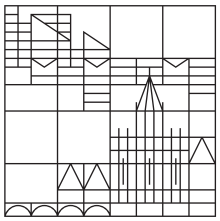
Graph Record Layout Research Environment

A graph database with some layout tools & methods

Fabian Klopfer

Databases and Information Systems Group
Department of Computer Science
University of Konstanz

Universität Konstanz



Abstract:
The here described software is focussed on finding new methods to layout the records of a graph database such that the least possible accesses are necessary to satisfy a certain benchmark. A benchmark is here usualy a set of queries, for example traversal-based queries. The current state-of-the-art methods use the graph structure to statically layout the graph on disk using clustering, partitioning or community detection methods to form blocks along with some ordering algorithm. The methods that are to be explored are dynamic — either in terms of the query or in terms of a changing database. This document provides a high-level user guide to the structure of the database and the methods and tools that are implemented. Furthermore the specification as well as a more extensive design document are captured, along with some benchmark and visualization techniques.

Contents

Chapter 1

User Guide

1.1 Introduction

1.2 Database Architecture

1.2.1 Overview

1.2.2 Records

1.2.3 Access

1.2.4 IO

1.2.5 Cache

1.2.6 Query

1.2.7 Import

1.2.8 Layout

1.2.9 Visualization

1.2.10 Building, Tests, Coverage

1.3 Record Layout Methods

1.3.1 Theoretical Aspects

1.3.2 Static Layout

1.3.3 Dynamic Layout

1.4 Conclusion

1.4.1 Summary

1.4.2 Future Work

Chapter 2

Technical Documentation

2.1 Software Requirement Specification

2.1.1 Introduction

Purpose

The purpose of this Software Requirements Specification is to describe the features, constraints and demands of a research environment for the optimal layout of graph records on disk in detail. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Dr. Theodoros Chondrogiannis, the supervising postdoctoral researcher.

Scope

This Software system shall implement a graph record layout research environment, that consists of a graph database along with tools, that rearranges the recrods of the database based on a predefined format, measure the number of disk accesses needed to service a certain query and that provide other layouts to compare against.

Definitions, Acronyms, Abbreviations

word	shortform	meaning
database	db	a software system to store and alter data in an organized manner.
Operating System	os	An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.
Portable Operating System Interface	POSIX	A specification for a set of OSes that covers for example Linux, macOS and BSD-style operating systems.
C Programming Language	C	a programming language.
Input/Output	IO	the notion of loading and storing information to media other than RAM and CPU registers. In this document hard drive and solid state disk are meant primarily.
Create, read, update, delete	CRUD	The basic database operations, that allow to create, read, update and delete a record.
Databases and Information Systems	DBIS	The name of the group at the university of Konstanz, at which the software system is build.
Stanford Network Analytics Project	SNAP	A porject of the university of stanford that hosts many large scale graph data sets.
Least Recently Used	LRU	A strategy when evicting pages from a pool of memory.

Breadth-first Search	BFS	A graph traversal scheme, where all neighbours of the current node are visited before continuing with the next node.
Depth-first Search	DFS	A graph traversal scheme, where the next node is considered before visiting all neighbours of the current node.
Single Source Shortest Path	SSSP	The problem of finding the shortest path to all nodes in a graph from one source node.
...	...	...

Overview

In the second chapter several conditions, assumptions and circumstances will be mentioned, that help charachterizing the software's special use case. In the thrid chapter the concrete requirements are listed.

2.1.2 Overall description

Product Perspective

The product shall rely on functions of unixoid OSes. That is it uses the interfaces specified by the OpenGroup in the POSIX.1-2017 specification (also called IEEE Std 1003.1-2017) [1]. There are no relations to other software systems than the operating system during the runtime of the environment.

