GRAPHMK – AN INTEGRATED FRAMEWORK FOR GRAPH COMPUTATION

Nageswaran Keshan

(189330G)

Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science

Department of Computer Science

University of Moratuwa
Sri Lanka

April 2021

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Full Name of Student: Nageswaran Keshan

Registration No: 189330G

Signature: ***UOM Verified Signature*** Date: 11/11/2021

The supervisor/s should certify the thesis/dissertation with the following declaration.

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the CS6997 MSc Research Project.

Supervisor Name: Dr. Indika Perera

Signature: …… Date: 11/11/2021

Abstract

The rapid change in technological innovation and end user expectation during the past decade resulted in data being an important area for new research and development. In addition, evolvement of internet of things, social graph and communication networks resulted in creation of large-scale real time, rapidly changing datasets. This led to the creation of graph systems supporting various graph analytics to become popular. Meanwhile, relational databases are used in storing and managing data including most advanced systems where graph analytics was never explored on top of a relational schema-based system.

This led to the questions whether graph analytics can be done in a relational environment whilst it is still being blindsided or does relational databases even have limitations to execute graph computations. The relational model where data needs to be imagined in a tabular format rather than a graph format with edges and nodes are inefficient in executing iterative graph analysis. Relational systems will end up creating expensive joints in order to execute such computations. Structured Query Language (SQL) queries are difficult in nature to express graph analysis,though we have downsided, relational systems that are composed of great functionalities which includes fault tolerance, integrity constraints, secure transaction and query optimization etc.

This thesis writeup reflects an integrated framework for graph analytics that comprise of three main components. Firstly, a data model that is capable of executing graph computation within a relational environment. Secondly, a query language which can be considered as a data log that helps to execute graph specific computations. Finally, a bolt-on solution which comprise of the above two, that executes sitting within a relational query engine and executes queries created from introduced data log language. The tests performed are evident that bolt-on solution introduced achieved better performances in provided scenarios.

Subject Descriptors

- **Graph Theory - Graph Problems, Graph specific algorithms**
- **Logical Design – Models & Schema**
- **Languages – Data log & Query**

Keywords

Bolt-on systems, Graph computation, Graph Algorithms and Iterative computation

Table of Contents

1 Introduction

Contents

1.1 Overview of the Chapter

This section will provide a history to the initiative and explain the need for Graph data and processing in today's world of Big data. Furthermore, the current available options, previous work, their strengths and limitations, as well as aims and objectives will be addressed. The Chapter will also include characteristics of the design, deliverables of the project, hardware and system specifications.

1.2 Motivation & Background Context

The modern era is known as the "Big Data" age, where an infinite number of sources produce large amounts of unstructured, semi-structured and structured data on a regular basis. Big Data is difficult to handle and necessitates massively parallel systems that run on many machines. “Every day around the world, unstructured data sources such as sensors, social media posts, and digital images generate 2.5 quintillion bytes of data.” [9]. Graphs are used to describe a variety of real-world artifacts, including social networks, web graphs, chemical compounds, and biological structures. Graph data management is needed in many real-world applications such information engineering, bioinformatics [60], chemistry [54], geography, human resources and bioengineering. Following figure depicts a subgraph of the Twitter online community which illustrates how Graph data is structured. A few examples of graphs that have been used in real-world applications are as follows:

1. An example of a social graph is the Friendship-Graph of the Facebook social network [15].
2. The World Wide Web is a huge hyperlinked repository. Google analyzes this graph as part of the ranking of the websites [38, 49].
3. Study on contact relations may be amalgamated into graphing as well.
4. The protein interaction networks [53] demonstrate how protein interaction occurs.
5. Natural language processing models can be developed using graphs based on co-occurrences in text quantities.

6. The road graph is a spatial graph in this context.
7. Graph problems are prevalent in machine learning and data mining [30].

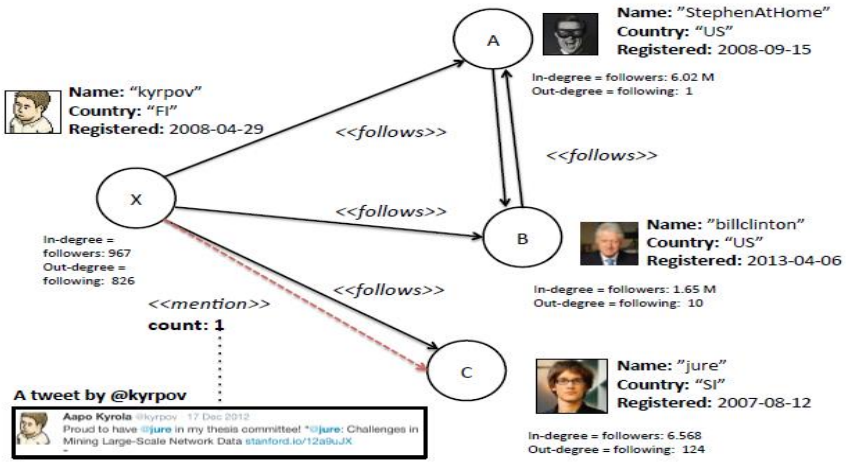


Figure 1.1 The Twitter social graph [27] has a subgraph.

The preceding examples show the significance and necessity of graph data in today's world. It also emphasizes the importance of analyzing these networks in order to discover lessons that can be applied to a variety of organizations. Mostly as a response, many graph systems were built to do graph computation. Pregel [32], Giraph [16], GraphLab [17], Giraph++ [56], NScale [2], AllegroGraph [6], Neo4j [34], and GraphX [17] are just a few of the graph systems that have been built to meet these requirements. Following figure shows two different types of graph structure partitioning methodologies.

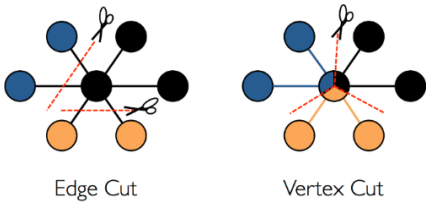


Figure 1.2 Types of partitioning approach. [17]

1.3 Identified Problem Scope

Since most organizations manage and store data in relational databases, these designed graph systems are commonly seen in graph database systems. In order to perform a graph computation,

two systems must be maintained by the organization. The following are the measurements of the regular system of use: (1) Transfer relational database to a text document (csv, txt, or xml) (2) Load the file into the graph system (3) Perform analytical data calculation and obtain results (4) Restore the outcomes to the relational database for any further analysis [3]. This is due to the fact that data analysts must move information between multiple systems, which would be obviously expensive. Learning and managing two separate methods is often challenging for analysts.

Given the limitations in relational databases to do graph analytics, the above-mentioned approach is currently used for all graph analytical activities. When it comes to SQL, it is difficult to interpret graph analytical procedures using it as a query language. Even for a simple procedure like a neighborhood, it has been proven that a query with several expensive joints is necessary, and executing it becomes more complex. Aside from that, running recursive algorithms like PageRank, linked elements, shortest paths, and so on is inefficient even if you write a query with expensive joints on relational data [3]. The graph data, comprises of vertices and edges, as well as attributes, and is accompanied by real-world information. Take, for example, attributes that describe the characteristics of each individual, such as name which can be represented in vertices. Meanwhile, edges will represent the common relationships within individuals to connect them such gender, same age etc. These meta information are normally stored in relational schema need to perform prior ETL to do graph analytics.

Figure 1.3 depicts an example procedure for performing the graph procedure iteratively.

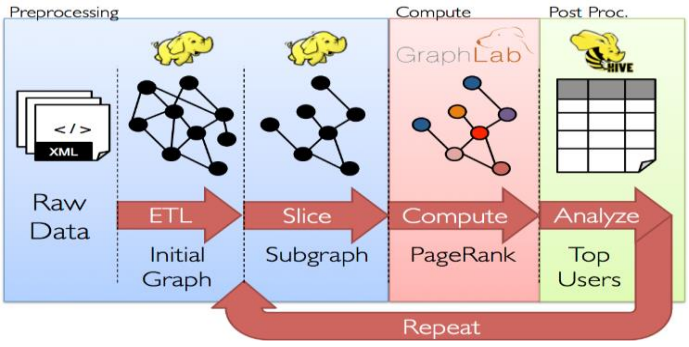


Figure 1.3 In a graph computation, an iterative method [17] is used.

It can therefore be inferred as a query, "Why not guide graph research on the relational database?" This issue would only be true if it has been proved. But even if it is assumed that graph analysis with relational databases is effective, the following benefits can result:

- This will be easy to link graph analysis and relational analysis to process interesting information without having to produce or load the data between two types of systems.
- It will not be sufficient to transfer or load the data between two types of systems.

Furthermore, some recent studies have shown that with some optimization techniques, graph analytic tasks such as triangle counting [31], sub-graph pattern matching [42], and a weakly linked portion [5] from relational databases can be performed better than the graph process. Unlike those structures, the aim of this research is to formulate a comprehensive structure that can extend relational databases to have the capability of executing iterative graph analysis.

1.4 Details of Previous Research

In Google Pregel [9], recursive super steps of vertex centric theory are used in the same way they are in the Bulk Synchronous Process model [7]. The user creates a compute function for the vertex, allowing it to access its value as well as the outgoing edge collection. Each vertex is computed at the same time and messages are published to the adjoining vertices. The created messages are transmitted in batch and only available to the targeted vertices after all processing is completed, resulting in a single block super step. Message passing computation is involved in recurrent level-synchronized super steps. The "VoteToHalt" method can be used at any super-step vertices; the vertex that has been voted to halt will not be activated in the next super-step unless it has input messages. If all vertices act to remove and no incoming data messages are open, the application is terminated. Apart from giraph, Google's Pregel has an open-source version called Apache Giraph [16], as well as other implementations such as Hama [10], Pregel.NET [11], and GPS [14].

Pregel has two main scalability bottlenecks, regardless of programming style: (1) the number of messages transmitted between vertices is high, and (2) the number of super steps necessary to finish the computation is countably larger. [15] Recently, it defined these drawbacks and proposed Giraph++, a partition-centric edition. This feature enables the computing process to reach all vertices and edges of a division on a disk, as well as allowing the user to define algorithms that operate on the entire division in a super step before moving messages from the partition's boundary vertices to adjacent vertices. Graphs are subdivided in such a manner that mean cuts are minimized. The number of messages exchanged and super steps taken will be reduced with core partition

computing. The remedy, however, has bottlenecks in the following situations. (1) Although Giraph++ refers to partitions as "sub-graphs," these sub-graphs are not linked components. This restricts the use of linked graph shared-memory graph algorithms, which can result in a sub-optimal algorithm that operates on hundreds of sub-graphs in a partition. (2) Only boundary vertices can receive messages from Compute. Sub-graphs are defined a priori as locally connected elements in the suggested abstraction and execution model, with users defining their Compute method.

Another popular graph processing abstraction is Distributed GraphLab [17], which is customized for local requirements in data mining algorithms. GraphLab also uses a vertex-centric iterative computing model, but it allows for simultaneous implementation with links to vertex neighbors. It does not tolerate graph variants in the same way as Pregel does. Trinity [18], for example, is a distributed graph processing system that provides shared memory representation in a parallel processing infrastructure. Algorithms use it with transferring message and the memory cloud, which is a distributed address space. This, however, necessitates the use of memory access features such as high connectors.

1.5 Details on Aim of the Project

Study, design, create, test and analyze a system for the effective and efficient computation of real-world graphic data using resources.

This project will create a platform for developers that will allow them to construct large data analytics applications that will perform graph data analysis in a batch processing manner. It will allow efficient use of resources and work exceptionally to aiding the ultimate aim of this project.

1.6 Details on Purpose of the Project

Following table describes in detail purposes that would fit achieve the aim defined above.

Purpose 01	Background and Introduction

Organize the Proposals, which will serve as a guide in the development and assessment process, describing the goals and objectives, product features, context, resource needs, project deliverables, and operation schedule.	
Purpose 02	Literature Analysis
Conducting a systematic literature review on the following topics ▪ Qualities of graph data sets and approaches for computation that support the existence of graph data. ▪ Choosing approaches from a range of methods that could be adapted to the project's scope and graph data sets. ▪ Following a comprehensive description of the literature review, provide a quick description of what is required to be developed as a framework. ▪ A thorough review of existing research and projects would also be carried out.	
Purpose 03	Design specification
Prepare the design specification for the prototype according to the specifications analysed, taken from the requirement selection process, and chosen software and hardware resources.	
Purpose 04	Implementation of prototype
To completely satisfy the user requirements defined as in requirement specification, build the prototype using the most suitable software and hardware resources.	
Purpose 05	Testing implemented model
Draw up a testing strategy, build test scenarios, and perform in-depth system testing to find bugs and ensure that the built prototype meets the users' specific and non-specifications.	
Purpose 06	Evaluation on rest of the process conducted
▪ Develop a crucial assessment including its model with identified customer segments of the framework and a summary of the review results to assess how well the project answered the theory. ▪ Conduct a study of various application areas used throughout the concept with subject matter experts to find areas for potential enhancements. ▪ Execute a self-evaluation to assess the job you've done.	
Objective 07	Project Documentation
Document a project report on all results and main steps involved in the project.	

Table 1.1 Details on purpose of the project

1.7 Requirements of Resources

A set of Software and Hardware specifications for the specified project as shown in below Table.

Software Requirements	Hardware Requirements
• GIT • Java JDK 11 • IntelliJ IDEA • Python 3.9 • NetworkX • Graph tool • SNAP • Postgresql database • Microsoft office packages • Star-uml	• OS: Any Linux Distribution or Windows Subsystem for Linux (WSL) • Processor: 12 Core with minimum 12 GB RAM • Core i7 2.6 GHz processor

Table 1.2 Resource Requirement

1.8 Details on Expected Features

This portion of the report explains the prototype's functionality, with the main characteristics of the solution suggested based on the solution's design. The data flow of the suggested solution is depicted graphically in the Figure below.

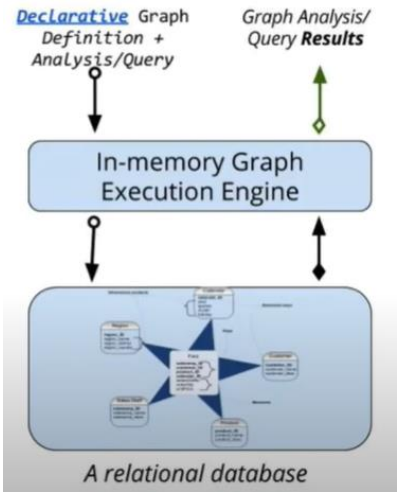


Figure 1.4 Initial diagram on execution flow of GraphMk.

1.9 Summary of the Chapter

This section starts with a description of the problem and the project's context. The project's history briefly summarizes all forms of Big Data computation on concurrent data systems, as well as the introduction of data sets and its current importance in the global trend. It continues to use real-world examples to demonstrate all use of graph data. An overview as to why parallel data structures doesn't support graph algorithms, as well as a solution, was discussed thereafter. Integrated model and frameworks for graph computation and their pros and cons have been developed and was further explained in this Chapter. Later, the element of intent was discussed, and the project's features were eventually determined. Following that, the project deliverables were outlined, and the Chapter concluded with a description of the prototype resource specifications, as well as a summary of the activity schedule planned for the life of the project until completion.f

2 Literature Review

Contents

2.1 Overview of the Chapter

The former section relates to the method by explaining the specific problem, as well as previous work on a graph computation domain, as well as the project goal and goals that must be met in order for the project to be completed successfully. The literature section will show major aspects relevant to graph-based computation and will address the significance of graph sets of data and how to efficiently compute them. It also provides readers with critical overview of potential methods, strategies, technology, formulas, methodologies, and current tools for developing the proposed framework effectively. It then moves on to a critical assessment of relevant work performed on the problem domain, including a study of current projects and their benefits along with drawbacks.

2.2 Background on different Analytical Systems

Commuting, social networking sites, communication networks, and bioinformatics are also examples of fields where graph analytics is used. It could be classified as graph theory in computer science since the essential and algorithmic features of theoretical graphs are the prevailing methodological causes of many graph analytics applications [19]. Broad networks are far more commonly accessible these days. Many companies and associations depend on the study of these networks to extract essential market insights. Since graph data is frequently processed and handled in relational databases in most occasion, where these advanced graph structures are frequently used in combination with relational databases by many businesses and organizations. As a consequence, a typical graph analytics use pattern is represented inside two separate frameworks.

2.3 Graph Analytical Framework Performance Factors

While accuracy is obviously a crucial element in every graph processor framework, several more studies have highlighted certain key areas for a productive graph analytical model, such as optimization, high availability, perseverance, confidentiality, and reliability. Even then, while most of the job on such frameworks has focused on enhancing data processing precision, [21] and [22] assert that parallelization and memory management can be just as essential as precision within

a graph data processing system. With the passage of time, the majority of systems have tried to discredit the processing of vast amounts of data. It's often spoke in a graph computation system, accuracy refers to the system in order to quantify and deliver results exactly to align multiple users choices, while robustness refers to the state's ability to supply massive data processing with stronger memory management. So, if robustness rises, storage capacity rises with it, resulting in a drop in precision in outcomes. This demonstrates that critical success factors have a poor association. Maintaining those variables at an acceptable level is a complex job. This study presents the method of simulation by choosing the optimum way to design a graph processing with precision, robustness, and memory size as key aspects.

2.4 Techniques for Graph Analytical Systems Analysis

2.4.1 Graph Computation

Below is a brief description of the analysis of the graph analytical methodology in Table 2.1.

Technique	Graph Computation approach
Description	A graph is an array of nodes with interactions within them that create a link. In a more real-life context, nodes are structures, and associations determine how they relate to the rest of the world. Graphs render modeling simpler, allowing for more knowledge to be found easily.
Advantages	**Disadvantages**
1. To recursively implement any user implemented logic that runs through vertices within a graph, Think-like-a-vertex architectures were introduced. 2. The vertex software is designed to compute data from neighboring vertices and event edges in addition to the data obtained as input.	1. As opposed to traditional graph omniscient algorithms, vertex programming is less expressive. 2. The systems have their origins in distributed algorithms. 3. Even though vertex centric concept provides results that are acceptable, they aren't enough frequent usage of concept behind think-like-a-vertex. Rather Bellman's algorithm is used for benchmarking purpose [24].

3. A synchronous or asynchronous implementation of the vertex software. 4. Scalable and parallelizable to a larger degree. 5. Distributed algorithms philosophy, such as synchronicity and coordination processes, has a significant impact on THINK LIKE A VERTEX systems [12]. 6. Addition of think-like-a-vertex concept paved the way for support of different algorithms within Maching learning and Data mining arena that helps to represent graphs better in the perspective of high performance and data-sets measured with large quantity [21].	4. Excessive graph message lengthens execution time [25], [27]. 5. Major overhead in scheduling to maintain data integrity [28], [29]. 6. Convergence needs further mega steps or iteration steps. 7. Each iteration or super stage takes longer.
Impact on performance	Graph analytical systems have excellent performance and precision, and they were developed especially for powerful fast computation datasets.
Impact on scalability & memory	Despite the fact that there are several approaches associated with the word graph analytical systems, and the memory performance of these systems varies based on the solution, their extensibility is strong since they help decentralized computing.

Table 2.1 Graph computation detail comparison

2.4.2 Relational Computation

Table below summarizes study of relational computation techniques.

