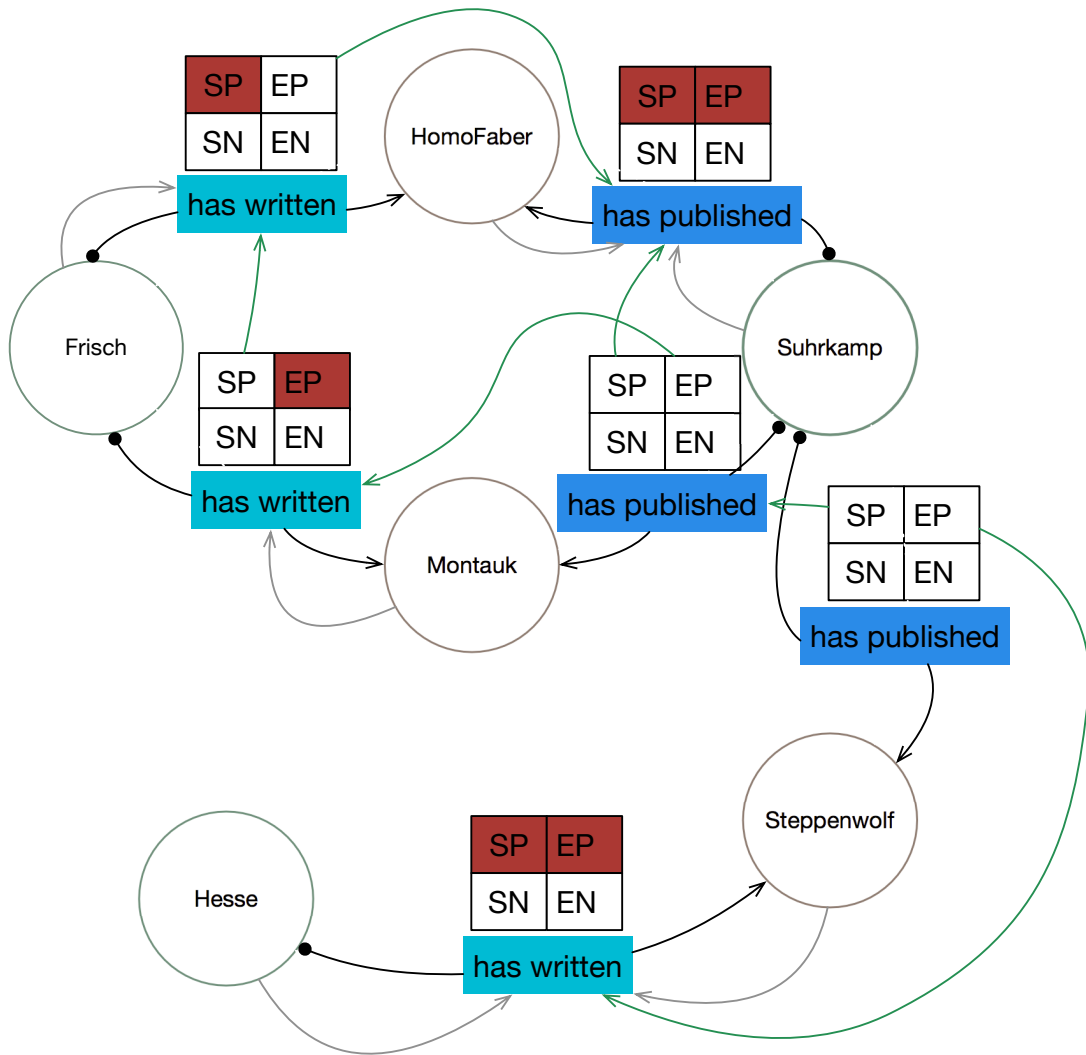


Figure 6: End Node Previous Relationship

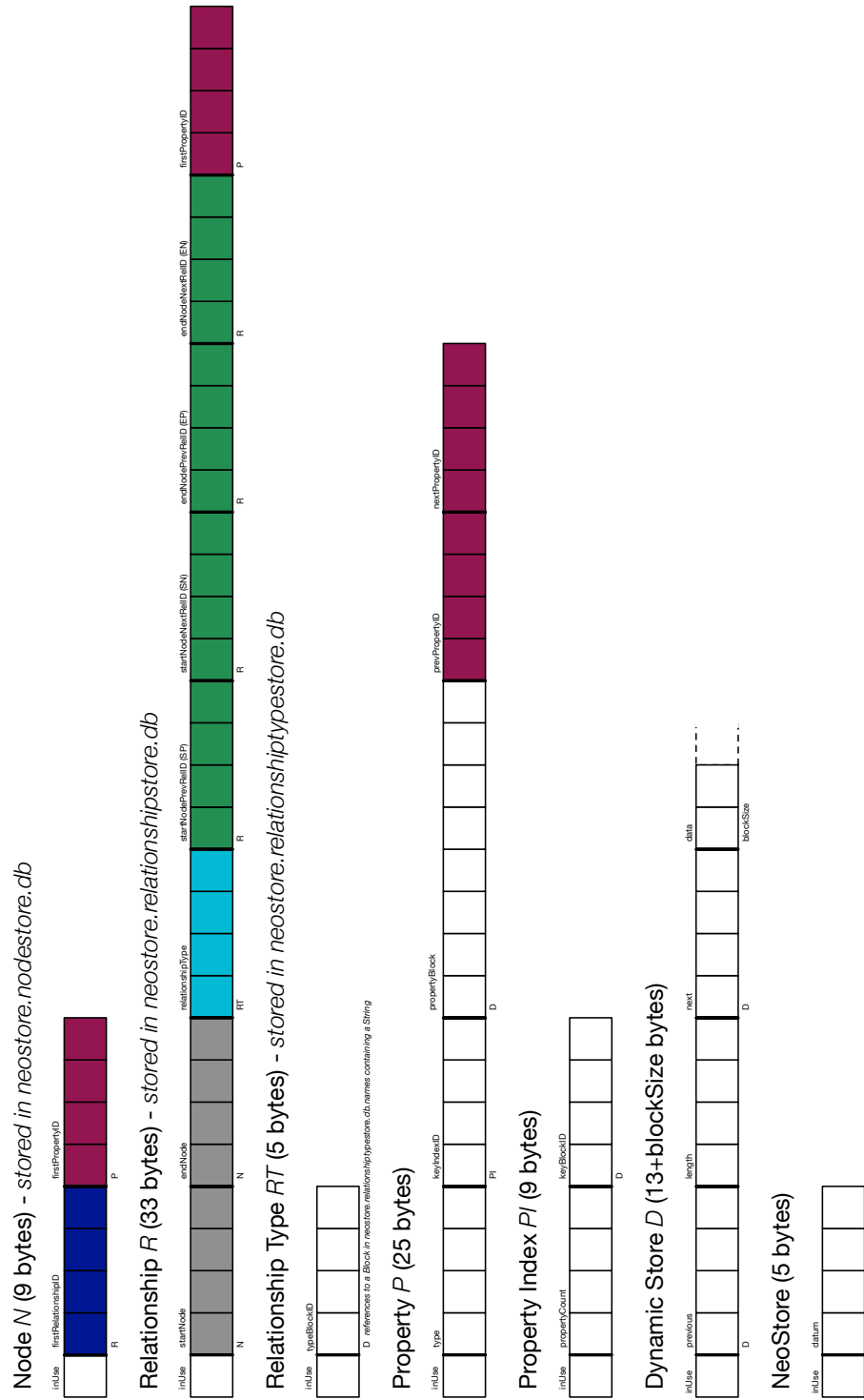


Figure 8: NeoBytes