Product Functions

The product shall support different tasks in graph record layout research:

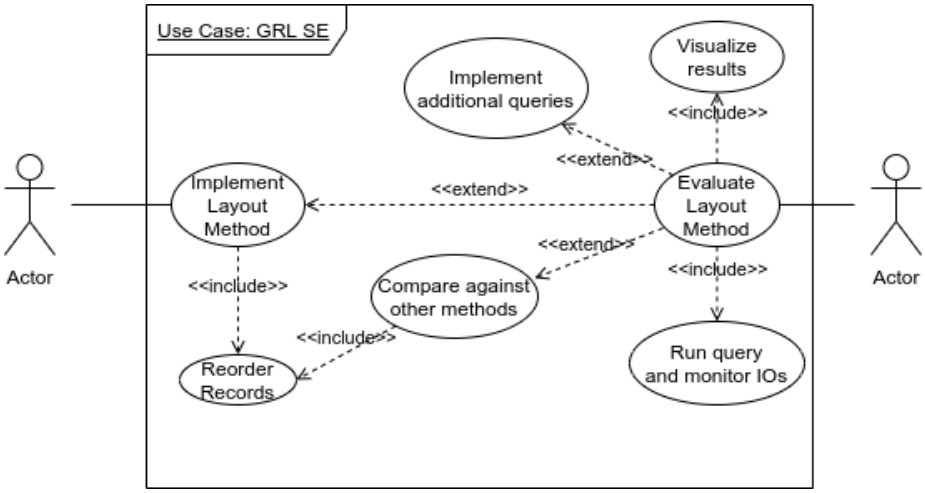
1. Import data into a graph database.
2. Query the database with a certain fixed function.
3. Rearange the record layout on disk with a layout given in a specific format.
4. Monitor the number of disk IOs.
5. Monitor the caching behaviour.
6. Provide interfaces for the implementation of new layout methods.
7. Provide a interfaces to ease the implementation of new queries.

This shall be done using a graph database, that shall save related information to disk and load them into a cache on access, as well as support the common CRUD functionality.

User Characteristics

The potential users are the staff and student assistants of the DBIS group. Therefore standard user will have technical knowldege, and have visited at least a basic course on databases. Futher the users are able to program in C. There are three different types of users of this research environment:

- Researchers: Implements new layout method, benchmarks the result of the method against other layouts, visualizes the result of the benchmark. Should at least know the theory of how records are stored.
- Supervisors: Works on the administration and coordination of the project. Eventually hires other researchers and developers. Also a researcher.
- (Future) Developers: Uses the existing framework to extend the functionality, for example to implement additional queries of a different type, add new features like storing properties and similar things. Needs to know details of database architecture and implementation, more advanced C programming and some software engineering.



Constraints

- The database is expected to be run on a standard notebook and desktop machine, that is it needs to run without momory-related errors on a machine with 8 GiB RAM, no matter the data set size.

Assumptions and Dependencies

The environment supports only POSIX-compliant OSes.

2.1.3 Specific Requirements

External interfaces

The POSIX interfaces.
The data format of some of the SNAP data sets.

2.1.4 Functional requirements

1 Data Storage and IO
The system shall

- 11 store a graph G consisting of nodes and edges (V, E).
- 12 store the records in a file on disk.
- 13 be able to grow and shrink the size of the file that is used.
- 14 be able to read from disk into main memory.
- 15 be able to write back to disk.

2 Data Caching and Memory
The system shall

- 21 not exceed a certain memory limit for both pages from disk and memory requirements from queries.
- 22 split an amount of memory into frames.
- 23 load data from disk into a frame on request.
- 24 maintain information on unused frames.
- 25 maintain a mapping from loaded data to frames.
- 26 evict data from memory when the memory limit is hit
- 27 be able to prefetch by reading more data than required in a neighbourhood

3 Data Access
The system shall

- 31 be able to calculate the location in the file of nodes and relationships efficiently
- 32 keep track of free record slots on disk.
- 33 be able to create, read, update and write nodes.
- 34 be able to create, read, update and write relationships.
- 35 provide an interface to easily retrieve and traverse data.
- 36 provide functions to retrieve common informations about nodes and relationships, like the node degree.
- 37 be able to monitor the numer of overall disk accesses.
- 38 be able to monitor the number of disk accesses that are necessary, if only a certain limited amount of memory is available.
- 39 be able to monitor the cache hit rate.
- 310 be able to monitor the prefetch hit rate.