Technique	Relational Computation approach
Description	Even before the advent of machines and data processing by IT, relational databases have controlled the IT universe. A number of applications and scenarios have already adopted Relational databases to store and retrieve data. Relational structures are clearly described as tables that link different characteristics of data from various

<table>
<tr><td></td><td colspan="2">scenarios. Databases are mostly used to store tabular details such as transaction notes and financial statements.</td></tr>
<tr><td colspan="2">Advantages</td><td>Disadvantages</td></tr>
<tr><td colspan="2">1. Convenience: Knowledge presented in tabular format is simple to comprehend.
2. Flexibility: Data is expressed in a number of tables, and useful data is queried.
3. Precision: Using relational algebra and relational calculus to prevent uncertainty is helpful.
4. Security: Access control and authorization are much easier to enforce.
5. Data Independence: With a normalization structure, data independence is easier to achieve.
6. Data Manipulation Language: The ability to respond to queries using a relational algebra and relational calculus-based language.</td><td>1. Performance: If a large number of tables are used, the speed at which SQL queries are answered will be affected.
2. Physical Storage Specifications
3. Slow data meaning extraction: if data is inherently structured and processed in a hierarchical fashion, the holistic solution can provide fast interpretation for the data.</td></tr>
<tr><td>Impact on performance</td><td colspan="2">The efficiency of relational queries can be contrasted to those of other queries, and it can be shown that relational queries surpass them, but even that they suffer in the graph computing phase.</td></tr>
<tr><td>Impact on scalability & memory</td><td colspan="2">This systems typically lose memory and power, and even though space and memory are available to any of these systems via hardware, these systems do have difficulty when it comes to inputting data that must be processed.</td></tr>
</table>

Table 2.2 Relational computation detail comparison

2.4.3 Hybrid Computation

Table below summarizes study of hybrid computation concept.

Technique	Hybrid computation

<table>
<tr><td>Description</td><td colspan="2">The reason is that most graph computations are performed on relational schema data, according to integrated computation models. Present systems face the task of coordinating two systems to perform graph analytical computations. Analytical systems that are combined or hybrid are implemented such that it stays on top of relational core computation and extends its views with extra logic to support graph computation.</td></tr>
<tr><td colspan="2">Advantages</td><td>Disadvantages</td></tr>
<tr><td colspan="2">1. Systems like these which can perform operations of two different systems will be more effective in future for more complex requirements and computations
2. The framework is easy to understand and use.</td><td>1. When it comes to scalability and memory management considerations, this hybrid may be volatile, since relative values vary and device behavior varies.</td></tr>
<tr><td>Impact on performance</td><td colspan="2">The performance of these structures can be considered consistent as they have a hybrid model that integrates all frameworks' approaches.</td></tr>
<tr><td>Impact on scalability & memory</td><td colspan="2">The system's scalability will be consistent as well; it will not equal the scalability of the graph structure, but it will be decent in comparison, and the same can be said for memory management.</td></tr>
</table>

Table 2.3 Hybrid computation detail comparison

2.5 Graph System Approaches and Review

2.5.1 Review on Vertex-centric

Vertex-centric structures were created to handle large-scale graphs effectively in a scalable way. A sizeable graph is split into many partitions of vertex-centric structures. Each one has a set of distributive vertices and outgoing sides. An sample data model for vertex-centric structures as seen in below figure. The source graph is split into 3 partitions (P1, P2, and P3) in below figure, with each partition containing a collection of vertices. A vertex has a specific ID (e.g. V1), a collection of values (in this case, a vertex has one value about out-degree), as well as a collection of outgoing links for identifying goals to send messages to.

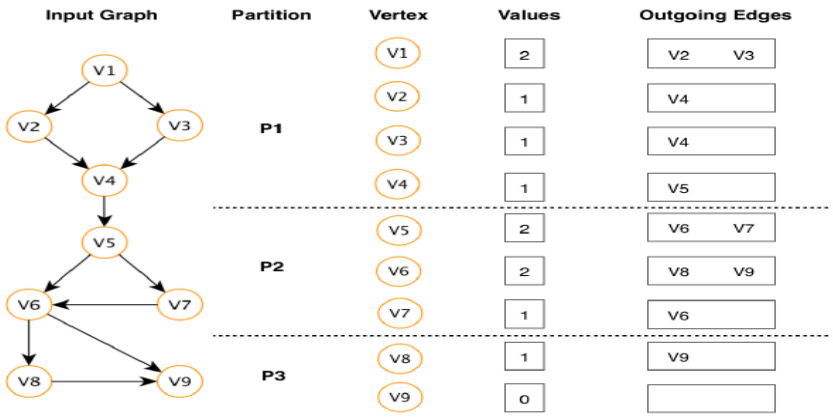


Figure 2.1 Vertex-centric data model and partitioning behavior [34]

Table below offers comprehensive overview on vertex-centric computation approach, including its benefits and drawbacks.

Approach	Computation based on vertex-centric
Description	The vertex centric concept-based computation is a well-known framework for dealing with graph processing problems of large scale. User preferred programs based on requirements can adopt the concept of 'think like a vertex' then its called system built on the vertex's perspective. In their studies, [33, 34, 35] used a vertex-centric approach.

Advantages	**Disadvantages**
1. Improves locality to a degree [24, 26, 27]. 2. Shows that linear scalability is possible [31, 32]. 3. Allows many iterative graph algorithms to be represented and computed in a natural way [34, 35]. 4. It's easy to program. 5. Proper pacing and partitioning, as well as efficient contact [36, 27].	1. Hides information about partitioning from users [41, 42]. 2. Prohibits users from making many algorithmic optimizations. 3. Excessive graph message lengthens execution time [44, 45]. 4. A high degree of scheduling overhead is needed to ensure data accuracy [47, 48]. 5. Convergence needs further mega steps or iteration steps. 6. Each iteration or super stage takes longer.

Impact on performance	Vertex-centric computation model can be considered to have high performance due to the fact that it adopts a scatter-gather approach during message passing which results in a better results in the perspective of performance.
Impact on scalability & memory	In most instances, this approach offers sufficient data and memory management scalability in a distributed manner.

Table 2.4 Comparison on computation based on vertex-centric concept

2.5.2 Review on Neighborhood-centric

Soon after the proposal of vertex-centric systems, neighborhood-centric systems were developed. Because the vertex-centric model uses vertices which are not related in collection rather than having a proper breakdown of subgraphs within the input graph, the subgraph information is hidden. As a result, for some algorithms, the vertex-centric model limits optimization (e.g. connected component and PageRank) (44). Giraph++ [45] (based on Giraph) and NScale [46] are two typical neighborhood-centric systems. Figure 2.2 portrays neighborhood-centric based data-model that structures centered on Giraph++ concepts. Below Table offers comprehensive overview on computation based on Neighborhood-centric, including its benefits, drawbacks and impacts.

Approach	Computation based on Neighborhood-centric
Description	Subgraphs are formed out of initial information diagram in a neighborhood-driven model. Every vertex consists of unique id along with collection of qualities. Vertices gets divided into two categories in this computation where they are named as inner and limit respectively. The limit vertices helps to isolate the information diagram. A vertex is an inner vertex in exactly one subgraph, which is known as the vertex's proprietor, but the same inside vertex can be considered as a limit vertex in more than one subgraph.

Advantages	**Disadvantages**
1. User is provided with partitioning information.	1. Convergence requires iterative steps more than expected.

2. Have options for users to enhance the algorithms and support user defined needs. 3. Execution delays are avoided with aid of optimized Message passing concept.	2. Each super step, also known as an iteration, takes longer.
Impact on performance	This outperforms the vertex-centric approach in terms of performance because it was introduced to outperform vertex-centric in the arena of partitioning and ease of use as an alternative and optimized phase of vertex-centric.
Impact on scalability & memory	The approach's extensibility is much more effective and linearly proportional to memory management; as size rises, memory consumption grows as well, albeit in an uniformly comparative way.

Table 2.5 Comparison on computation based on neighborhood-centric concept

Vertex-centric computation can be considered as easily understandable model that needs less effort to program and proved with records of showing different useful implementation of graph algorithms. It's always stated that neighborhood-centric model cannot be considered as a substitute to vertex but rather can be used along with vertex-centric model in combination. This was proved from successful implementation of framework as Giraph and Giraph++ to boost performance. Literature appendix provides a more detailed analysis of the vertex-centric method.

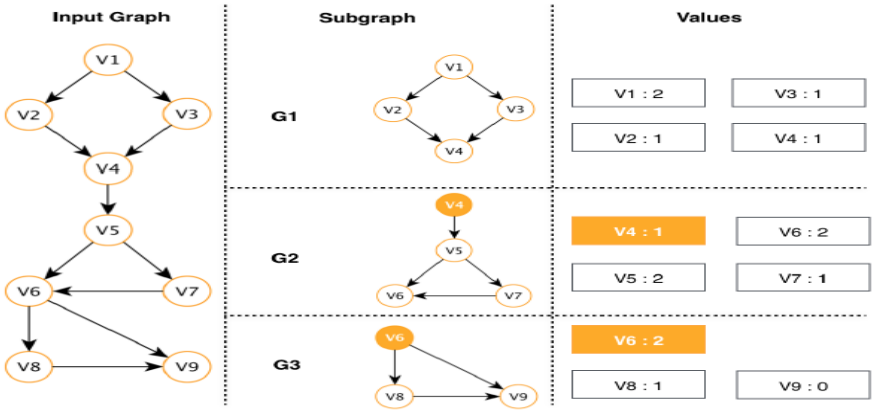


Figure 2.2 Neighborhood-centric data model and partitioning behavior [48]

2.6 Graph Analytical Algorithms and Review

2.6.1 Analysis on PageRank

In order to implement the graphmk framework, all three algorithms are used. Table 2.6 offers a short description on analysis of PageRank algorithm and its computation nature.

Algorithm	PageRank algo & computation nature
Description	A formula derived with a mathematical concept that aids in the ranking of entities in a dataset is defined as PageRank algorithm.
Advantages	**Disadvantages**
1. Improves locality to a degree [42, 43, 44]	1. It can take some time for a newly added item to be added to the computation list.
Impact on performance	Performance relies based on nature of implementation carried out and optimization process. It would also have an effect on its efficiency and accuracy. The data format has an influence on results.
Impact on scalability & memory	The algorithmic design will impact the extensibility and resource consumption of the simulation; as the parallelization grows, resource consumption to calculate the function may improve, and the two will be comparable.

Table 2.6 Comparison on computation based on PageRank algorithm

2.6.2 Analysis on Clustering

A brief overview of Cluster algorithmic computation is given in Table 2.7.

Algorithm	Cluster Algo & computation nature
Description	Grouping individual elements into a known cluster where the cluster can be identified with a commonality can be considered as the procedure of Clustering.
Advantages	**Disadvantages**

1. Validates linear extensibility [48, 49]. 2. Minimize the amount of time it takes to send a message. 3. Graph algorithms that can be reused.	1. Users are unaware of partitioning knowledge [51, 52]. 2. Prohibits users from making certain algorithmic optimizations.
Impact on performance	Since many different algorithmic functions available for clustering concept, such as (Connected Components, Strongly connected components, and Connected components), the efficiency and correctness can only be determined by the algorithm selected for execution.
Impact on scalability & memory	Scenario and used algorithm would have an effect on scalability and memory; as scalability increases, the memory needed to compute the algorithm will increase, and the two can have a linear proportion.

Table 2.7 Comparison on computation based on Clustering algorithm

2.6.3 Analysis on Path

Below table offers short description of path algo and its computation nature.

Algorithm	Path Algo & computation nature
Description	Finding the best paths between two points is what route algorithms are all about. The best algorithmic implementation of this type is Dijkstras shortest path algorithms.
Advantages	**Disadvantages**
1. Lowers and removes graph overhead. 2. Decreases the number of super steps in the computation.	1. Writes incomplete, intermediate results to disk for later retrieval, doubling sequential IOs. 2. Due to the doubling of sequential IOs, there is an increase in computing expense and data loading overhead.
Impact on performance	The algorithm's accuracy is directly proportional to its efficiency.

Impact on scalability & memory	The scenario and the used algorithm will have an effect on scalability and memory; as scalability increases, the memory needed to compute the algorithm will increase, and the two can be linearly proportionate.

Table 2.8 Comparison on computation based on Path algorithm

2.7 Graph Databases and Review

As diagrams for graph analytics, graph databases emphasize productively overseeing and handling information. For example, social databases must use expensive join operations on tables to perform graph computation operations like finding likely neighbors. The main concept behind including connections within objects which will result in building a long-term image is considered as information model. Information model can be divided into two main aspects which includes a property diagram and then one with triple stores with resource description framework.

2.7.1 Property graphs in Information model

A brief overview of the Property graph model in graph databases is given in Table 2.9.

Model	Property Graph
Description	Analysts refer to these graph models as "white board friendly" since can be illustrated easily within a white screen. Usually, graphs are formed by vertices and edges, which are defined like G = (V, E). Nevertheless, property graphs consist of attributes that are depicted along with adjacent vertices relationships as well. This model is used in the implementations of many common graph databases.
Advantages	**Disadvantages**
1. This contains added meta data possibly used in graph algorithmic computations. 2. Integrates semantics into relationships like consistency and weight.	1. This method has a high implementation overhead since it uses different APIs than others to provide features, forcing users to re-implement several graph algorithms. 2. This method does not help a few graph algorithms.

Impact on performance	Since the data is expressed in a graph that is effective for computation, productivity and correctness within a graph computation process can be enhanced with the use of property graphs.
Impact on scalability & memory	Memory utilization within this computation can be efficiently maintained, whereas extendibility is an inherent nature of this computation.

Table 2.9 Comparison on computation analysis for property graph

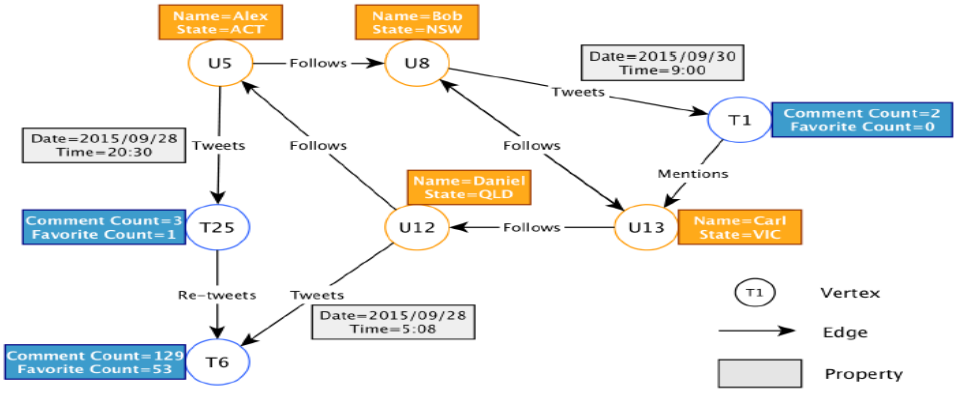


Figure 2.3 Representation of property graph data model using tweets [35]

2.7.2 Triple stores

Below table provides quick overview of the Resource description framework Triple store model computation.

Model	Resource description framework Triple store model
Description	RDF was developed in 1999 with the aim of supporting the web of semantics incorporating markup for semantics that links resources in web. Data is stored as subject-predicate object data rather than as a graph.
Advantages	**Disadvantages**

1. They do not help adjacency without an index. 2. It has the ability to optimize graph queries. 3. They can outperform relational databases in terms of efficiency.	1. This does not allow for the evolution of graphs over time. 2. It is unable to accept dynamic graphs because it does not keep track of value changes in vertices.
Impact on Performance	These are less effective than property graphs because they don't store data in graph format. The output is still there, but it isn't up to par.
Impact on scalability & memory	This model preserves data as predicate object and subject data instead of graph format where it helps to provide more scalability than property graphs. However, it does not provide effective memory supervision for execution.

Table 2.10 Comparison on computation analysis for RDF triple store model

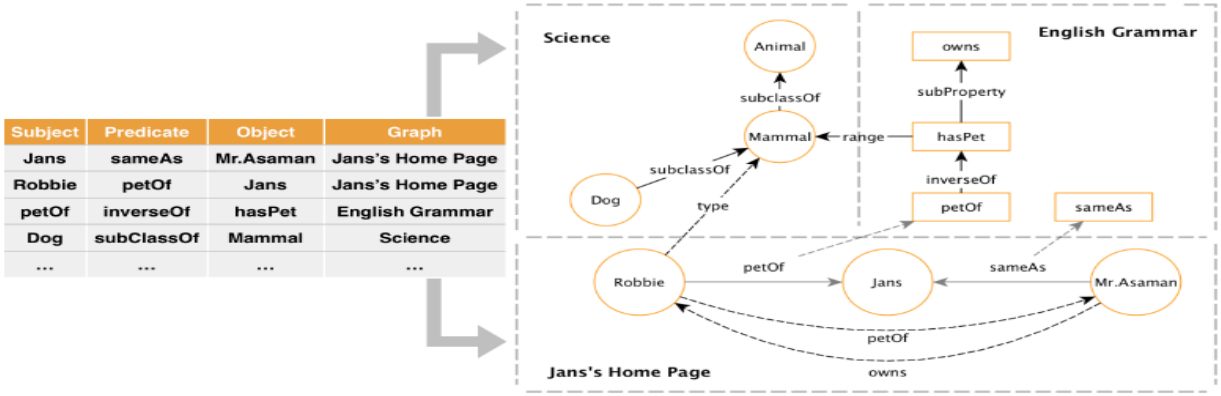


Figure 2.4 Representation of triple graph data model using user relationship [44]

2.8 Relational Systems and Review

A brief overview of Cluster algorithmic computation is given in Table 2.11.

Technique	Relational system
Description	Vertex centric proposed methodology is a well-known analytical method for coping with extensive graph processing problems. A system that encourages users adopt "think

	like a vertex" and executes client functions via a vertex's viewpoint. In their studies, [33, 34, 35] used a vertex-centric method.
Advantages	**Disadvantages**
1. Inside supersets, reduces disk buffering and memory strain. 2. Cuts down on the time it takes to compute each super move.	1. Hides details about partitioning from users [41, 42]. 2. Prevents users from performing many algorithmic optimizations.
Impact on accuracy	For normal computations, relational structures provide efficient performance and precision, but not for iterative algorithms.
Impact on scalability & memory	They have a limited edition of scalability, despite the fact that the hardware supports it, and the same can be said for resource administration.

Table 2.11 Relational system computation and review

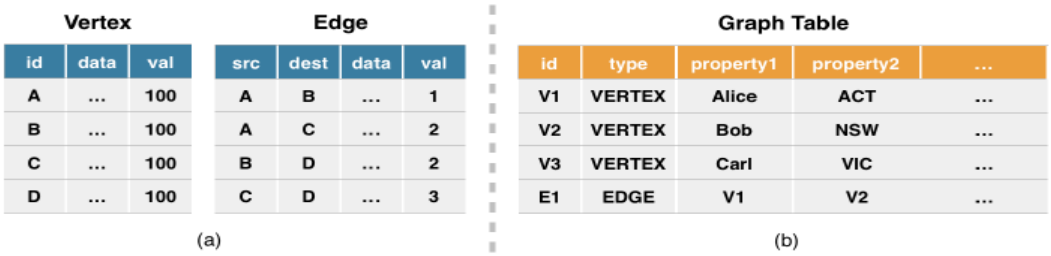


Figure 2.5 Illustration of relational table structure for graph views. [45]

2.9 The Integrated Graph Analytical System's suitability

Disadvantage of Graph system	Advantage of Relational system that could negate
Popularity of Graph databases for the purpose of storing data is not widely considered.	Data in relational databases are used to execute graph data and computation.

Since there are no declarative languages for querying data in the graph model, users must do imperative programming.	In relational structures, SQL queries have a declarative language for computation.
When it comes to machine efficiency, not all algorithmic computations can behave the same. Each algorithm, dataset, and hardware used for computation have different computation efficiency.	Furthermore, several recent studies show that for graph computation, such as single source shortest path, triangle counting, PageRank, and linked elements, merely using a SQL-based method will obtain excellent or equivalent output than vertex-centric systems [31, 34].
Graph databases aren't widely used for data storage.	Relational databases are still used to capture and track graph data.
A powerful query language with proper optimization is lacking in graph systems.	Relational systems based on SQL have developed powerful query languages with high query optimizations.
For query processing, graph systems lack a robust fault tolerance mechanism.	The fault tolerance function in relational systems is built on a solid basis.
Graph systems place less emphasis on safe transactions.	Stable query execution transactions are ensured by relational systems.

Table 2.12 Reveals Fitness on Relational System

Disadvantage of Relational system	**Advantage of Graph system that could negate**
For graph analytical activities, relational models are inefficient.	Graph models are designed with the aim of supporting iterative computation and get quick glance of relationship among entities.
When it comes to iterative algorithms, relational models are inefficient.	Graph models are abandoned in favor of models that enable iterative algorithm computation.
The SQL language makes expressing graph analytical operations difficult.	Relational databases are still used to capture and track graph data.
Since query languages lack the capacity to interface with SQL, a declarative data log that can communicate with SQL can produce competitive success for just a few graph	And graph databases' query languages aren't strong enough to be declarative languages.

computations tasks, necessitating the development focusing on scalable method that supports majority analytical computation.	
For graph analytical activities, relational models are inefficient.	Provide as many computations and algorithms as possible to meet a wide range of requirements.

Table 2.13 Reveals Fitness on Graph System

Above two tables show strong review association between the structures and the results. Clients should be provided definitive languages to recover knowledge efficiently in vertex-driven and neighborhood-driven frameworks. Requests for challenging details in graphs and also questioning structure knowledge must be strengthened in map databases. Furthermore, the overwhelming majority of users use social graphs to achieve and prepare records. Therefore, for machine analysis projects, an architecture that is based on SQL, delivers a definitive query language which executes quickly must be implemented. Established SQL-based social systems currently have the following limitations: (1) capability to have graph views and ability to total data view through SQL is expected. (2) If you're looking for some framework that would have capability to generate graph models on top of query engine and provide outcomes instantly while exploiting features of relational query engine and query language for effective execution. Thus, a system must be proposed that performs this type of half-and-half procedure with the help of a question engine, query language and knowledge model.

2.10 System Analysis – Pros & Cons

Below two tables show comparison on both benefits & drawbacks for the two schemes under consideration.

Advantage & description	Impact on performance	Impact on scalability
Suitability to open domains – Both frameworks can be used in any kind of domain.	This would have no effect on the graph computation results.	There will considerably amount of negative impact when it comes to scalability. Consideration of storing data in required format is expected

Enhanced reliability – Since data quality and value can never change at some point of time, it will be easier to maintain.	This would have no effect on the recommendation system's results.	The scalability will suffer as a result of this.

Table 2.14 Detail description on advantages of the system

Limitation & description	Impact on performance	Impact on scalability
Query persistency issue – The query executor isn't smart enough to check for query persistency in real time.	This would have a negative effect on the accuracy of added query computation. Accuracy drop is expected when plan tree selection is hidden from user.	It will necessitate more computing resources and have a negative effect on the system's scalability.
Memory issue – These devices sometimes run out of memory and overflow.	For a while, this will have a negative effect on accuracy before the device is fixed.	The additional resource requirements would have an effect on the system's scalability.
Data Integrity Problem – At any given time, data integrity is not upheld.	Certain better matching plan trees will be concealed from the user, resulting in a drop in accuracy.	The additional resource requirements would have an effect on the system's scalability.

Table 2.15 Detail description on limitations of the system

2.11 Deep dive of Existing Systems

Present graph computing frameworks are illustrated and reviewed in depth in the tables 2.16, 2.17, and 2.18. Appendix B contains a more thorough review of current structures (Section 11.2.6)

Name	Graph-Chi
Features	• A numerical approach that is focused on the vertex. • Modeled after a parallel sliding glass.

	• Deterministic scheduling and asynchronous, simultaneous execution. • Can run graphs with large quantities of edges on a regular user PC with linear scalability. • By caching, it is possible to use a huge amount of memory. • RAID-style operation with multiple disk striping. • Compatible with both hard drives and solid-state drives. • Allows for the development of changing graphs. • Supports graph updates in real time. • Ability to add and remove edges.
Performance	**Scalability and Memory**
• Allows you to articulate and compute several iterative graph algorithms in a natural way. • Simple to configure. • Unnecessary graph messages result in a longer execution period. • A high level of scheduling overhead is needed to ensure data accuracy.	• Enhances locality to some degree. • Exhibits linear scalability • Proper timing and partitioning, as well as effective contact. • Users are not aware of the partitioning details. • Prevents users from making many algorithmic optimizations.

Table 2.16 Graph-chi framework analysis of pros and cons

Name	**X-stream**
Features	• An executional method that is focused on the edge. • Usage of Scatter and Gather paradigm is involved. • Random access to edges is avoid with stream edges. • Can run graphs with large quantities of edges on a regular user PC with linear scalability. • Performs admirably on solid-state drives (SSDs). • Allows for the development of changing graphs. • Supports graph updates in real time. • Ability to add and remove edges.
Performance	**Scalability and Memory**
• Avoids random access to the edges by not sorting the edg during processing. • Writes incomplete, intermediate results to disk for later retrieval, doubling sequential IOs. • Due to the doubling of sequential IOs, there is an increase computational expense and data loading overhead.	• This method has a high development overhead since it uses different APIs than others to provide features, requiring users to re-implement several graph algorithms. • This method does not help a lot of graph algorithms.

Table 2.17 X-stream framework analysis of pros and cons

<table>
<tr><th>Name</th><th colspan="2">Grail</th></tr>
<tr><td>Features</td><td colspan="2">Can run graphs with large quantities of edges on a regular user PC with linear scalability.
It performs admirably on a hard drive.
Allows for the development of changing graphs.
By caching, it is possible to use a huge amount of memory.
RAID-style operation with multiple disk striping.
Compatible with both hard drives and solid-state drives.</td></tr>
<tr><th colspan="2">Performance</th><th>Scalability and Memory</th></tr>
<tr><td colspan="2">Enhances locality to some degree.
Exhibits linear scalability
Allows you to articulate and compute several iterative graph algorithms in a natural way.
Simple to configure.
Proper timing and partitioning, as well as effective contact.</td><td>Users are not aware of the partitioning details.
Prevents users from making many algorithmic optimizations.
Unnecessary graph messages result in a longer execution period.
A high level of scheduling overhead is needed to guarantee data accuracy.</td></tr>
</table>

Table 2.18 Grail analysis of pros and cons

2.12 Summary of the Chapter

This section was articulated with the intention of setting a thorough background knowledge on key research areas involved in the project objective and problem, as well as to identify the most appropriate approaches for solving the problem domain. The Chapter began with a summary of the overview of Graph Analytical Systems. Following that, key success factors of graph analytical systems were investigated, with the aim of this study being to determine the best approaches for achieving a tradeoff between the system's precision, scalability, and memory management. The next section of the Chapter goes through the components of a graph analytical framework. Since it was established that the analytical engine is at the heart of any graph system, further research into the analytical engine architecture and design was conducted. Hence, discovery of analytical systems possibly divided into 3 groups following the reference method used. Diverse data related methods, as well as the techniques used in those models, were examined critically. Following that, the Chapter explains an overview of the graph analytical systems approach, as well as the advantages and disadvantages of the graph approach, as well as the effect these have on the system's scalability, precision, and memory. The critical research review addressed and concluded the issues with those

approaches. They don't have user friendly languages defined that are declarative which users are provided with to easily get data in vertex-centric and neighborhood-centric structures. The study of the relational analytical system method continues in this Chapter, which is accompanied by an assessment of the advantages and limitations of the relational approach, as well as the effect these have on the system's precision, memory, and scalability. Each of the GA and RA approaches has its own set of flaws, a hybrid approach was considered. The value of providing an integrated framework is demonstrated by a thorough examination of various database types and data. Lastly, current structures were assessed thoroughly. The next Chapter reflects context more on Project Management, that will cover aspects of project management process involved in GraphMk project.

3 Design & Architecture

Contents

- Overview of the Chapter
- GraphMk - High level design description
- Description on solution and design goals
- Details on introduced integrated graph model
- Detail on introduced SQL functions
- GraphMk – Detail on Data flow diagram level 1
- GraphMk – Details on Class Diagram
- GraphMk – Details on Sequence Diagram
- GraphMk – Details on Architecture refinement
- GraphMk – Details on packages & dependencies
- GraphMk – Details on ER Diagram
- Summary of the Chapter

3.1 Overview of the Chapter

This section will discuss in detail proper design for the prototype to be built. This Chapter includes details of high-level and low-level aspects of design and architecture and overall prototype design will be discussed too. This is an introduction to the system's subcomponents. Components and data flow aspect of the systems will be discussed which will reflect the functionality expectation that can be completed by the system. It also includes UML diagrams. The system's design objectives are discussed at the end of this Chapter.

3.2 GraphMk - High level design description

Interaction among components of a systems is hard to illustrate and understand, high-level approach not only makes it easier to comprehend them, but it also provides a quantitatively accurate image of the end objective and the steps that must be taken to get there. The methods mentioned above will assist in meeting the requirements specified in specifications of requirement.

The system consists of three main components involved in the execution of its main functionalities. A Bolton solution where it sits on top a relational query engine like postgresql, additional SQL operations to represent graph functionalities and to respond for user queries which contain them and a hybrid model which can assist and execute needed logic behind the framework to execute needs graph analysis.

3.2.1 GraphMk Framework description on rich picture

A medium that can help to illustrate a problem and possible solution in a high-level perspective using diagrams can be considered as rich picture. This can further be enhanced and made matured from an initial design model to an effective design following a proper methodology. Following image gives details on graphmk system rich picture.

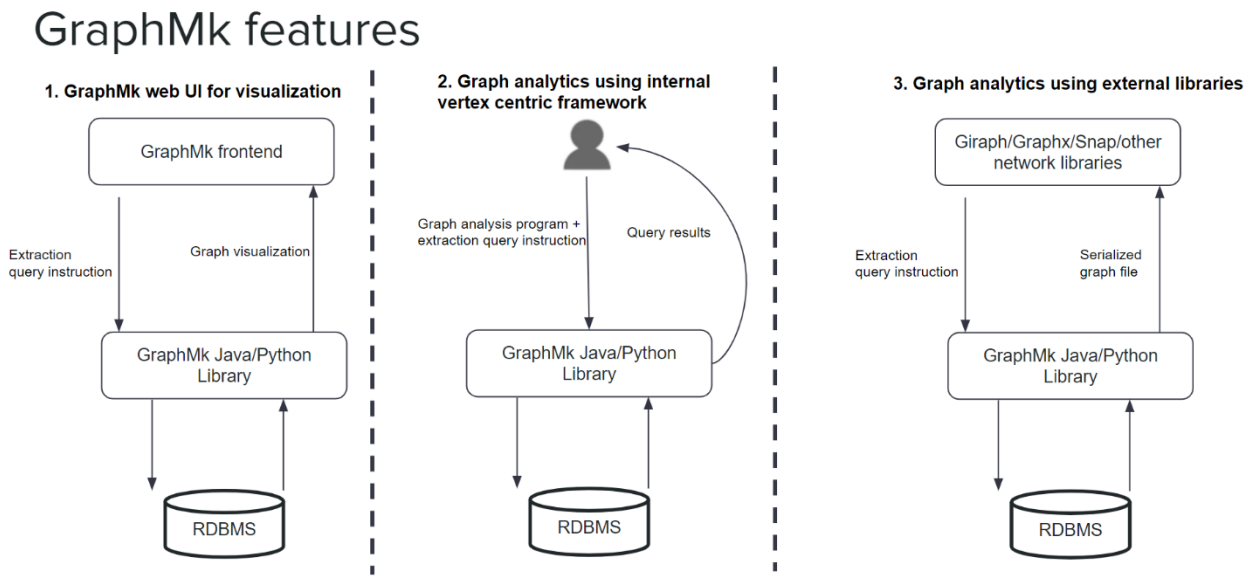


Figure 3.1 GraphMk system in a rich picture

3.2.2 GraphMk – Description on High Level Architecture

Software architecture consist of Three-tier architecture concept that has been proven in the industry and can be used to model enterprise-level applications. The independency of the three tiers can be considered the key benefit of this architecture, since it allows the user to easily update or replace them. As state by [57], issue resolving capability is found within three-tier architecture that includes security, scalability, and fault tolerance. Proposed GraphMk graph engine's high level architecture was modelled using the three-tier architecture for the reasons mentioned above. The graphmk framework's high-level architecture is depicted in Figure 5.2.

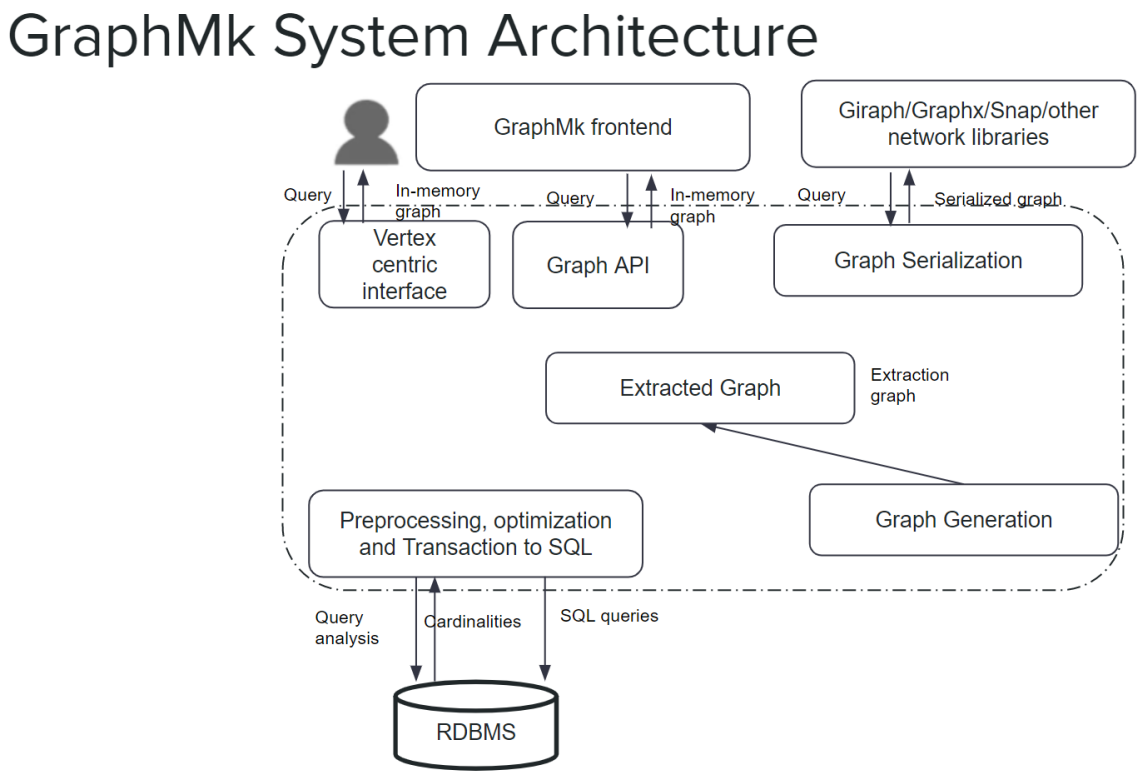


Figure 3.2 The graphmk framework's high-level architecture

3.3 Description on solution and design goals

The overall system's design process meets the key design objectives outlined below.

Accuracy and Performance

The accuracy and efficiency of the system, as well as the information generated and processing time, are important factors to consider. During the design process, special attention was paid to achieving the best possible outcomes in terms of efficiency and accuracy.

Memory Management and Scalability

Meanwhile, scalability is a critical success factor for a graph computation engine, and the GraphMk framework was designed with the highest possible scalability in mind.

Adaptiveness

The system must be effective in such a way that each consumer who is introduced to it is given a reasonable period of time to become acquainted with it. Users should be given continuous assistance in completing all tasks that are available to them.

3.4 Details on introduced integrated graph model

The Relational Mapper View (RMV) model presented in the thesis includes graph views, relational core and relation-graph mappers. The relational core, which is at the heart of the data model's various relationships, surrounds a variety of graph views. Relation mapper views utilized to map relationships in graph views. The model is known as an Relational Mapper View because it allows users to build graphs with multiple relationships. The RMV model is based on the ACM bibliographical graph, as shown in Figure.

3.4.1 Relational Core

The relational center in the RMV model is made up of a set of relations. The relational schema labels each relation, which contains a number of tuples. In real-world scenarios and applications, each tuple represents an entity. For the RMV data model, the following bullet points describe the detail concepts of the connection core component:

- Let $D = \{Di\}$, where I denote the infinite number of domains ($i \in \aleph$) that can be created, and each Di denotes a single domain. Strings, Integers, and Booleans are all examples of domains.
- Let R be described as a relational schema with a finite number of attributes $\{A1, \dots, An\}$ and domain assignments. This can be expressed as $dom : R \rightarrow D$, with each R attribute Ai being assigned to a domain.
- $t(A) \in dom(A)$ for all the Aattrs in the Relational Schema (R) ($A \in attr(R)$) – tuple is a mapping, $t : R \rightarrow D$. The attribute (A) in the tuple (t) is mapped to the relations schema (R).
- R-tuples are a finite number of R-tuples in a Relational Schema (R).
- A relational core C is a collection of Relational Schema(R) $C = \{R1, R2, \dots, Rn\}$.Connection.

Value domain & Identifier domain are two sections of relational core, which are expressed in set notation as $Did \subseteq D, Dva \subseteq D, Did \cap Dva = \emptyset \; and \; Did \cup Dva = D$. The identifier domain represents a collection of entity identifiers, while the value domain represents allowed values.

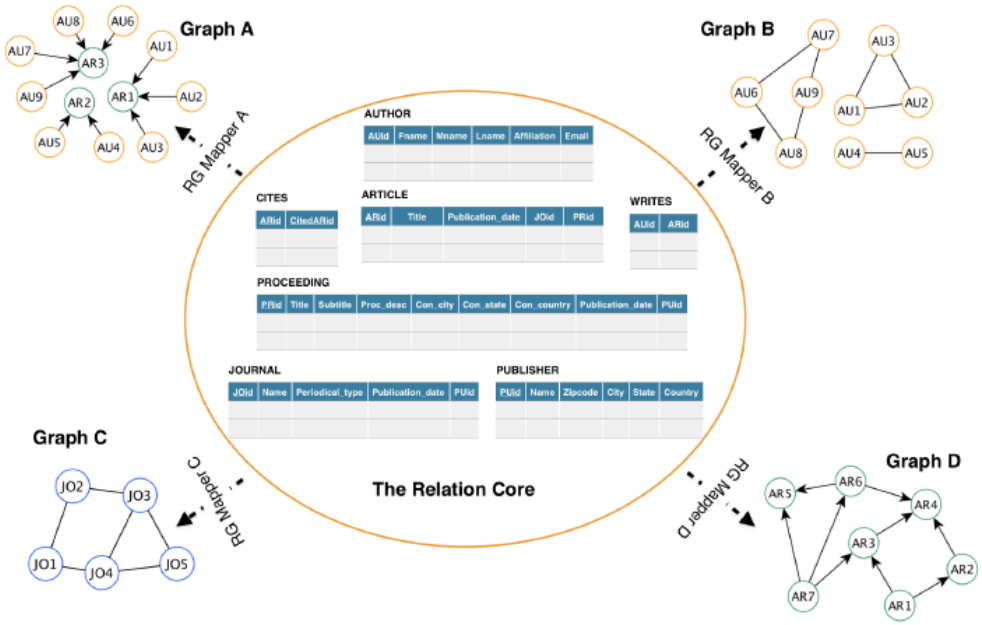


Figure 3.3 Relational main element illustration

3.4.2 Graphical Views

Based on the relational core, the RMV model can create graphical views. Each view can be interpreted as a graph, with each vertex corresponding to a single individual. The edge establishes a link between two individuals. Each graph can be defined in terms of a graph schema, which specifies the types of entities that the graph's vertices should contain, as well as the connections between them, which are represented by edges. This entire scenario is defined by entity and connection classes, and the definition is described as follows:

- Entity class $\mathcal{E}$ represents entities with identical attributes and behaviours. In the RMV model, each entity class represents a set of entity identifiers from the same identifier domain.
- Connection class l represents the relationship between two potential entities $\mathcal{E}1\ and\ \mathcal{E}2$. When a link class meets these conditions: $(\mathcal{E}1, \mathcal{E}2)\ \epsilon\ l\ and\ (\mathcal{E}2, \mathcal{E}1)\ \epsilon\ l$, it is considered symmetric; otherwise, known as asymmetric.
- Graph context $\mathcal{g}$ consists of classes of two types relating with one relation that connects them. During an instance where ties are considered symmetric then relating class is described as $l\ \subseteq\ \mathcal{E}1x\mathcal{E}2$, the graph schema is considered an undirected graph; otherwise, it is asymmetric and the graph is known as directed.