4 Queries
The system shall

41 implement the most basic traversal schemes — BFS and DFS.

42 implement Dijksta's algorithm for finding all shortest paths from a single source (SSSP).

43 implement the A* algorithm for the shortest path problem.

44 implement the ALT algorithm for the shortest path problem.

45 provide data structures for the results of the above algorithms.

46 implement the Louvain method for community detection.

47 implement a random walk.

5 Data Import
The system shall

51 be able to import a set of standard data sets from the SNAP data set collection.

6 Layout Tools
The system shall

61 provide a function to generate a randomized record layout.

62 provide functions to reorganize the record layout on disk, given updated record IDs.

63 provide a function to sort the incidence list structure after reorganizing the record structure.

64 provide a function to reorganize the relationships given a layout of the vertices.

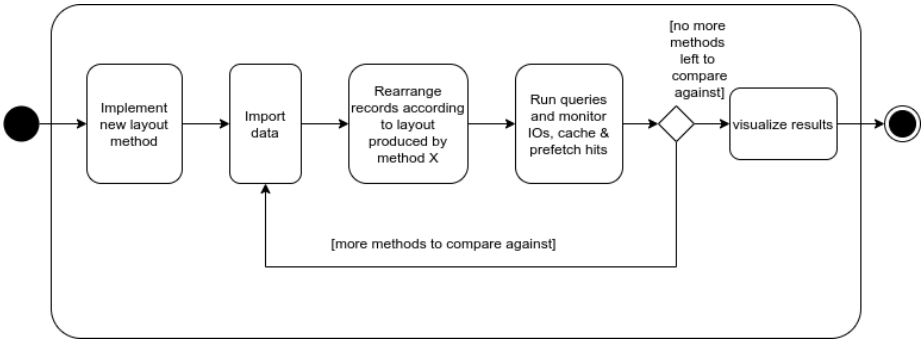
7 Layout Methods
The system shall

71 implement one static and one dynamic history-based layout method for a static database.

72 provide an interface for implementing new layout methods.

8 Visualization Tools
The system shall

81 provide functions to visualize and visually compare the data that was monitored by the caching functionality.



2.1.5 Performance Requirements

1 The system shall work with limited RAM resources even for very large datasets.

2.1.6 Software System Attributes

1 Maintainability

2 Extensibility

3 Correctness

Bibliography

[1] *The Open Group Base Specifications Issue 7, 2018 edition, IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)*. June 1, 2021. URL: https://pubs.opengroup.org/onlinepubs/9699919799/functions/contents.html (visited on 06/01/2021).

2.2 Software Design Document

2.2.1 Introduction

Purpose

This document is written, to document the proposed architecture of a research environment for the optimal layout of graph records on disk. The document is inteded for the developers implementing the system, the persons supervising projects using the final system and the other stakeholders to get an overview of the software system.

Scope

This Software system shall implement a graph record layout research environment, that consists of a graph database along with tools, that rearranges the recrods of the database based on a predefined format, measure the number of disk accesses needed to service a certain query and that provide other layouts to compare against.

Definitions, Acronyms, Abbreviations

word	shortform	meaning
database	db	a software system to store and alter data in an organized manner.
Operating System	os	An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.
Portable Operating System Interface	POSIX	A specification for a set of OSes that covers for example Linux, macOS and BSD-style operating systems.
C Programming Language	C	a programming language.
Input/Output	IO	the notion of loading and storing information to media other than RAM and CPU registers. In this document hard drive and solid state disk are meant primarily.
Create, read, update, delete	CRUD	The basic database operations, that allow to create, read, update and delete a record.
Databases and Information Systems	DBIS	The name of the group at the university of Konstanz, at which the software system is build.
Stanford Network Analytics Project	SNAP	A porject of the university of stanford that hosts many large scale graph data sets.
Least Recently Used	LRU	A strategy when evicting pages from a pool of memory.