- The graph schema $\mathcal{g} = (\mathcal{E}1, l, \mathcal{E}2)$ can be used to describe a graph G = (V, E) as a collection of vertices $V = \mathcal{E}1 \cup \mathcal{E}2$ & a set represented as edges $E \subseteq l$.

G = (V, E), where E & V are the edges & vertices, in regular graph structure. V stands for Vertex identifiers, while E stands for Vertex Pair identifiers. In the relational heart, the RMV model stores entity identifiers as graphs and others. Both identifier domains must be pairwise disjoint to ensure that each vertex in the graph represents exactly one entity.

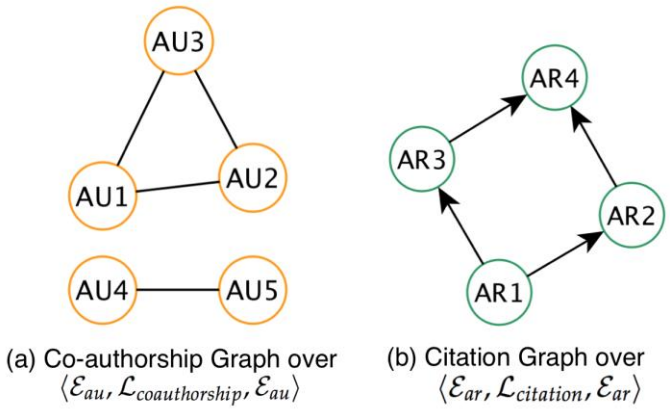


Figure 3.4 Graphical views concept illustration

3.4.3 Relational Mapper views for Graph

The Relational Mapper View (RMV) model describes an Integrated model to map views for graph, which recursively generate graphs for each relation given a set of relations. The following is an explanation of the descriptive concept:

- Let's say the input schema is Sm, and it takes a collection of relational schemas denoted as $Sm = \{R1, R2, \dots, Rm\}$. $I(Sm)$ denotes a set of relations between relation schemas in Sm.
- Let's say that the output schema is $Om = \{\mathcal{E}1, l, \mathcal{E}2\}$, and graph schema is $I(Om)$.
- RMV is a mapping of a set of input context relations to a graph context represented as output context, symbolised like $I(Sm) \rightarrow I(Om)$.

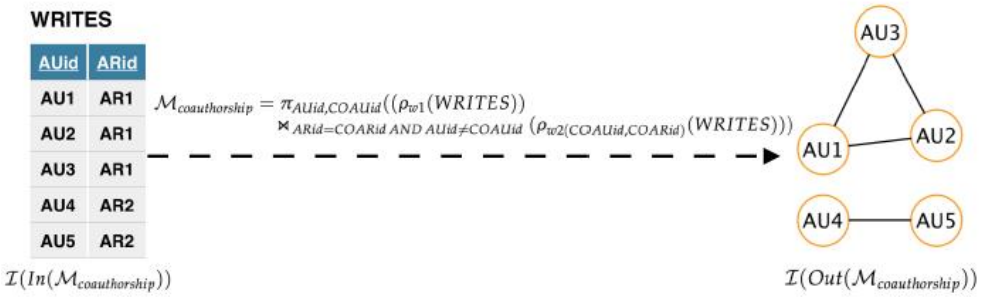


Figure 3.5 illustration of graph mapping of views in a co-author scenario

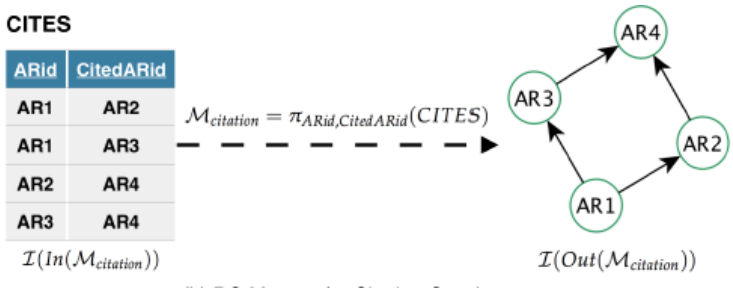


Figure 3.6 illustration of graph mapping of views in a citation scenario

3.5 Detail on introduced SQL functions

3.5.1 Selection of Design Methodology

In a research project, design methodology is an essential aspect to consider because procedures based on the methodology chosen must be followed from the beginning to the end of the project development phase. The methodology selection process is influenced by a number of factors, including the development context, type of software, specifications, and constraints relating to time. While accessible to numerous design methodologies via internet is possible, the following 2 are the most commonly utilized:

- Object-Oriented Analysis and Design Method (OOADM)
- Structured Systems Analysis and Design Method (SSADM)

GraphMk usage of Structured systems analysis (SSADM)

The approach on SSADM aids in the development of functionally focused applications. Since it does not endorse the modeling of complex systems, this approach does not seem to be very common

and is not commonly used as a key design methodology in today's world of growth. The SSADM approach encourages the development of lightweight, straightforward applications.

Diagram of level zero The data flow as a whole in the graphmk framework is represented by the context diagram, and chunks are broken down into individual data flow diagram processes in level one. Details on individual DFD diagrams in level 1 is described in the table below. GraphMk depends on postgresql or any other relational query engine for computation process and maintain database constraints, where these well-equipped systems are designed with proper structured process and GraphMk needs to follow the same guideline to work successfully as a Bolton solution.

Module	Description
Query Console	The console helps to give guided information on the process involved in computation of inputted query, time consumption, errors and results.
Relational Query Processor	Execution of relational queries with aid of parser and executor process modules.
Graph Processor	Processor that depends on input definition from user to generate materialized views for graphs in postgres.
Query Parser	Query parser's job is to validate, interpret, and rewrite graph-subqueries.
Query Optimizer	This module assists in the development of a plan tree from a user-specified query, as well as plan tree creation based on a cost estimation logic.
Executor	Predefined executors are built to support rank path and cluster and open interface for vertex centric model-based executor is available to implement any graph analytical based logics.

Table 3.1 data flow diagram details on the modules utilized in level 1

Although above Table lists data flow process within level 1 context inclusive of all modules. All modules mentioned in the above table adhere to object-oriented paradigm inclusive of executor module. Executor module strictly follows object-oriented development since communication dependency on a third-party graph computing system that supports artifacts. The executor module developed an object-oriented approach as a result of this. Since both methods can be done with Python, the systems' integrations were not a concern. Python is a sophisticated programming language that can accommodate both object and organized orientations in execution.

3.6 GraphMk – Detail on Data flow diagram level 1

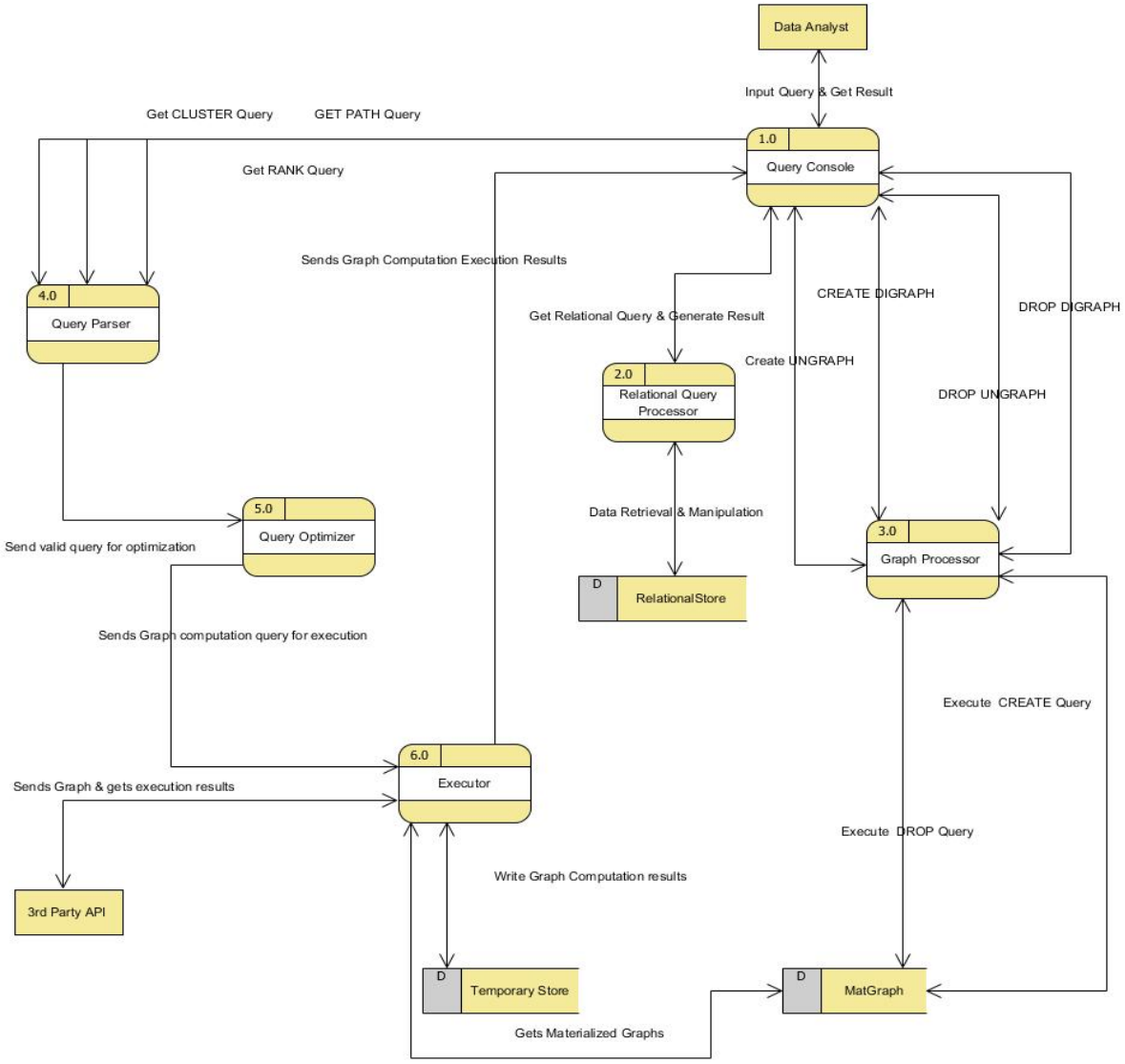


Figure 3.7 Representation of data flow diagram for graphmk

GraphMk usage of object-oriented design (OOADM)

The object oriented and design methodology tactic facilitates the creation of applications based on object-oriented style. Because of its ability to model complex and larger software, the OOADM method is commonly recognized as a prominent development paradigm within software industry. OOADM can deconstruct a complex system within constituent objects and make sure integration of data and operations to work monolithic.

Because the external systems that are used to integrate with GraphMk are mostly built based on object-oriented way and they expect inputs and outputs in the context of objects. To have a streamlined process its decision of grpahmk to adopt to object-oriented context.

Module	Description
Graph generator	instance guides stakeholders through the process of generating graphs and manipulating graph-based algorithms.
Graph	As a result, the graph's nature is represented, along with its nature consisting of attributes and functions that aid them.
Graph Type	Represents an enum definition which provides list of different graph types that are supported in the system.
Graph Execution Type	Represents an enum definition which provides list of different graph computation types that are supported in the system
Path Graph	The Path computation graph style behavior is described by this.
Cluster Graph	Cluster execution process implemented instance.
Rank Graph	Rank execution process implemented instance.

Table 3.2 Description of components involved in creation of functionalities as instance

3.7 GraphMk – Details on Class Diagram

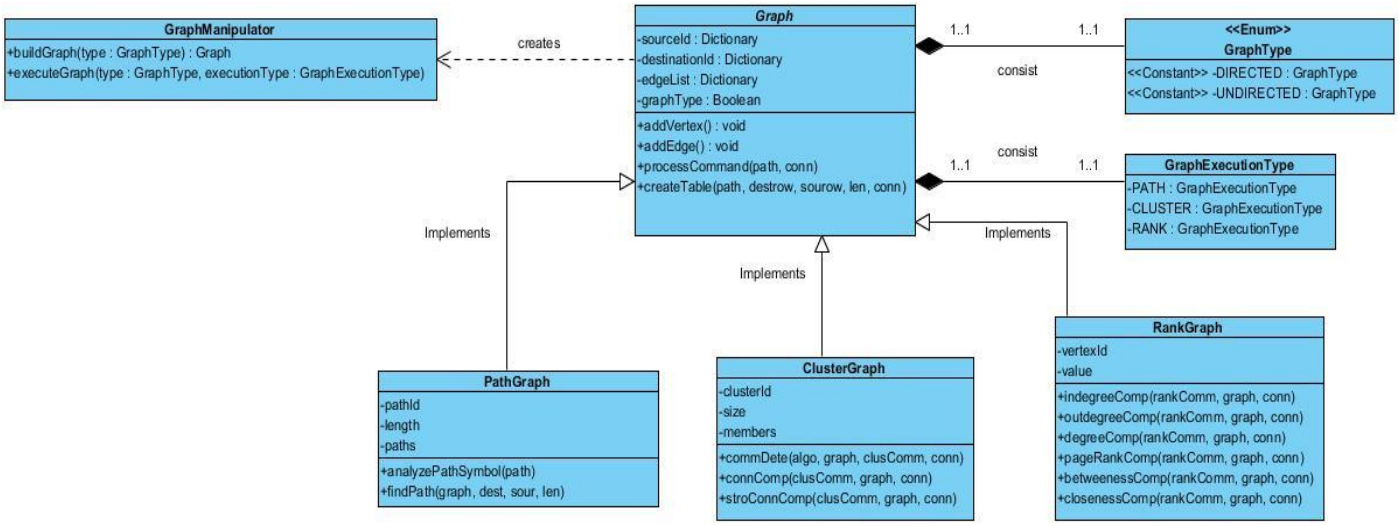


Figure 3.8 Representation of class diagram for GraphMk framework

3.8 GraphMk – Details on Sequence Diagram

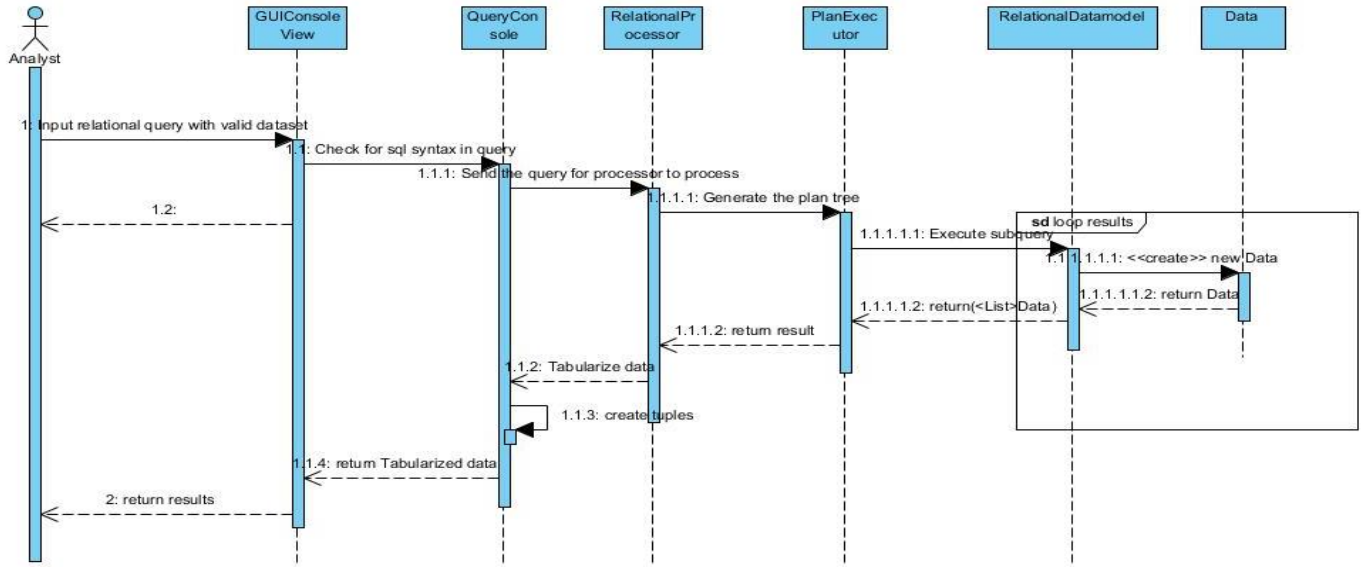


Figure 3.9 Representation of relational computation via sequence

Above diagram show how relational execution can be performed in the system via a sequence representation. The diagram details entire flow on how user enters query into the system and how system process it until the user sees results from the system.

3.9 GraphMk – Details on Architecture refinement

System design was optimized with aid of architecture and design patterns

Details on suitability of Factory design

One of the most commonly used outline designs is the production line. This type of outline example falls under the category of creational design since it demonstrates that creation of object through this pattern can be considered as ideal approach. In Factory design, create an object and not disclosing reason for its development to the consumer, and use a common interface to allude to a recently created post.

Techniques to use during design of Data Flow

The aspect of explaining how data is carried within the system via proper documentation and diagrams. This includes various stages that includes transformation and manipulation of data from one format to another, store them, process them and what output formats are provided from different modules in the systems needs to be detailed in the documentation. This can be represented to certain extend through data flow diagram.

Techniques on how to use ETL for data extraction

ETL are typically used to combine data from a variety of applications (frameworks), which are often developed and maintained by different vendors or run on partitioned PC hardware. Various members often supervise and work on the various structures comprising the initial data.

3.10 GraphMk – Details on dependencies & packages

Check on design specification appendix for more information

3.11 GraphMk – Details on ER Diagram

Check on design specification appendix for more information

3.12 Summary of the Chapter

Graph Computation Engine's design and architecture are demonstrated in this Chapter. This Chapter starts with a detailed illustration of the GraphMk graph engine and then moves on to a discussion on modules involved with an illustration of architecture on a high-level. Details on integrated graph model, bolt-on computation process and sql additional operation added are discussed and designed. The design discusses on key modules that are necessary for the development of expected functionalities were discussed and presented few modules which include executors, query validator and executor, graph specific processor, parser and optimizer of queries where these will be included

in the bolt-on solution which will be placed on top of PostgreSQL engine. Discussion on suitable architectural patterns were conducted along with design of low-level designs of data flow diagram, sequence diagram and ER diagrams. The Chapter then went on to highlighting the GraphMk system's package dependence and ER diagrams. Finally, the overall design goals for the GraphMk solution were defined as accuracy, scalability, and adaptability. The prototype implementation of the GraphMk project will be discussed in the following sections.

4 Implementation

Contents

- Overview of the Chapter
- Graphmk selection of Technology
- Graphmk supportive tools used
- Description on implementation of GraphMk
- Summary of the Chapter

4.1 Overview of the Chapter

This section focuses on GraphMk – Integrated Graph Computation framework's implementation mechanism. This includes an overview of five key component implementations. The Chapter starts with a review of the technologies chosen for the project, as well as a discussion of the resources and libraries chosen. The framework's key functionalities are then discussed, along with the implementation process, issues experienced, and solutions found. It also includes pseudocode and code snippets that have been introduced.

4.2 Graphmk selection of Technology

4.2.1 Selection of operating system

Ubuntu operating system has been selected as the operating system for graphmk. Bigdata analytics are mostly available as opensource and they are maintained by opensource stakeholder organizations where OS that falls under the Linux distribution umbrella, which is often considered open source. Massive data analysis necessitates processing that is adaptable, adaptable, and solid at an expense that does not have a tremendous impact on IT budgets.

4.2.2 Programming Language selection

The key core language for developing the graphmk integrated graph framework was Python. Python is easy to learn open-source language used mostly in Big Data analytics, and has effective libraries for information control and investigation, and other justifications made for the language's selection are as follows:

- It will be easier to adopt to python for most of developers working in backend development.
- Python is a one-of-a-kind combination of a capable, widely useful programming language and an easy-to-use language for analytical and quantitative computing. Meanwhile, Python has been used to create extremely scalable web applications such as YouTube, as well as powering most of Google's internal framework.

- Python is easy to learn and use for analysts, but it's powerful enough to handle even the most difficult problems in almost any field. It works well with established IT infrastructure and is extremely stage independent. Python-based arrangements are legendary for their agility and reliability among today's languages.

4.2.3 Selection of an IDE and a development environment

For the following reasons, Pycharm was chosen as the IDE for the implementation of the graphmk system alongside pure Python, which includes a large list of inbuilt tools and libraries:

- Opensource IDE
- Smart python editor with refactoring, auto completion and debugging

4.2.4 Selection of Storage System

The use of a relational database for the graphmk system has already been chosen, debated, and justified (Refer to Literature Review 2.5.2 section). Due to the following justifications, graphmk framework chose postgresql as the persistent relational data storage over all other SQL-based systems.

- PostgreSQL is a capable relational database management system that is open-source and free.
- PostgreSQL is operated by a committed and knowledgeable team that can be reached for free at any time through learning bases and Q&A pages.
- Regardless of the advanced features, PostgreSQL comes with a slew of fantastic and open-source third-party resources for planning, managing, and using the administration system.
- PostgreSQL can be automatically enhanced with stored procedures, much as a more advanced RDBMS should be.
- PostgreSQL is both a social and a target database administration system, with support for other features.

4.3 Description on implementation of GraphMk

parser-optimizer-executor pattern is used by RMV model to execute query, that is similar to relational querying. A query that follows RMV-SQL conventions and is sent through the graphmk system is consumed by the query console component, which validates it before converting it into a plan tree. The query optimizer will use the generated plan tree to compare plan trees considered as alternative, estimate

costs, and choose most fit plan tree for computaiton. A plan tree (for more details, see Section 6.4.5) includes various types of operation nodes, such as relational operation nodes and graph operation nodes. The query optimizer can extract graph operations from the plan tree and transfer them to graph executors.

All graph operators are computed using three graph executors: (1) Rank operations with the rank executor (3) Route operations with path executor and (2) cluster operations with cluster executor. Graph executors must download graph data from data storage in order to produce graphs and run algorithms over them during these executions. Following the execution of graph operations, their corresponding executors will save the results as graph analysis results in the data storage. From bottom to top, the plan executor processes the activity nodes of the selected best plan tree. During the implementation process, the plan executor must retrieve relational data results and graph analytical results from data storage. Plan executor can submit the results to question console at the end of the execution. As a component diagram, the diagram below shows the details of the components mentioned above that are involved in the graphmk system.

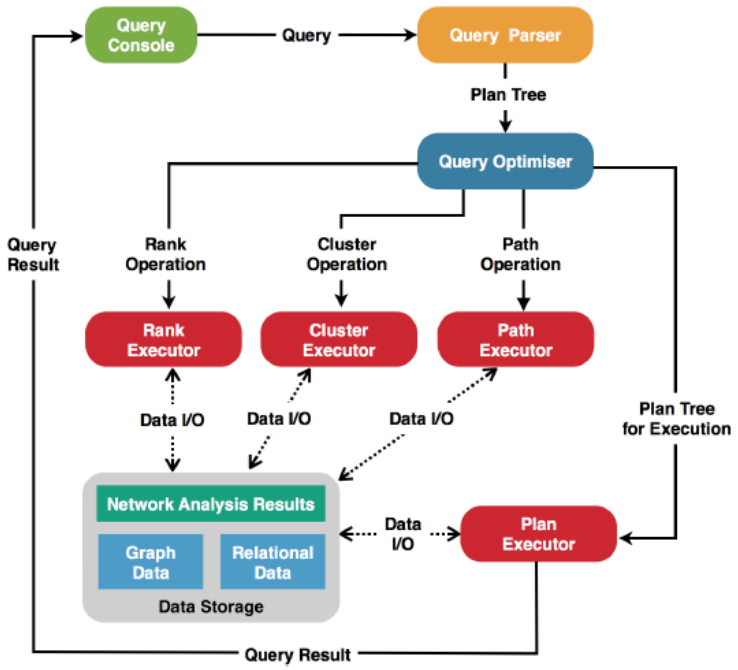


Figure 4.1 Process part overview

4.3.1 GUI implementation

Main user interface window component the user can see while communicating with the device. There are only two sections of this: panel for users to input their query and result rendering panel .

4.3.1.1 Implementation of Key trigger events

The event library, which comes with the Python kit, was used to implement this. The main goal of the project's primary trigger event is to signal the termination of query, clear results panel. This was possible with simple if conditions using Tk manual's key-codes, and it worked. The implementation of main trigger event implementation is given in below figure.

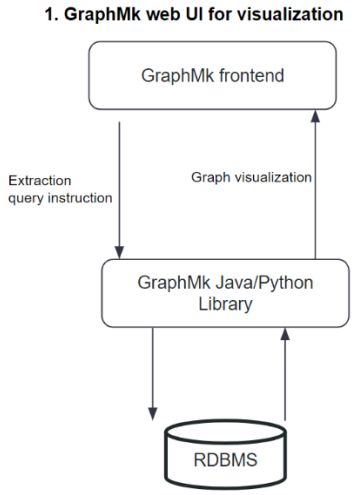
```
#Key click events for Query Input Text
def query_Text_Event(event):
    #Query input is taken once enter is pressed and query ends with char ";"
    if event.keycode == 36:
        query = query_Panel_Text.get(1.0, END).strip()
        if query[-1] == ";":
            queryConsole.start(query, result_Panel_Text)
            query_Panel_Text.config(state=DISABLED)

    #press ESC to clear text
    if event.keycode == 9:
        query_Panel_Text.config(state=NORMAL)
        query_Panel_Text.delete("1.0", END)
        result_Panel_Text.config(state=NORMAL)
        result_Panel_Text.delete("1.0", END)
        result_Panel_Text.config(state=DISABLED)
```

Figure 4.2 Implementation of a primary trigger event

4.3.1.2 Implementation of UI window

To make it more user friendly, a basic UI was introduced with the goal of providing a question panel and a result panel for users. This was possible thanks to the Label Frame built-in by Tkinter library, that was used for additional support apart from Python. Constructor is provided by lable frame function and its name provided as required by developer, and the Label function's grid attribute assists in forming the grid as defined by the developer. Figures 6.2 and 6.3 show how to make a UI panel and a widget.



```
#the root window
root = Tk()
root.title("Heft Framework Console")
root.resizable(FALSE, FALSE)


#The below section creates the Panel for query editor
query_Panel_LabelFrame = LabelFrame(root, text="Query Editor")
query_Panel_LabelFrame.grid(row=0, column=0, padx=25, pady=15)

#The below section creates the Panel for result panel
result_Panel_LabelFrame = LabelFrame(root, text="Result Panel")
result_Panel_LabelFrame.grid(row=1, column=0, padx=25, pady=15)

#The below section creates the Scroll bar for query panel
query_with_scrollx = Scrollbar(query_Panel_LabelFrame, orient=HORIZONTAL)
query_with_scrollx.grid(row=1, column=0, sticky=W+E)
query_with_scrolly = Scrollbar(query_Panel_LabelFrame)
query_with_scrolly.grid(row=0, column=1, sticky=N+S)

#The below section creates the Scroll bar for result panel
result_with_scrollx = Scrollbar(result_Panel_LabelFrame, orient=HORIZONTAL)
result_with_scrollx.grid(row=1, column=0, sticky=W+E)
result_with_scrolly = Scrollbar(result_Panel_LabelFrame)
result_with_scrolly.grid(row=0, column=1, sticky=N+S)
```

Figure 4.3 UI creations in the form of panels

```
#The below section creates the wideget for query panel
query_Panel_Text = Text(query_Panel_LabelFrame, width=100, height=18, wrap=NONE, xscrollcommand=query_with_scrollx.set, yscrollcommand=query_with_scrolly.set)
query_with_scrollx.config(command=query_Panel_Text.xview)
query_with_scrolly.config(command=query_Panel_Text.yview)
query_Panel_Text.grid(row=0, column=0)

query_Panel_Text.bind('<Key>', query_Text_Event)

#The below section creates the wideget for result panel
result_Panel_Text = Text(result_Panel_LabelFrame, width=100, height=18, wrap=NONE, xscrollcommand=result_with_scrollx.set, yscrollcommand=result_with_scrolly.set)
result_Panel_Text.config(state=DISABLED)
result_with_scrollx.config(command=result_Panel_Text.xview)
result_with_scrolly.config(command=result_Panel_Text.yview)
result_Panel_Text.grid(row=0, column=0)

mainloop()
```

Figure 4.4 Widget creation for the user interface

4.3.1.3 Encountered problems

	Problem	Solution
1	Identifying and activating events for the user's clicks was a difficult job, and the solution in Python is unknown.	For assistance, I used online resources and discovered that the Tkinter python package has an inbuilt library called event with key-code that can be used to initiate a desired task when a key is pressed.
2	Creating simple UI windows and images with Python's built-in libraries seems to be difficult.	I did some research online and discovered that Tkinter is a GUI package for Python, as well as PIL and ImageTk, which can be used to build a more sophisticated and enhanced UI.

4.3.2 Query processor implementation

Query processor helps to transfer query inputted by user in GUI for further processing where sequential step starts here. RMV specified query operations are similar to SQL where it ends with a semicolon. Fallback details will also be shown with aid of query processor module

4.3.2.1 Implementation of Input Query

In order to complete this mission, two functions are required. Database connection need to be reinitiated, clear temporary data and tables which were created dynamically, query dictionary and parser needs to be cleared with results from previous execution in the initialization feature. The initialized configurations are then passed as a parameter to the execute function, where utilization of if-else clause to determine the keyword and execute appropriate actions. This can be broken down into three sections: graph execution tasks (rank, cluster, or path), graph formation and deletion tasks, or a standard query. The implementation of the input query method shown below.

```
#starts to execute the input query
def executeQuery(connection, cursor, commandToExecute, query_Result_Text):
    lowerCaseCommand = commandToExecute.lower()

    #graph query contains rank, cluster and path operation
    if ("rank" in lowerCaseCommand) or ("cluster" in lowerCaseCommand) or ("path" in lowerCaseCommand):
        newExecuteCommand = queryParser.queryAnalyse(commandToExecute, connection, cursor)
        cursor.execute(newExecuteCommand[:]) #remove the first space
        printResult(connection, cursor, query_Result_Text)

    #query about creating or dropping a materialised graph
    elif ("create" in lowerCaseCommand or "drop" in lowerCaseCommand) and ("ungraph" in lowerCaseCommand or "digraph" in lowerCaseCommand):
        newExecuteCommand = GraphProcessor.matGraphProcessor.processCommand(commandToExecute, connection, cursor)
        cursor.execute(newExecuteCommand[:]) #remove the first space
        connection.commit()
        query_Result_Text.insert(INSERT, "Graph Operation Done")
        query_Result_Text.config(state=DISABLED)

    #normal relational query without any graph functions
    else:
        #print commandToExecute[:]
        cursor.execute(commandToExecute[:])  #remove the first space
        printResult(connection, cursor, query_Result_Text)
```

Figure 4.5 Input query implementation

4.3.2.2 Print results implementation

Results need to be produced in a way that it's understandable by user, hence tabular format was chosen amd pylsytable package was just and d3.js was used to render graph format. Pylsytable was a simple package where constructor need to be created with column definitions and data can simply populate the rows when given as an array. Package implemented with for loop where it populates each tuple extracting based on indentation and render a proper table for easy usability. After all of the rows and columns have

been correctly formatted and stored within the table, the pylsytable package's table will be inserted into the UI's results panel.

```
#prints results of query execution
def printResult(connection, cursor, query_Result_Text):
    print "Printing results"
    if(cursor.description == None):
        connection.commit()
        query_Result_Text.insert(INSERT, cursor.statusmessage)
    else:
        tabcolnames = [description[0] for description in cursor.description]
        table = pylsytable(tabcolnames)
        rows = cursor.fetchall()
        tabcol_index = 0
        row_num = 0
        for i in rows:
            row_num += 1
            for each in i:
                print str(each) + '\t',
                table.append_data(tabcolnames[tabcol_index], str(each))
                tabcol_index += 1
            print
            tabcol_index = 0

            #printing large result set using pylsytable problem is solved
            if row_num == 100:  #a new table is created to print for every 100lines
                query_Result_Text.insert(INSERT, table)
                row_num = 0
                table = pylsytable(tabcolnames)

        if row_num != 0:
            query_Result_Text.insert(INSERT, table)
        connection.commit()
        query_Result_Text.config(state=DISABLED)
```

Figure 4.6 Implementation of the print result feature

4.3.2.3 Encountered problems

	Problem	Solution
1	The display of results in a readable format is a major issue that needs to be tackled.	I looked online and found the pylsytable external package on GitHub, which allows you to print results in a tabular format in Python. I used it to print the results.
2	There was an issue with pylsytable where it couldn't print a large number of tuples in one table; the cap was 100.	To solve this problem, a new table without a header will be created after 100 tuples have been reached, and if more tuples need to be added into the table, a new table without a header will be created recursively, solving any number of outcomes.

4.3.3 Query Parser implementation

Validator, Analyser, Rewriter, and Translator are the four major sub-components of the question parser. Each part and its implementation will be explained in detail below.

4.3.3.1 Validator implementation

Validator takes an RMV query input where it verifies that query's syntax and format, such as keyword spelling, parentheses, and path expression, are correct. The question is then validated using the device catalog and validator. In the system catalog, Postgresql which will help to store schema and metadata. The pg matgraph project adds schema meta data for materialized graphs to the framework catalog.

```
#avoid repeated graph operation again
if graphQuery not in graphQueryAndResult:
    resultTableName = "rank_" + rankCommands[0] + str(operatorID)
    rankCommands.append(resultTableName) #the last element is the result table name
    rExe.processCommand(rankCommands, conn, cur)
    graphQueryAndResult[graphQuery] = resultTableName
```

Figure 4.7 Validates for graph computations that are repeated.

4.3.3.2 Implementation of Analyzer

The analyzer starts separating graph sub-queries and relational sub-queries after a query is accepted. A question tree will be shown, demonstrating the query execution request, with the base queries being executed first. The "Diagram Sub-query 1" recovers an emerged graph in Figure 4.3, while the "Diagram Sub-queries 2" with a relational sub-query recovers a dynamic graph.

```
#rewrite the queries
for eachString in graphQueryAndResult.keys():
    executeQueryCommand = executeQueryCommand.replace(eachString,graphQueryAndResult.get(eachString))
return executeCommand
```

Figure 4.8 Query rewriting code snippet

4.3.3.3 Implementation of Translator

Translator will help to convert queries into plan trees which are an internal format not exposed to user, which is then transferred to the query optimizer for optimization. A relational algebra expression can be used to describe a plan tree.

4.3.3.4 Encountered problems

No problems were encountered

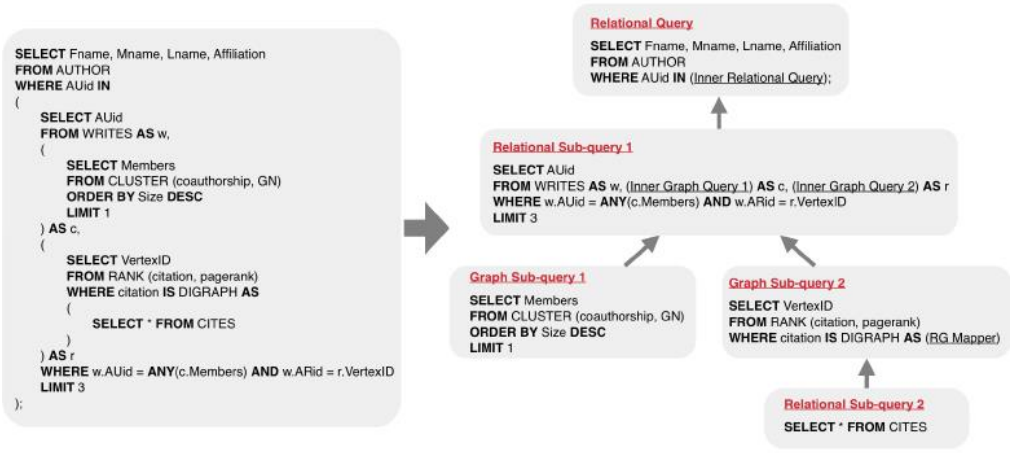


Figure 4.9 Details on graphmk plan tree for queries

4.3.4 Implementation of Query Optimizer

In general, the query optimizer performs the following tasks:

- Evaluate different combinations of plan tree generation for provided plan tree passed by query parser after generation.
- Estimates the cost of alternative plan trees that it generates.
- Then compare and conclude with less cost executable plan tree for execution.

Query optimizer in Graphmk is used to identify alternative plan trees that can be considered efficient to execute, where this will be analyzed from sets of available trees and it will follow heuristic rules.

Transformation rules would possibly include few of the below executions, selection operations are deconstructed to get sequence of selections in an individual manner, execute selection before joins and combine selections into joins. Once alternative plan trees are defined, optimizer would run a cost optimizer checker where it will look into cost of each plan trees in respect of resources needed for I/O and CPU processing time. Strategy to use will then be selected and used. Optimizer also executes extraction and identifying of graph operations and transfer them to graph executors.

```
#retrieve the sub-query
def getSubQuery(whereClauseCon, indexCon):
    leftBracketArray = []
    bracketPair = []
    foundMapper = False
    pointerIndexVal = indexCon + 2

    while foundMapper == False:
        pointerIndexVal += 1
        if whereClauseCon[pointerIndexVal] == "(":
            leftBracketArray.append(pointerIndexVal)

        if whereClauseCon[pointerIndexVal] == ")":
            bracketPair.append((leftBracketArray.pop(),pointerIndexVal))
            if len(leftBracketArray) == 0:
                foundMapper = True
    subQuery = whereClauseCon[bracketPair[-1][0] + 1 : bracketPair[-1][1]].strip()
    return subQuery
```

4.3.4.1 Encountered problems

No problems were encountered

4.3.5 Details on different graph executors of graphmk

According to their operation styles, According to their operation styles, currently graphmk is designed with three graph operators that are built on top of vertex centric interface implementation graphmk provides. Users can simply use three of these graph operators that includes rank, cluster and path executors involved in rank execution, cluster execution and path execution operations respectively. Three graph analysis software, SNAP, NetworkX, and Graph-tool, were used as algorithm support for the graph operation executors. Based on the results of these three graph analysis tools' performance evaluations,

```
if len(graphInformation) == 3:  #info about creating graph
    graphFileInfo = open(tmpGraphDirectory + graphInformation[0], 'w')
    cursor.execute(graphInformation[2] + ";")
    conn.commit()
    rows = cursor.fetchall()
    start_exec_time = time.time()
    for i in rows:
        graphFileInfo.write(str(i[0]) + '\t' + str(i[1]) + os.linesep)
    graphFileInfo.close()
    print "Graph writing time: ", time.time() - start_exec_time
else:
    raise RuntimeError, "Error about creating Dynamic Graph!!"
```

4.3.5.1 Rank Executor implementation

Snap library was chosen to execute ranking algorithms including pagerank, outdegree, indegree and degree. Graph-tool was involved to get support on algorithms on closeness and betweenness.