Breadth-first Search	BFS	A graph traversal scheme, where all neighbours of the current node are visited before continuing with the next node.
Depth-first Search	DFS	A graph traversal scheme, where the next node is considered before visiting all neighbours of the current node.
Single Source Shortest Path	SSSP	The problem of finding the shortest path to all nodes in a graph from one source node.
...	...	...

References

1. SRS Document
2. Neo4J record structure documentation

Overview

The proposed architecture will be shown from a static point of view, e.g. the decomposition into subsystems and their structure. Then the dynamic aspects of the system are following. This contains the description of interfaces between components, the behavior of the system and information flow. Finally, used design patterns are mentioned.

2.2.2 Software Architecture

Overview

The graph record layout research environment consists of a database and a set of functions to assist in reordering records, measuring the number of IOs of a certain layout, visualizing the results, queries with certain access patterns, and an interface to implement new layout methods easily into the environment.

IO

Cache

Access

Queries

Import

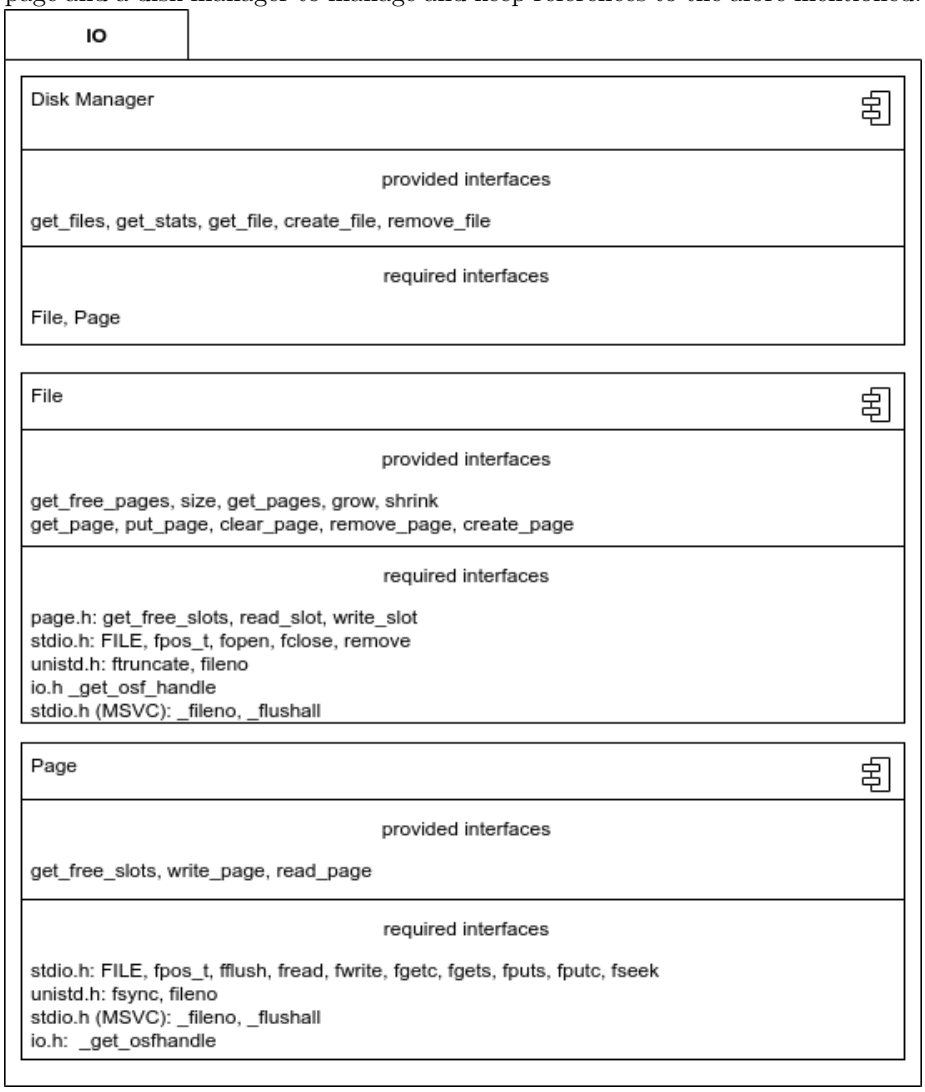
Layout Utilities

Layout Methods

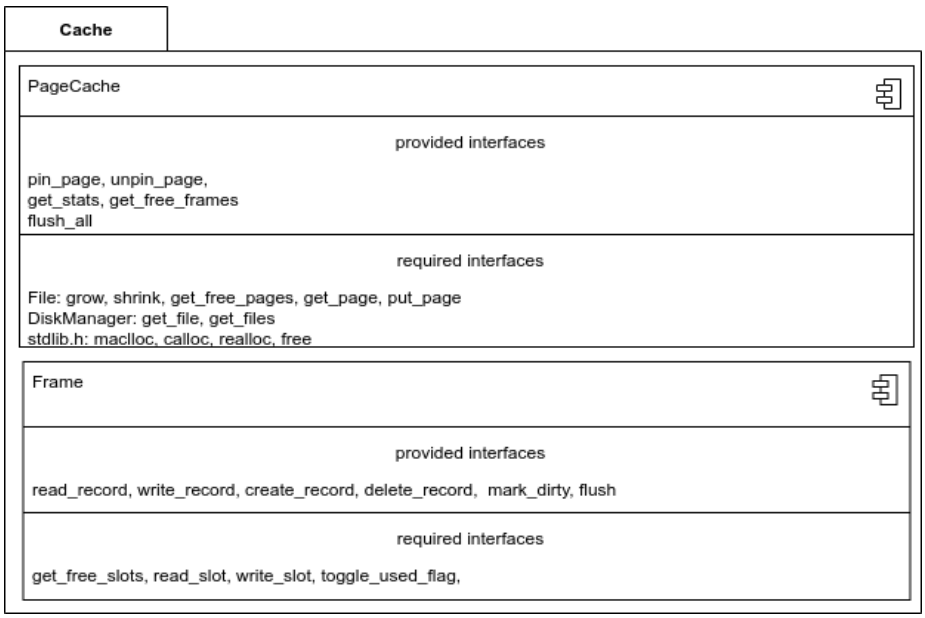
Visualization Utilities

Subsystem decomposition

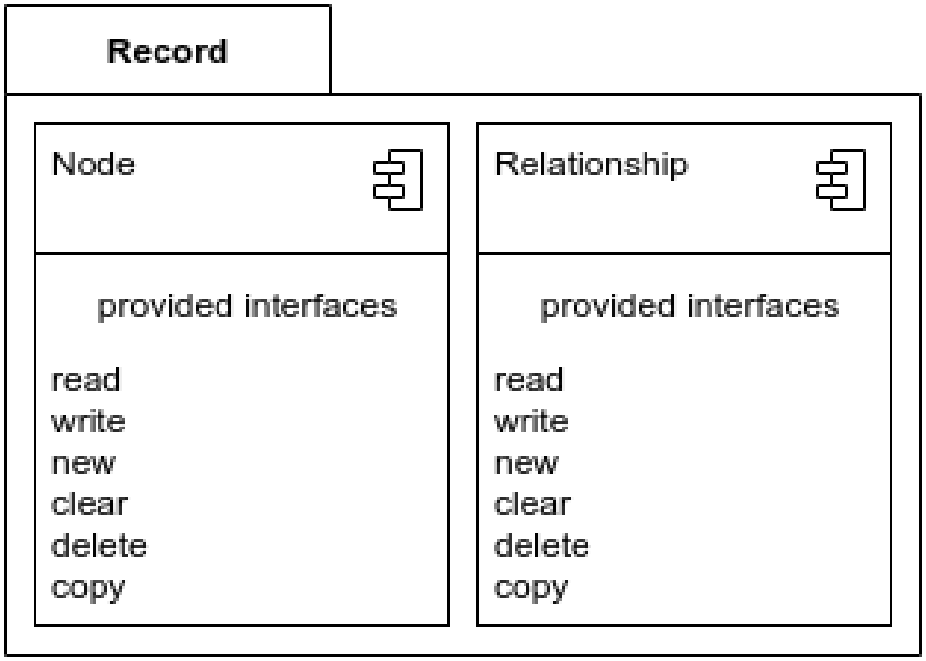
IO The storage module shall provide IO methods, the concepts of a file, a page and a disk manager to manage and keep references to the afore mentioned.