Algorithms	Methods	Tools
Degree **Indegree** **Outdegree** **Pagerank**	GetDegreeCentr( ) GetNodeInDegV( ) GetNodeOutDegV( ) GetPageRank( )	SNAP
Betweenness **Closeness**	centrality.betweenness( ) centrality.closeness()	Graph-tool

4.3.5.2 Implementation of Cluster Executor

Choose Graph-tool for the Monte Carlo algorithm [55] and SNAP for four clustering algorithms (the Clauset-Newman-Moore algorithm [43], finding connected components [57], the Girvan-Newman algorithm [51] and finding strongly connected components [59]).

Algorithms	Methods	Tools
Connected Component **Girvan-Newman** **Clauset-Newman-Moore**	GetWccs( ) CommunityGirvanNewman( ) CommunityCNM( )	SNAP
Monte Carlo	community.minimize_blockmodel_dl( )	Graph-tool
Path Algorithm	allsimplepaths()/allsimpleshortestpath()	NetworkX

4.3.5.3 Implementation of Path Executor

NetworkX graph analytical library was chosen to support implementation of path executor for graphmk. The use of NetworkX for Path computation is justified in Appendix G. Even though other sub-processes are involved in the process' completion, this implementation consists of two main sub-processes. The implementation of the path symbol analysis function is shown in the screen shot below. This function

extracts the path portion of the command and analyzes the user-specified path choice. It then divides path segments into divisible components.

The following diagram depicts the networkX API interaction for computation implementation. The following screenshot depicts the interaction with the graph that was generated for the data, as well as the source and destination that were separated from the user-specified direction. Then rest of them were passed into networkX for computation of different nature specified by user for path based scenarios.

Algorithms	Methods	Tools
Path Algorithm	allsimplepaths()/allsimpleshortestpath()	NetworkX

4.3.5.4 Encountered problems

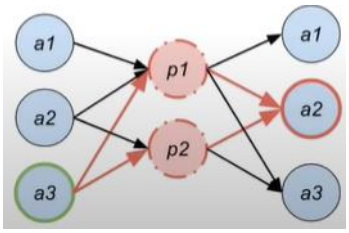
No problems were encountered

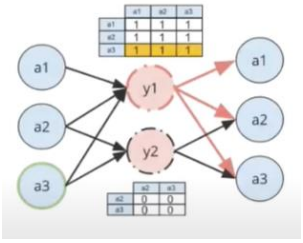
4.3.6 Details on Plan Executor implementation

Plan executor is responsible in getting optimized plan tree from query optimizer module and start executing plan tree. During execution it needs to get relevant table records and send it back to query handling module. Normally a plan tree would consist of different operation types within graphmk system and they will be executed iteratively using plan executor. Plan tree execution happen in bottom to top order where results generated from bottom operation will be used in its consecutive operations nodes. Lower nodes in a plan tree resembles selection execution most of the time where as consecutive upper nodes mostly will consist of sorting, aggregating operations. The image below reflects such plan tree created to find top 10 writers authorship graph data.

4.3.6.1 Encountered problems

Duplicate edge identified as problem encountered and that was solved with an efficient Bitmap approach.



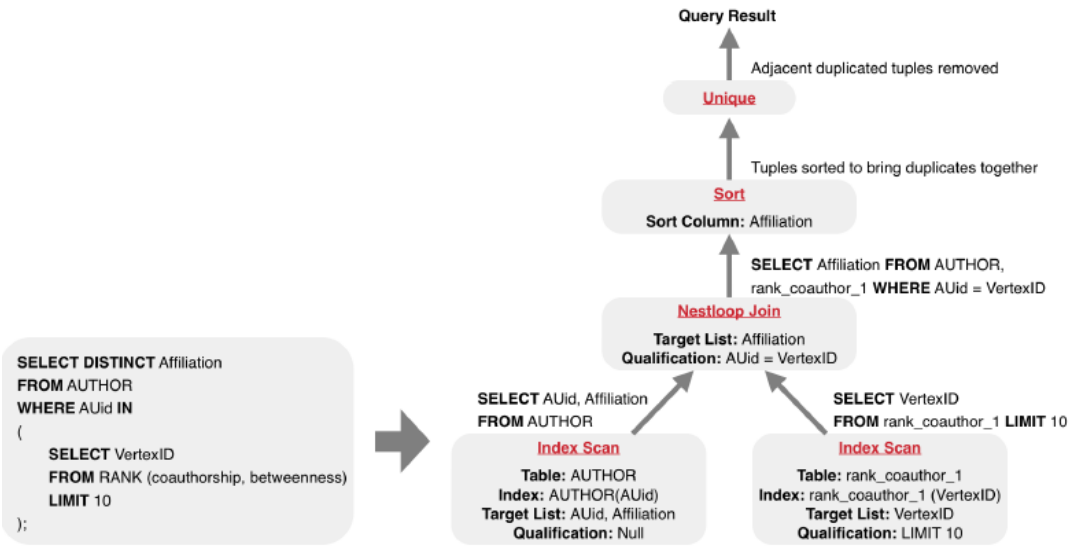


4.3.7 Summary on process involved in Graphmk

In graphmk system, RMV model-based queries will have following manipulation steps during process of graph computation,

- A validator to verify and validate queries.
- An analyzer which extracts graph creational and executional queries.
- Rewriter helps to rewrite graph queries in an understandable manner for RMV model
- Query parser involved in converting rewritten queries into a plan tree to identify cost and efficiency with RMV model

The query optimizer enumerates alternate plan trees based on a query parser's plan tree, calculates their costs, and chooses the plan tree with the lowest cost to execute. The plan executor is responsible for execution of operations within each node in execution plan tree, and as a last step in plan executor any graph analytical algorithm execution takeplace within graph operation interface implementation based on extracted graph query operation in plan tree. Finally, query output is returned to the query console by the plan executor.



4.4 Summary of the Chapter

The implementation of the graphmk graph computation engine was discussed in detail in this Chapter. The selection of operating systems was the first step in the Chapter. Then, for the graphmk graph computation engine, the development environment and programming language is justified. Later steps addressed extensions and libraries involved in creation of system, as well as how to choose and use them. Lastly, implementation process was discussed in detail with problems that were encountered and solutions taken along with code snippets and flow diagrams.

5 Testing

Contents

- Overview of the Chapter
- Purpose of testing within GraphMk
- Criteria for testing within GraphMk
- Testing stages within GraphMk
- Description on environment to conduct test
- Testing functionalities within GraphMk
- Integration testing perspectives for GraphMk
- Testing non-functionalities within GraphMk
- Details on testing limitation
- Summary of the Chapter

5.1 Overview of the Chapter

Implementation section of proposed Integrated Graph Computing Framework – GraphMk System was discussed in the previous Chapter. This Chapter focuses on evaluating the implemented system's functional and non-functional specifications, with the goal of ensuring that all implemented requirements reach the anticipated standards. The Chapter begins with an overview of research objectives and priorities, testing techniques, testing requirements, and testing levels, as well as an assessment on execution output.

5.2 Purpose of testing within GraphMk

To validate if intended specifications are met by the software package functions and if those functions fit the requirements perfectly, software testing process will be performed.

- Check and confirm that technical specifications have been incorporated and are functioning properly.
- To check to see if non-functional specifications have been enforced and are operating.
- Verification and detection of flaws and errors in the system should be carried out in order to keep the number of bugs and errors in the finished product to a minimum.
- Based on the test results, improve the device even more.

5.3 Criteria for testing within GraphMk

Testing is the method of validating an application that has already been implemented in order to ensure that it meets functional and non-functional specifications as well as consistency standards. Testing can be used to assess software quality, and the following parameters can be used to do so.

5.3.1 Graphmk functional assessment and software quality

Specifically focuses on functionality development of the product, including technological specifications for a provided blueprint, built based on requirements specified.

5.3.2 Graphmk structure assessment and quality

Process that assess functional and non-functional requirements stated for GraphMk in combination.

5.4 Testing stages within GraphMk

5.4.1 GrapMk testing

Process in which all of the components are evaluated together, implying that the final device generated is tested. graphmk – Graph computing engine will be used for this testing.

5.4.2 GraphMk integration

Module testing is the process of independently testing the various modules that make up the device. Individual functional modules need to be tested and once they are assured that it performs as expected, it can then be integrated as required and tested as module integration. This practice need to be done before device integration and its important that GraphMk as a system to function its individual components need to be tested for its standalone functionalities

5.5 Description on environment to conduct test

5.5.1 Hardware Information

Experiments were conducted on personal laptop with specification of following.

- Asus Intel core i7-9750H
- 2.60 GHz
- 12 GB and 12 processor threads

5.5.2 Software Information

Operating System	Windows Subsystem for Linux
Programming Language	Python 3.9
Relational Database	PostgreSQL
Graph Database	Neo4j

Graph Analysis Tools	Snap.py Graph-tool NetworkX
PostgreSQL Adapter	psycopg

5.6 Testing functionalities within GraphMk

5.6.1 Functional testing method

The approved technique for development, spiral methodology, lets for versatility with independence on evaluating software in parallel with the implementation process. As a result, the black box testing technique was used to evaluate the implemented functional specifications in tandem with the device implementation.

The results of the functional requirement testing are summarized in Table 7.1. Test cases and extensive descriptions can be found in testing appendix.

Functional Requirement	**Pass Rate**	**Status**
Every type of data set should be accepted as database input by the framework.	**0%**	**Fail**
Convert such data sets into relational tables an ETL operation.	**50%**	**Pass**
To perform graph analytical processes, graphs are created from relational schema.	**100%**	**Pass**
Based on a subgraph-centric approach, graph data should be partitioned as subgraphs of similar data. Each one is a Subgraphs of meaning that are able to be computed	**100%**	**Pass**
Executing cluster graph analytical computation using external graph libraries	**100%**	**Pass**
Executing rank graph analytical computation using external graph libraries	**100%**	**Pass**
Executing path graph analytical computation using external graph libraries	**100%**	**Pass**
The user should be able to perform basic SELECT queries.	**100%**	**Pass**
NESTED Joins should be computed by the user.	**100%**	**Pass**

The final output should provide the user with useful details.	**100%**	**Pass**
AGGREGATE functions should be able to be computed by the user.	**100%**	**Pass**
The prototype's user should be able to add, uninstall, and change graph analytical frameworks that are used as plugins.	**0%**	**Fail**
Adopt to integrate easily external graph analytical platforms as plugins to do specific executions.	**50%**	**Pass**
Interface to define and implement their own iterative graph computation by users	**50%**	**Pass**
More user friendliness by having a nice frontend UI like a dashboard where all functionalities can be carried out through a UI.	**50%**	**Pass**
Export results into some format that can be used for future or futher computation	**50%**	**Pass**

Table 5.1 Functional testing statistics

5.7 Integration testing perspectives for GraphMk

In its architecture, the GraphMk system consists of several modules, where modules are coupled and dependent on other module for input information, but rather independent on implementation logic it needs to perform. Since one module's functionality requires another module to perform individual operation. Integration testing end to end has be agreed as one of the approach to conduct testing along with individual module testing. Each module was checked ten times, and table 7.2 summarizes the system's unit and integration testing performance.

Module	Input	Expected Output	Actual Output	Status
Web app	User-supplied input information/query	The query's output information or any related fallback messages	Output results were shown and error where appropriate.	Passed

Query processor	Query from web app used as input for query module	Gets input and sends it for processing and returns the results returned after processing to web app visualization.	Module formats and manipulates data as needed from processor in backed and webapp when response received	Passed
Graph processor	Graph creation and deletion queries are sent.	It should generate or remove graphs from the database as specified in	It generates or deletes the graphs based on the query parameters.	Passed
Query Parser	The graph manipulation queries are retrieved from the	The database optimizer module receives the authenticated, evaluated, and rewritten query.	Works as expected where it gets the rewritten query and sends it to optimizer.	Passed
Query optimizer	The Query parser module has rewritten graph manipulation queries.	The question is sent to the Graph operator along with the generated plan tree and cost estimator.	Plan tree creation will be done based on cost estimations and will send it to graph operators.	Passed
Graph operation interface	Gets the Query optimizer module's rewritten plan tree graph manipulation queries.	It should use algorithms (Rank, Path, or Cluster) to carry out the specified process and send results to query processor to display results as expected.	Runs the query's ranking, path, or cluster process and query processor will get results from interface that needs to be formatted and displayed.	Passed

Table 5.2 Details on testing modules and integration for graphmk

5.8 Testing non-functionalities within GraphMk

5.8.1 RMV Engine evaluation

Queries representing different functionalities of graphmk were generated through datasets of three types in this experiment. The RMV model was compared with graph database and relational database to assess

its perspective of supporting both attributes after processing these queries. Data sets used in the testing and their nature was explained below. Later, details on different queries that were created for different scenarios were discussed which will give useful information for user. Experiment results were analyzed finally for a better understanding and comparison.

5.8.1.1 Details on datasets used

Testing will be conducted on three different data sets which includes, flicker graph, amazon movie review graph and stackoverflow graph. Description regarding nature of graph and its stored mode will be discussed in the below table. Most of them were stored in XML and TXT formats where ETL needs to be performed to convert them into relational tables and python scripts need to be created for this purpose.

The following method was used to assess all four non-functional criteria. Postgresql (relational system) and spark-mazerunner were chosen as the frameworks to compare (graph system). To check scalability nonfunctional requirements, different scalable datasets were used, time taken for execution was calculated in all three frameworks and compared for performance nonfunctional requirements, and the same query was run in all three frameworks for the same dataset to check framework accuracy. Finally, the adaptability and usability of various in-depth difficulty querying scenarios were tested with a total of 12 queries (12 scenarios).

5.8.1.2 Details on queries used

Based on the three datasets listed above, 12 queries were generated that can be divided into three categories. More information about the queries can be found in the tables below.

- Queries 1-3 are relational and it involves mostly join, aggregate and sort
- Queries 4-10 are complex queries that involved graph computations like pattern matching, rank , path and cluster detection.
- Rest of the queries have combined graph computations

	Raw Data Size	Number of Vertices in Neo4j	Number of Edges in Neo4j	Number of Records in PostgreSQL
Amazon Network	8GB (XML)	1,128,243	2,488,849	Reviews : 8,000,000 Users : 750,000 Movies : 10,000
ST Network	30.6 GB (XML)	1,713,109	1,747,662	Question : 2,000,000 St Answer : 3,684,117 Network Tag : 38,205 Labelled By : 3,466,686
Flicker Network	295 MB (TXT)	250,200	368,800	Users : 2,000,000 Co-related : 4,000,000

Table 5.3 Three main datasets and its nature information

5.8.1.3 Experimental Results

Three query engines were used to evaluate all of the experiment queries. PostgreSQL, on the other hand, is unable to execute Queries from 6 to 12. Due to limitation in computing iterative logic within relational query nature. Queries 1 to 5 was not able to execute by neo4j and queries from 10 to 12 as well. Because Cypher's restricted expressive power: All of these queries can be represented in RMV-SQL and processed by the RMV engine, which is a benefit.

Sorting + Join Operations		
Scenario 01	Stackoverflow	Output results of most viewed questions with attributes of question, owner and tag.
Sorting + Join + Aggregate Operations		
Scenario 02	Stackoverflow	Output results of top 5 answerers whose answers were accepted and viewed recently in descending order.

Aggregate + Join + Set + Sorting		
Scenario 03	Movie review	Output results on review number for every film and title in descending

Table 5.4 Scenarios with basic relational query computation

Pattern Matching		
Scenario 04	Tweets	Find friends of friend for twitter users.
Triangle Counting		
Scenario 05	Movie review	Find list of reviewers mutually commenting
PageRank Centrality		
Scenario 06	Movie review	Output results of 10 most prominent reviewers in the co-commenting graph based on their pagerank centrality.
Connected Component		
Scenario 07	Movie review	Count how many linked components within mutually commenting.
Path Finding		
Scenario 08	Movie review	Find connection between two reviewers who reviewed two different movies and within mutual commenting circle
Shortest Path		
Scenario 09	Movie review	In the co-commenting graph, find the shortest path for any two reviewers.
Community Detection		

Scenario 10	Stackoverflow	Find a set of labels that are commonly used together to indicate a query.

Table 5.5 Scenarios with graph analytical query computation

Connected Component + PageRank		
Scenario 11	Stackoverflow	Output top three reviewers who are within the co-commenting circle and computed with pagerank.
PageRank Centrality + Path Finding		
Scenario 12	Stackoverflow	Show how top two reviewers in the co-commenting graph are linked based on pagerank centrality.

Table 5.6 Scenarios with complex query computation

	PostgreSQL	**RMV Engine**	**Neo4j**
Scenario 1	✓	✓	✓
Scenario 2	✓	✓	✓
Scenario 3	✓	✓	✓
Scenario 4	✓	✓	✓
Scenario 5	✓	✓	✓
Scenario 6		✓	✓
Scenario 7		✓	✓
Scenario 8		✓	✓
Scenario 9		✓	✓
Scenario 10		✓	

Scenario 11		✓	
Scenario 12		✓	

Table 5.7 Summary of testing results for three different query engines

Aim of queries discussed above with some useful scenarios and datasets is to compare RMV engine that was built for this thesis. These experiments were carried out with the aim of comparing the established systems postgres, neo4j and RMV model in order to evaluate their time efficiency. Neo4j's mazerunner extension, is used to execute Queries 6 and 7.

Each query was run ten times for the time performance evaluation, and the average time was used to plot the results. For Queries 1–5 and Queries 8–9, the time from when the question was sent to when the results were received was reported. Since Neo4j uses mazerunner and interacts with it by sending a GET API call will trigger execution of graph algorithm within extension, request and query processing times were taken into consideration in such scenarios. RMV model is a bolt-on solution where scenarios relating to relational queries it would perform same as postgres query engine because underlying query engine is same. Check below figures to have an idea on comparison. For most part, the RMV engine has outperformed the competition. This is mostly because of the following factors.

- Since it inherits query optimization techniques from relation databases, the RMV engine has a performance advantage for relational queries. This feature contributed to improved performance for Queries 1–3. Relational query outperforms in triangle counting scenario due to the capability of three-way-self-join by [45] that is reflected .
- Mazerunner extension for Neo4j is key component that helps to compute graph algorithsm with help from Spark system. As a result, Queries 6–7, which require the use of a mazerunner, take longer to execute algorithmically and complete requests.
- Neo4j is entirely designed for computations such as hyper-connectivity on graphs (Sherif et al. 2012). Query 4 is used for the pattern matching operation. Finding a course is the subject of questions 8 and 9. These types of tasks, which involve navigating hyper-connectivity on graphs, are handled well by neo4j but not by the RMV engine because query optimization techniques have not yet been implemented.