Cache This module shall provide the simplest for of a n page cache. Further it shall provide a basic one page buffer manager, i.e. implement the notions of a frame and the cache/buffer manager itself.

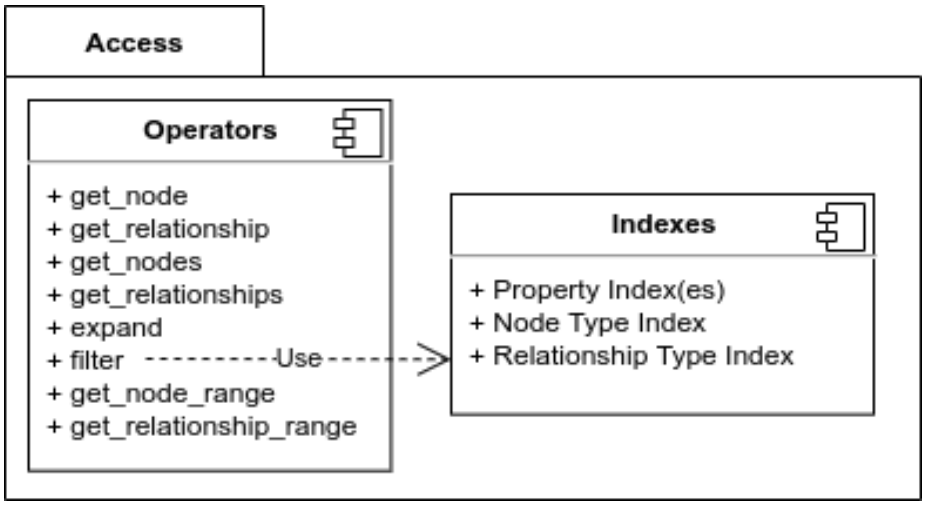


Record Structures Finally the layout of the records and the structs to keep them in memory shall be described. Read and write operations shall be specified by these structs and contained as function pointers. The records must follow the index free adjacency list schema used in Neo4J. This directory contains the structs and read/write functions of nodes and relationships



Access This module shall provide access operators for the database. That is it shall provide the following operators:

- createNode
- createRelationship
- getNode
- getRelationship
- get Nodes
- getRelationships
- getNodesInRange
- getRelationshipsInRange
- Expand
- filter



Queries Further is shall provides functions that use the other modules in order to generate access patterns, that is sequences of records and thus also disk blocks retreived from disk. The first query to be considered is breadth first search. Additionally Dijkstras algorithm and the A* algorithm are planned to be considered. Another class of possible queries to consider are random walks.

Import

Layout Utilities

Visualization Utilities

Layout Method Interface

```
Interface MessagingService {
  public void sendMessage(string sender,
      string accesstoken, string recipient, Message message) {
    require accessValid(username, accessToken);
    require messageHandle != null;
    require message.length > 0;
    ensure getMessages(recipient, database.getUser(recipient)
        .getAccessToken()).contains(message);
  }

  public Message[] messages getMessages(string username, string accessToken) {
    require accessValid(username, accessToken);
    require messageHandle != null;
    ensure messages != null;
  }

  public void messageRead(String username, string accessToken Message message) {
    require accessValid(username, accessToken);
    require message != null;
    require getMessages(username, accessToken).contains(message);
    ensure !getMessages(username, accessToken).contains(message);
  }
}
```

2.3 Benchmarks & Visualizations

2.3.1 Benchmarks

Traversal-based Queries

2.3.2 Visualizations

Total Accesses per Query per Method

Access sequence