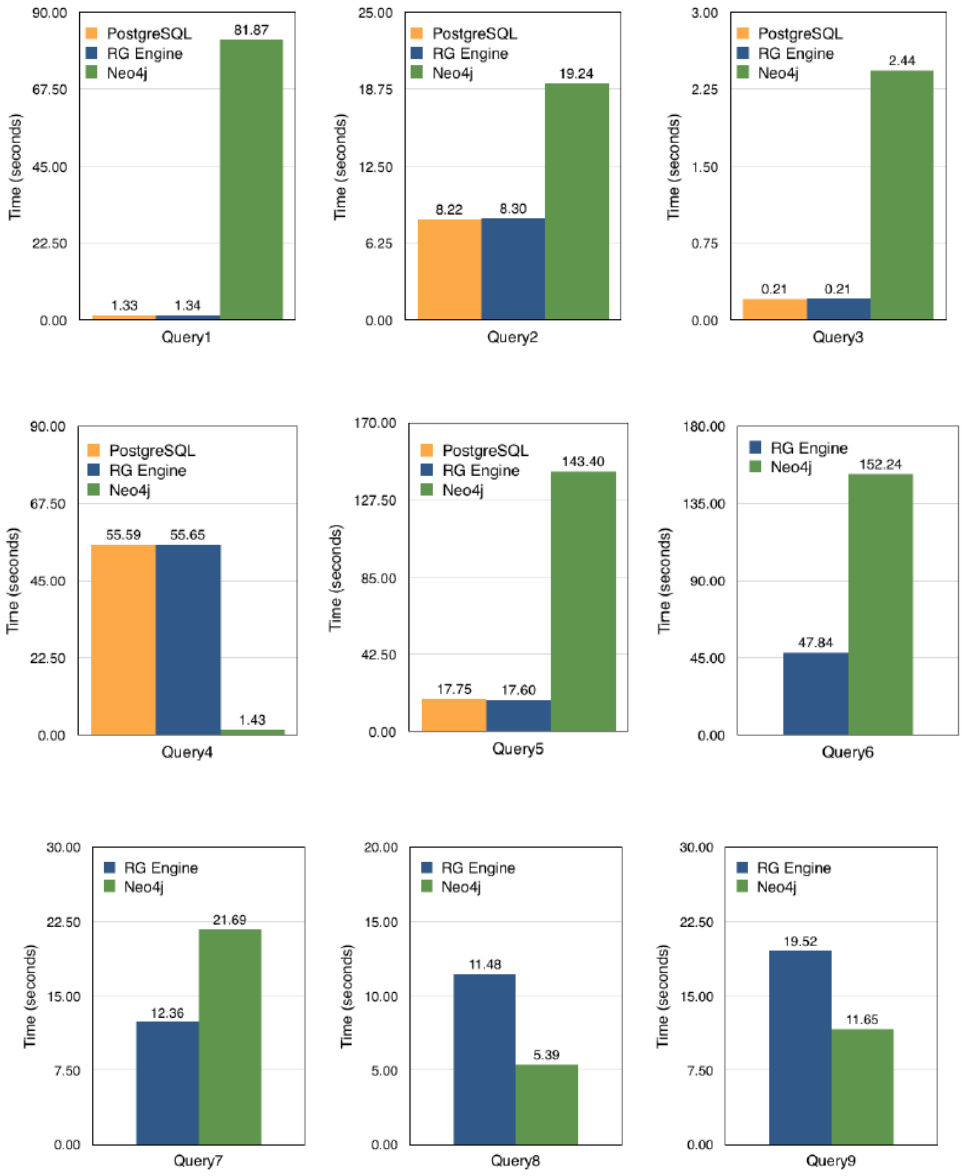


Figure 5.1 Statistics on query execution.

Apart from the above-mentioned query executions, results had unexpected turn when it comes to closeness algorithm execution in twitter dataset and movie review. RMV engine successfully executed those relevant queries, while Neo4j was not able to execute them because of system-reported "out-of-memory" failure. Probable cause is that the edges in the specified graph are too wide, exceeding Neo4j's memory limit and the hardware device on which all of these executions were performed. Appendix on testing contains a graph visualization of the remaining evaluation outcomes.

5.9 Details on testing limitation

- **Testing hardware limitations**

Performance and scalability of graphmk system was testing in a single node/laptop instance with considerable configuration. It could have been enhanced with a better hardware configuration which includes better RAM and SSD for memory utilization. This reflects hardware used for testing is directly proportionate to performance of system. But testing was performed to an acceptable stage.

- **ETL program to adopt relational schema**

ETL operation was mandatory through python script for a successful execution of testing. Datasets were available in text document or xml format where relational database cannot store them and export/ETL mechanism externally needed for data manipulation.

5.10 Summary of the Chapter

Section discusses prototype based testing approach, beginning with testing goals, testing requirements, levels of testing, methods of testing, and environment to test. Prototype testing views were mainly focused in the Chapter, and it starts outlining different phases of testing, requirements, methods and levels. Two major testing methods discussed are functional and structural quality testing. Functional testing, module testing, and quality testing were tested using a black box testing methodology. Following an interpretation of the testing results, an estimated amount was achieved with this testing. Computer structural consistency testing was used to assess non-functional specifications. Non-functional requirement testing was defined as consistency, accuracy, scalability, and memory management testing. The overall quality of the software structural testing process was satisfactory. The evaluation portion, which follows, will explain the project's evaluation process based on different evaluation criteria.

6 Evaluation

Contents

- Overview of the Chapter
- GraphMk project criteria for Evaluation
- GraphMk – Evaluator selection process
- GraphMk – Details on methodology for Evaluation
- GraphMk – Details on evaluation questionnaire and format
- GraphMk – Details on findings from survey
- GraphMk – Details on self-assessment
- Summary of the Chapter

6.1 Overview of the Chapter

The testing process was discussed during previous section to ensure that defined functional and non-functional requirements are working as expected. This section mainly focuses on evaluators of the prototype, methods of evaluation it and validate the defined assessment criteria for evaluation where the mentioned above in a constrained process will help to define the project completeness. The strength and weakness of the Project is also evaluated through a self-validation process.

6.2 GraphMk project criteria for Evaluation

Table below defines the evaluation requirements for the project which would reflect all the main phases and its completeness comparing the defined scope in previous chapters.

Criteria	Description& purpose
The project's overall definition	Critique of the overall concept of this research project is critical for evaluating the project, and it is expected that criticism in the form of feedback, opinions, and assessment would aid in proving the accuracy of the concept.
The project's scope and depth	It's important to get opinions and feedback from industry experts on the reach and depth of the project. Since both the Big Data and Graph Computing domains are so large, it is crucial to speak with domain experts to get a better understanding of the scope.
Design, architecture, and implementation of a system	Each module's design, layout, and implementation should be thoroughly evaluated, with justifications.
Prototype and Solution	To assess and judge if the prototype acts as a proof of concept, it should be evaluated and justified.
The prototype's usability, efficiency, and accuracy	It is necessary to assess the process scope of the implementation of non-functional specifications.
Framework for Integrated Graph Computing	The effectiveness and suitability of the proposed integrated graph computing system for the hypothesis must be assessed.

Future improvements and limitations of the solution	Future enhancements, as well as the recognition of the graphmk system's shortcomings, are extremely necessary.

Table 6.1 Graphmk project criteria's for evaluation process

6.3 GraphMk – Evaluator selection process

Different categories of evaluations mentioned below have been defined for the project's assessment process, with a greater emphasis on experts in graph computation systems due to the project's emphasis on the integrated graph computation framework. End users are also critical since the established system would be used by them in the end. Table 8.2 provides a detailed overview of the evaluator selection process.

Evaluator Category	Description & criteria evaluated
End users	People who have experience in Software engineering and business analyst roles were chosen to evaluate the overall performance and usability of the system, since they can have a very broad perspective.
Database experts	A few experts in the field of analytical analysis were chosen to test the prototype and address possible improvements.
Graph framework Developers	The system's design, architecture, and implementation were evaluated by experienced graph analytical framework developers.
Experts with domain knowledge in graph analysis	Graphmk was evaluated by experts in graph analytical systems where their feedback was valuable along with enhancement suggestions.

Table 6.2 Graphmk evaluator selection process

6.4 GraphMk – Details on methodology for Evaluation

The progress of the system remains determined only by assessment process, that includes information primarily focusing on phases of project development life, implementation challenge and review on design. Assessment was conducted based on quantitative and qualitative approach. Project report's testing portion, for the most part, aids in quantitative assessment. This Chapter puts a stronger focus on the s qualitative assessment of the project, which was performed predominantly via questionnaire and interviews. Questionnaire was developed providing general philosophy of the project, as well as quantitative and qualitative measurements of the system's criteria. It was then circulated to users of the system for input. Expert evaluators were able to provide their feedback through email and short video calls on areas of graph algorithm implementation, vertex centric interface implementation and how qualitative the system is in the perspective of its execution and threshold. The quantitative and qualitative dimensions of the general design, prototype and scope of device were assessed via the requirement collection phase with experts and end users.

6.5 GraphMk – Details on evaluation questionnaire and format

Table below summarizes the questions raised to evaluators for them to carry out the evaluation process.

No	Question
Knowledge and experience related questions	
1	State number of years of experience in the field of Big Data analytics and parallel systems?
2	State number of years of experience in the field of graph analytics and computation?
Understanding of Graphmk project concept	
3	State initial impression of project idea as and when you heard?
4	State impacts that this project would provide on the selected user groups?
Understanding of Graphmk project scope	
5	State the relevance of complexity of the project to a Master's thesis?
6	To what extent could the graph computation have been discussed in the solution?
Understanding of Graphmk project design and implementation	
7	State personal comments on design and implementation of graphmk system?
8	State whether decisions taken throughout project phases do have a justifiable reason?
9	State any further improvements that you would suggest for graphmk system?
Understanding of Graphmk prototype	
10	State whether proposed prototype solution will solve the problem stated?

11	State whether solution is complex enough to understand or not?
12	State personal opinion the features of the system?
Understanding of Graphmk prototype performance	
13	State the prototypes completeness in the perspective of usability and efficiency?
Recommender Engine	
14	What are your thoughts on the graphmk system's hybrid approach?
15	What aspects of the given graph computation engine need to be improved?
Limitations of the solution and future enhancements	
16	What are the solution's general shortcomings, and any alternative thoughts to that?
17	State useful features of graphmk project that will benefit mostly?

Table 6.3 Description on categories and questions within questionnaire

Refer appendix for detail explanation of questionnaire.

6.6 GraphMk – Details on findings from survey

Evaluator feedback and suggestions were represented in summary structure.

6.6.1 Opinions on Concept

"As a two-in-one device with a variety of unique features, this system will probably createefficient computations that will save time"

Dr. Antonis Michalas (PhD)

University of Westminster, London

"I assume approach can set patterns which provide a roadmap for peer candidates in same research field to investigate various perspectives of integrating graph analytical process into relational computation framework."

Mr. Chamil Jeewantha

Software Architect, Zone 24x7

"Research topic and prototype presented have promising level of acceptance where the research problem addressed is a vital scenario within graph analytical domain."

Prof. Alex Hughes (PhD)

University of Westminster, London

Evaluation module	Summary of feedback
concept	Design of the project received positive reviews from the evaluators. The majority of them believed that this approach will save them time and effort, and they werevery pleased with the integrated system definition. Experts in the field were enthusiastic about the financial advantages where system could contribute to graph analytical perspective.
Review on feedback	Idea of an integrated solution was appreciated by many evaluators where they mentioned efforts taken are accountable to build the system. This solution would help to make life easier for many analyses in different disciplines to make use of graph computation functionalities

Table 6.4 Description on Graphmk system overall feedback

6.6.2 Opinions on Scope

"Graphmk system is extendable and more research insights can be derived in future, but now it's also useful for different stakeholders of data based systems which is very promising."

Dr. Antonis Michalas (PhD)

University of Westminster, London

"Theoretical breadth is very impressive within the system, and this has covered enough standard for a masters research."

Prof. Alex Hughes (PhD)

University of Westminster, London

"Graphmk system seems to be a topic that consist of scope from a broader area. Application of concepts for Master's study seems acceptable."

Prof. M.E. Paul De Bra (PhD)

Eindhoven University of Technology, Netherlands

Evaluation module	Summary of feedback
Scope	As a general comment from most of evaluators the project scope seems sufficient enough for master's research standard. The elevators also praised the solution's application of the integrated system concept.
Review on feedback	Since graph analytical systems is a large topic with a lot of scope, there was some debate about how deep the study should go. As feedback provided from domain experts it is mentioned that project meets standard complexity of a master's study.

Table 6.5 Detail feedback on scope of the project

6.7 GraphMk – Details on self-assessment

6.7.1 Implementation of functional requirement and Status

Justification on requirements that are not covered will be discussed in the appendix section along with self-evaluation.

Functional Requirement	Priority	Status
Functional Requirement 1	Critical	GraphMk supports the requirement
Functional Requirement 2	Important	GraphMk supports the requirement
Functional Requirement 3	Critical	GraphMk supports the requirement
Functional Requirement 4	Critical	GraphMk supports the requirement
Functional Requirement 5	Critical	GraphMk supports the requirement
Functional Requirement 6	Critical	GraphMk supports the requirement
Functional Requirement 7	Critical	GraphMk supports the requirement
Functional Requirement 8	Critical	GraphMk supports the requirement

Functional Requirement 9	Critical	GraphMk supports the requirement
Functional Requirement 10	Critical	GraphMk supports the requirement
Functional Requirement 11	Critical	GraphMk supports the requirement
Functional Requirement 12	Desirable	GraphMk supports the requirement
Functional Requirement 13	Desirable	GraphMk supports the requirement
Functional Requirement 14	Important	GraphMk supports the requirement
Functional Requirement 15	Desirable	GraphMk supports the requirement

Table 6.6 Graphmk status of Functional requirement implementation

6.7.2 Details on challenges

Several challenges were encountered throughout the project having both ups and downs. To address the challenges, a large amount of research material was consulted, and input from experts in different fields was obtained; Due to this, project plan required frequent updates to support extra effort and time required to resolve issues that were raised. According to the author, the project can be considered to be highly successful because the system built does support many different graph algorithms and have an interface that will help developers to implement their graph algorithm logic. System is not scalable but it shows capacity to handle large graphs with precision and at last UI built is useful for any users to have a better visualization. The novelty of the system has been validated by testing and domain expert reviews, but there are a number of potential future enhancements that can be integrated into the system (see Chapter 9.7) to further improve accuracy and scalability. Many academic and technological principles, as well as soft skills, were learned or improved during the project and possibly these skills can be utilized in future.

6.8 Summary of the Chapter

The section describes evaluation methodology, the different criteria selected for evaluation and evaluator selection considering domain knowledge and usability knowledge depends on various phases of integration within Graphmk application and provides justification for every decision that was taken. the questionnaires and interviews were selected as the assessment methods based on the justifications.

The reviews and analysis outcomeson assessment results comes next. Based on feedback received, general design of the project seems stable and has flexibility for future enhancements, and completeness of the project was praised by the evaluators. When evaluating the feedback received in detail, it is clear that the majority of them stated that the scope fits appropriately for a Master's research and was presented with high quality. The integrated system's architecture, as well as the technologies chosen for implementation, were lauded by the majority of the evaluators. Evaluators were quite surprised that solution works well on a non-functional aspect even if it monolithic and they were in an acceptable level. Experts mentioned that they were pleased with graph mapper views and acknowledged rationale presented for it. They assume that this might be beneficial in some way. Many people complained that it couldn't handle fault tolerance and scalability well, many believe because it’s not distributed it cannot solve the scalability issue but it fairly does. These legitimate factors have been listed as possible enhancements forthe future. The Chapter ends with a self-evaluation that addresses all facets of the project that were completed successfully. The following Chapter brings the project to a close by adding a conclusionbased on the above..

7 Conclusion

Content

- Overview of the Chapter
- Details on Purpose & Aim achieved
- Application of Knowledge from course modules
- Enhanced Existing Skills
- Experiences from Project
- Details on Challenges encountered
- Details on Limitations from the research
- Possible Future work for GraphMk
- Contributions from GraphMk
- Closing comments

7.1 Overview of the Chapter

The assessment method for the GraphMk System, as well as an overview of the results, were presented in the previous Chapter. This section concentrates on bringing the development to a closure by mentioning the accomplishment of the project's goals and purposes, the issues and difficulties encountered during the project's life cycle, project constraints, acknowledged potential improvements, and final comments.

7.2 Details on Purpose & Aim achieved

7.2.1 Aim

To investigate, design, build, test, and assess an integrated system for creating, computing, storing, and using actual data as graphs while conserving resources and performing well.

Within limited time allotted for this research project, the goal was achieved successfully. Guidance from experts, users, and evaluation, the prototype helped largely to test the prototype qualitatively and quantitatively.

7.2.2 Purposes

Table 9.1 shows how far each of the goals described in the report's introduction Chapter has been completed.

Purpose 1	Organize a set of guidelines.
This Chapter sets initial stage for the project by explaining the background study, problem elaboration, domain problem belongs to, possible solution, related work conducted previously and possibly activity timeline.	
Purpose 2	Review of the Literature

The literature search yielded a thorough understanding of different recommender styles & architectures, as well as their strengths & weaknesses, and the effects of those weaknesses &strengths on the precision and manageability of project framework. This Second Chapter of the report is purly devoted to a literature review.	
Purpose 3	The approach for software development must be chosen.
After evaluating many techniques, spiral methodology has been chosen because of versatility to deal with regular changing requirements. Refer to Section 3 of this report for more background on analytical techniques and managing time explanations.	
Purpose 4	Requirement gathering process
The GraphMk system's requirements were gathered from domain experts and end users, as well as using various requirement collection techniques such as interviews and questionnaires to gather information from the above-mentioned target people. Please see Chapter 4 of this report for more detailed information on specifications.	
Purpose 5	Build a list of program specifications (SRS)
This section focuses on gathering functionality requirement for system through various ways including survey, interviews and literature. Refer Chapter 4 for more in-depth details.	
Purpose 6	Core technology & resources are selected.
After defining the result for the heft method, best software, APIs, frameworks, hardware, and algorithms were chosen. The Sixth Chapter of this study provides a concise summary.	
Purpose 7	Prepare software design specification
The framework was designed using requirements identified through requirement analysis and techniques identified through literature review. This data has been applied to Chapter 5.	
Purpose 8	Prototype development
This section presented the development process, which was carried out due to factors described in the literature review, requirement specification, and design chapters. Development process is detailed, including related program code as well as approaches to solve found along the way. Chapter 6 now has a detailed description.	
Purpose 9	The model is being tested.

Well defined testing strategy was picked after a thorough analysis on qualitative and quantitative aspects. Section 7 discussed regarding testing process for Graphmk.	
Purpose 10	Evaluation of the work carried out
Project standard was assessed using aid from experts in the domain and personal evaluation. An analysis of the assessment results was then undertaken to assess the project's progress. Refer to Chapter 8 of this document for a more detailed description of the assessment.	
Purpose 11	Documentation
Throughout the project life cycle, each process was meticulously recorded. The documentation of each goal is accomplished throughout Chapters.	

Table 7.1 Purpose of GraphMk elaborated

7.3 Application of Knowledge from course modules

Most of modules learnt during these two years of the Masters degree programare related to the research topic chosen for the final project. The expertise and experience acquired during this academic tenure contributed significantly to the success of this project. Few modules that assisted in gaining the basic knowledge about the research project and carrying it out in an effective manner to ensure its success are given in Table 9.2 below;

Module	Final year project module
The final year project module assisted in planning and executing each step of the project life cycle, as well as keeping track of common project notes with the aid of the lecturer. In the second year, the project management module was used to each phase requires proper estimation of time and resources since project timeline is very limited.	
Module	Software Architecture, Operating system & Advanced algorithm
Software Architecture, operating systems and distributed computing are useful modules that helped in designing the system and resolved complex issues that were raised during the project process. Performance modelling and advanced algorithm modules helped in understanding basic graph structure, computation and different algorithmic styles.	
Module	Software quality assurance testing (SQAT)

With the basic information obtained from the SQAT module, defining test levels, procedures, and implementing test plans for the entire project with the aim of identifying defects in the prototype is greatly aided.	
Module	Software process and management
This module provided more in-depth knowledge of how to build software from scratch and how to set proper planning to adhere which will end in a successful development cycle.	
Module	Requirement Engineering
The final year module assisted in the learning of concept diagrams and methods. Based on the knowledge gained from requirements engineering module, requirement specification section in this writeup was able to be articulated in a proper manner.	
Module	Performance modelling and Analysis
Performance modelling and analysis provided expertise on mathematical depth knowledge, which improved analytical thinking, logical thinking along with guidance on considering performance.	

Table 7.2 Details on impactful modules that helped in making of Graphmk

7.4 Enhanced Existing Skills

- Established expertise in designing systems using data flow diagrams, package diagrams and object-oriented instances of class, sequences had been effectively used for the system's design.
- Prior experience working with a Big Data parallel framework like Hadoop, as well as knowledge of distributed file systems and the map reduce programming model, aided greatly in accelerating the project's launch.
- The project was completed successfully using existing programming skills in functional programming languages such as Scala and Python, as well as basic knowledge of graph theory.
- A thorough understanding of relational databases and how their querying engines operate, as well as graph analytical insights, aided in the development of a new view mapper that incorporates graph analysis with relational data.

7.5 Experiences from Project

- Masters' curriculum helped in many ways to improve the thought process in developing the system. Solid foundation on relational databases, basics of graph theory, computation helped to find ways to evolved the model introduced that act as a view mapper on top of relational data for iterative computation.
- For the project's successful completion, documentations, tutorials, examples, and videos were used to self-learn different models used in graph and relational engines, query languages and hands-on experience from different engines.
- The prototype needs to be tested quantitatively and qualitatively where it reflects proof of completeness. To carry out this process, solid knowledge of software quality assurance and testing skills were needed. This was made possible by the mentor's encouragement, the final year's Software Quality module, and, most importantly, self-learning.
- Writing and Chapter during the project life cycle, as well as incremental learning and hands-on experience, enhanced documentation and critical thinking skills.

It can be inferred that experience acquired through Master's program, existing capability and self-taught knowledge must be integrated and supported for the project to be completed successfully.

7.6 Details on Challenges encountered

7.6.1 Scope Definition

Graph computation is considered to be trending in Big Data analytical arena. The research area covers a broader scope and deeper learning knowledge. But the issue was thought to be complicated and too general for a bachelor's level research at the start of the project. This was discovered after an in-depth analysis was conducted for the project's literature review process, and it became clear that the scope had grown to a point that it could not be completed in the period allotted. The requirement elicitation method and the conclusions drawn from it were used to reduce the complexity of the project to a manageable level.

7.6.2 Availability of Academic Publications

Even though there are numerous graph analytical systems that exists within Big Data research arena, the concept of utilizing same resources for multiple requirements was not addressed properly. The idea of doing graph computation for a relational data within a relational query engine is not tapped fully for research opportunities. As a result, the issue domain discussed in the project has received less scholarly publications. This problem was solved by reading similar theories and principles as well as expert advice and guidance. Apart from that, self-study and experimentation were helpful in making good progress.

7.6.3 Support materials to learn key concepts and technologies

Awareness of graph analytics within Big Data and data analytics is lacking. It will take a considerable amount of time to set up data-parallel systems and test out sample computations. While there were numerous online support desks for data-parallel systems, they were difficult to comprehend. Amidst, building an integrated view mapper on top of relational query engine and relational data, seems a daunting task due to the lack of experience with relational data models in general. It was a difficult task to comprehend graph views within relational data and query engine, where the medium of querying need to be defined and understood by the same infrastructure. Contacting domain experts, systematic self-study, and experiments were used to solve this problem.

7.6.4 Constraints related to time

Since it is a research project, there is a chance of exceeding the project's allotted time due to a lack of domain expertise and regular changes in specifications. A spiral development approach, which makes the development process iterative and supports fluctuating requirements, could be adapted to face and overcome the time constraint risk. The production approach chosen greatly aided in managing the project's time constraints and resulting in its successful completion.

7.7 Details on Limitations from the research

7.7.1 Restrictions on a graph analytical system's main success factor

Present study focused solely on graph analytical device precision, memory use, and scalability as key success factors. However, there are a variety of other key success factors, and this study can be extended to look at the accommodation of certain domain-specific key success factors that have been overlooked.

7.8 Possible Future work for GraphMk

Below table shows detailed overview on possible improvements graphmk system can adopt, as well as new dimensions of computations that the framework could accommodate.

Enhancement	**ENH1**	**Priority Level**	**Medium**
Description	Help for dynamic graph analysis will be extended using the GraphMk graph model - The framework currently only supports static graph analytics, but the current world trend necessitates several more features, such as dynamic data streaming analytics. This will result in a new level of data analytics trends.		
Enhancement	**ENH2**	**Priority Level**	**Medium**
Description	Support for different graph analytical platforms need to be provided easily. Graphmk should be designed in such a way that it can help to add and remove external systems easily. It can be considered perfect when operated by a user interface without being too technical. This will allow every analyst to gain hands-on experience with any graph computing system, allowing him or her to easily validate output from different frameworks.		
Enhancement	**ENH3**	**Priority Level**	**High**
Description	Prototype should be flexible to adopt distributed computing as well as single node computation – The framework currently only supports single node computation, but modified to parallel execution with aid of a cluster. Many common graph computing frameworks would be able to execute computation in parallel.		
Enhancement	**ENH4**	**Priority Level**	**Low**

Description	Extend the prototype to provide a frontend querying UI, similar to Neo4j Querying Web-UI – Currently, the framework has a basic user interface with a question editor and a result screen. It would be much easier if the user interface could be improved to provide graphical representations of graphs and a real-time editor with query error detection. Customers will have a much greater user experience with the system as a result of this.		
Enhancement	**ENH5**	**Priority Level**	**Low**
Description	Implement a Web interface that would support different purpose of computation and make the framework easily accessible with better UI. It is important that UI reflects all the functionalities of the framework.		
Enhancement	**ENH6**	**Priority Level**	**Low**
Description	Introducing better query language that would have a better optimization of computation. It would also make it easy for users to have capabilities of executing functionalities and operators to do more computation.		
Enhancement	**ENH7**	**Priority Level**	**High**
Description	To improve the results output from graphmk system it is always useful to have an integrity checker on generated graph and computation executed on these graphs for duplicates and redundant data. System would be considered more reliable due to the fact that it can produce more accurate results with the introduction of integrity checker		

Table 7.3 Gramk future enhancement list

7.9 Contributions from GraphMk

The following are the four key contributions of this thesis:

- The Integrated-Graph (IG) model, a newly developed data model for graph analytics, is the first contribution of the research project. This RMV model is constructed with a relational core in the center and several graphical views surrounding it. Relational mapper views (RMV) are built to form relations in numbers to create graphs in the space between the relational core and graphical views. Using the newly implemented RMV model, any user can easily perform both relational and graph analytics tasks using a relational database.

- After that, the project contributes to research that results in Integrated-Graph Structured Query Language, a well-designed data log format querying which reflects more of SQL additional functions (RMV-SQL). Graph development, ranking, path finding, and clustering will all be supported by this newly updated query language. This does not preclude the user from using conventional SELECT, WHERE, and FROM statements to perform additional execution on graph analysis data, such as nested query operations, extracting a sub-graph or using aggregate. It also allows you to perform research on graphs and relationships.
- Finally, Graphmk provides an architecture that's well designed for relational mapper views (RMV) Engine, an improved query engine. The implementation of this well-designed engine is based on the open-source relational database PostgreSQL. Different graph analysis tools are used as plugins in this architecture to support graph computing tasks. Within this architecture, various operations such as add, delete, and change algorithms are simple to manipulate.
- Two tests were carried out to demonstrate the stability of the implementation. Initial experiment was on existing graph analytical tools that can be used with Graphmk frameowkr and RMV querying model and asses its supportability and execution efficiency. Experiment was conducted on Networx, SNAP and graph-tool where random graphs were generated and provided as input and check execution of various iterative graph computation to get output results. The tools were then judged on their ability to keep track of time and remember information. To determine the efficiency of the RMV engine, a comparison was made with other query engines such as Postgresql, a relational database engine, and Neo4j, a graph database engine.

7.10 Closing comments

The GraphMk framework was created with the aim of developing a single framework that could potentially extend the capability of relational databases with graph computation where it will be less hassle to migrate to different systems to have dedicated iterative computation. Research suggest that relational mapper of views can help to perform better execution of creating graph views from relational data. It also includes additional computational definitions included with SQL annotations and a query engine that supports. The suggested future enhancement features listed earlier in this Chapter, will aid the current solution for further enhancement.

References

[1] Aapo Kyrola, Guy Blelloch, Carlos Guestrin. (2012) 'GraphChi: Large-Scale Graph Computation on Just a PC'.

[2] Abdul Quamar, Amol Deshpande, and Jimmy Lin. NScale: neighborhood-centric analytics on large graphs. Proceedings of the VLDB Endowment, 7(13):1673–1676, 2014.

[3] Abhishek Jindal and Steve Madden. GRAPHiQL: A graph intuitive query language for relational databases. In Big Data (Big Data), 2014 IEEE International Conference on, pages 441–450. IEEE, 2014.

[4] Adam Welc, Raghavan Raman, Zhe Wu, Sungpack Hong, Hassan Chafi, and Jay Banerjee. Graph analysis: do we have to reinvent the wheel? In First International Workshop on Graph Data Management Experiences and Systems, page 7. ACM, 2013.

[5] Alekh Jindal, Samuel Madden, Malu Castellanos, and Meichun Hsu. Graph Analytics using the Vertica Relational Database. arXiv preprint arXiv:1412.5263, 2014.

[6] AllegroGraph RDFStore Web 3.0's Database. http : / / franz . com / agraph / allegrograph/.

[7] Amitabha Roy, Ivo Mihailovic, Willy Zwaenepoel. (2013) 'X-Stream: Edge-centric Graph Processing using Streaming Partitions'.

[8] Apache Jena - Home. https://jena.apache.org.

[9] A Comprehensive List of Big Data Statistics. [Online]. Wikibon blog < http://wikibon.org/blog/big-data-statistics/ > Accessed 20 Sept 2015.

[10] Bin Shao, Haixun Wang, and Yatao Li. Trinity: A distributed graph engine on a memory cloud. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pages 505–516. ACM, 2013.

[11] Chen.R, Weng.X, He.B, and Yang.M (2010) 'Large graph processing in the cloud', pp 1123–1126, Indianapolis, Indiana, USA,. ACM

[12] Dean.J and Ghemawat.S. (2004) 'Mapreduce: Simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation', Vol 6, pp 10–10.

[13] Eiko Yoneki, Amitabha Roy. (2013) 'Scale-up Graph Processing: A Storage-centric View'.

[14] Emil Eifrem. The NewWay to Access Super Fast Social Data. http://mashable.com/ 2012/09/26/graph-databases/.

[15] Facebook, 2014. http://facebook.com.

[16] Giraph - Welcome To Apache Giraph! http://giraph.apache.org.

[17] Gonzalez.J.E, Low.Y, Gu.H, Bickson.D, and Guestrin.C. (2012) 'Powergraph: distributed graph-parallel computation on natural graphs', pp 17–30.

[18] Google, 2014. http://google.com.

[19] Graph-tool: Efficient graph analysis. http://graph-tool.skewed.de.

[20] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 135–146. ACM, 2010.

[21] Guozhang Wang, Wenlei Xie, Alan J Demers, and Johannes Gehrke. Asynchronous Large-Scale Graph Processing Made Easy. In CIDR, 2013.

[22] Ian Robinson, JimWebber, and Emil Eifrem. Graph databases. " O'Reilly Media, Inc.", 2013.

[23] Jiefeng Cheng, Qin Liu, Zhenguo Li, Wei Fan, John C.S. Lui, Cheng He (2015) 'VENUS: Vertex-Centric Streamlined Graph Computation on a Single PC'.IEEE.

[24] Jiewen Huang, Kartik Venkatraman, and Daniel J Abadi. Query optimization of distributed pattern matching. In Data Engineering (ICDE), 2014 IEEE 30th International Conference on, pages 64–75. IEEE, 2014.

[25] Jing Fan, Adalbert Gerald, Soosai Raj, and Jignesh M Patel. The case against specialized graph analytics engines. 2015.

[26] Kamran Najeebullah, Kifayat Ullah Khan, Waqas Nawaz and Young-Koo Lee. (2014) 'BiShard Parallel Processor: A Disk-Based Processing Engine for Billion-Scale Graphs'.

[27] Kyrola.A, Blelloch.G, and Guestrin.C. (2012) 'GraphChi: large-scale graph computation on just a PC', pp. 31–46.

[28] Lassila Ora and Swick Ralph. Resource Description Framework (RDF) Model and Syntax Specification. http : / / www. w3 . org / TR / 1999 / REC-rdf-syntax-19990222/

[29] Leskovec.J, Lang.K, Dasgupta.A, and Mahoney.M. (2009) 'Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics' pp 29–123.

[30] Low.Y, Gonzalez.J, Kyrola.A, Bickson.D, and Guestrin.C. (2012) 'Graphlab: A distributed framework for machine learning in the cloud'.

[31] Mahesh Lal (2015) 'Neo4j Graph Data Modeling'. Packt Publishing

[32] Malewicz.G, Austern.M. H, Bik.A. J, Dehnert.J, Horn.I, Leiser.N, and Czajkowski.G. (2010) 'Pregel: a system for large-scale graph processing'.

[33] Min-Soo Kim, Sangyeon Lee, Wook-Shin Han, Himchan Park, and Jeong-Hoon Lee (2015) 'DSP-CC-: I/O Efficient Parallel Computation of Connected Components in Billion-Scale Networks'.IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.

[34] Neo4j, World's Leading Graph Database. [Online] < http://neo4j.com > Accessed 21 Dec 2015.

[35] Neo4j and Apache Spark - Neo4j Graph Database. http://neo4j.com/developer/apache-spark/#mazerunner.

[36] OrientDB - OrientDB Multi-Model NoSQL DatabaseOrientDB Multi-Model NoSQL Database. http://orientdb.com/orientdb/.

[37] Overview - NetworkX. http://networkx.github.io.

[38] Page.L, Brin.S, Motwani.R, and Winograd.T. (1999) 'The pagerank citation ranking: Bringing order to the web', Stanford InfoLab,

[39] Paul Erdos and Alfred Renyi. On the strength of connectedness of a random graph. Acta Mathematica Hungarica, 12(1-2):261–267, 1961.

[40] Pingpeng Yuan, Wenya Zhang, Changfeng Xie, Hai Jin, Ling Liu, Kisung Lee (2015) 'Fast Iterative Graph Computation: A Path Centric Approach'. International Conference for High Performance Computing, Networking, Storage and Analysis.IEEE.

[41] Power.R and Li. Piccolo.J (2010) 'Building fast, distributed programs with partitioned tables', pp 1–14.

[42] Santo Fortunato. Community detection in graphs. Physics Reports, 486(3):75–174, 2010.

[43] Raja Appuswamy, Christos Gkantsidis, Dushyanth Narayanan,Orion Hodson, and Antony Rowstron (2013) 'Scale-up vs Scale-out for Hadoop: Time to rethink?'. Cambridge University

[44] Raymond Cheng, Ji Hong, Aapo Kyrola, Youshan Miao, Xuetian Weng, Ming Wu, Fan Yang, Lidong Zhou, Feng Zhao, and Enhong Chen. Kineograph: taking the pulse of a fast-changing and connected world. In Proceedings of the 7th ACM european conference on Computer Systems, pages 85–98. ACM, 2012.

[45] Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of databases, volume 8. Addison-Wesley Reading, 1995

[46] Sherif Sakr, Sameh Elnikety, and Yuxiong He. G-SPARQL: a hybrid engine for querying large attributed graphs. In Proceedings of the 21st ACM international conference on Information and knowledge management, pages 335–344. ACM, 2012.

[47] Stanford Graph Analysis Project. http://snap.stanford.edu.

[48] Stardog: Enterprise Graph Database. http://stardog.com.

[49] Talbot.J, Yoo.R. M., and Kozyrakis.C. (2011) ‘Phoenix++: Modular MapReduce for Shared-Memory Systems’.

[50] Titan: Distributed Graph Database. http://thinkaurelius.github.io/titan/.

[51] Ulrik Brandes and Thomas Erlebach. Graph analysis: methodological foundations, volume 3418. Springer Science & Business Media, 2005.

[52] Wook-Shin Han, Sangyeon Lee, Kyungyeol Park, Jeong-Hoon Lee,Min-Soo Kim1, Jinha Kim, Hwanjo Yu. (2013) 'TurboGraph: A Fast Parallel Graph Engine Handling Billion-scale Graphs in a Single PC'.

[53] Xenarios.I, Salwinski.L, Duan.X. J., Higney.P, Kim.S.M., and Eisenberg.D. (2002) ‘The database of interacting proteins: a research tool for studying cellular networks of protein interactions. Nucleic acids research’, pp 303–305.

[54] Yan.X, Yu.P. S, and Han.J. (2004) ‘Graph indexing: A frequent structure-based approach’, pp 335–346.

[55] Yifang Jiang, Kai Chen, Yi Zhou, Diao Zhang, Qu Zhou, Jianhua He (2014) 'An Improved Memory Management Scheme for Large Scale Graph Computing Engine GraphChi'. 2014 IEEE International Conference on Big Data

[56] Yuanyuan Tian, Andrey Balmin, Severin Andreas Corsten, Shirish Tatikonda, and John McPherson. From think like a vertex to think like a graph. Proceedings of the VLDB Endowment, 7(3):193–204, 2013.

[57] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. arXiv preprint arXiv:1408.2041, 2014.

[58] Yuhanna Noel, Owens Leslie, and Elizabeth Cullen. Market Overview: Graph Databases. https://www.forrester.com/Market+Overview+Graph+Databases/ fulltext/-/E-res121473.

[59] Zaharia.M, Chowdhury.M, Franklin.M. J., Shenker.S, and Stoica.I. (2010) ‘Spark: Cluster computing with working sets. In HotCloud’.

[60] Zhao.P and Han.J. (2010) ‘On graph query optimization in large networks.PVLDB’, pp 340–351.

[61] Zhiyuan Lin, Minsuk Kahng, Kaeser Md. Sabrin, Duen Horng (Polo) Chau (2015) 'MMap: Fast Billion-Scale Graph Computation on a PC via Memory Mapping'. IEEE.

8 Appendixes

8.1 Design Appendix

8.1.1 Data flow representation

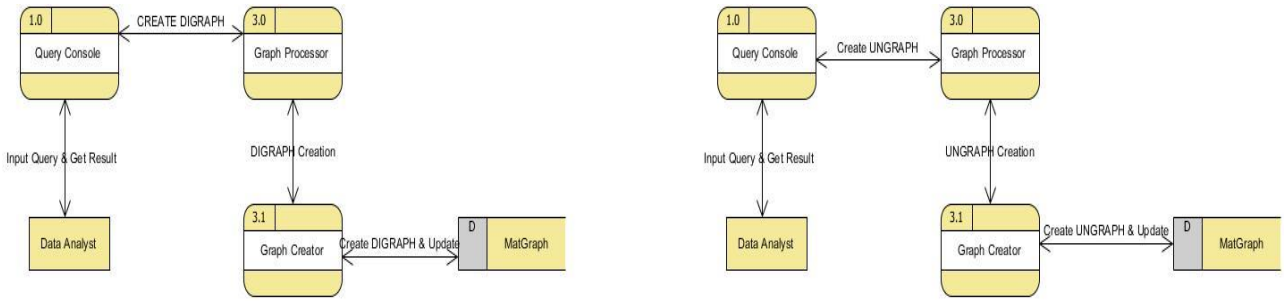


Figure 8.1 Graph creation – Data flow

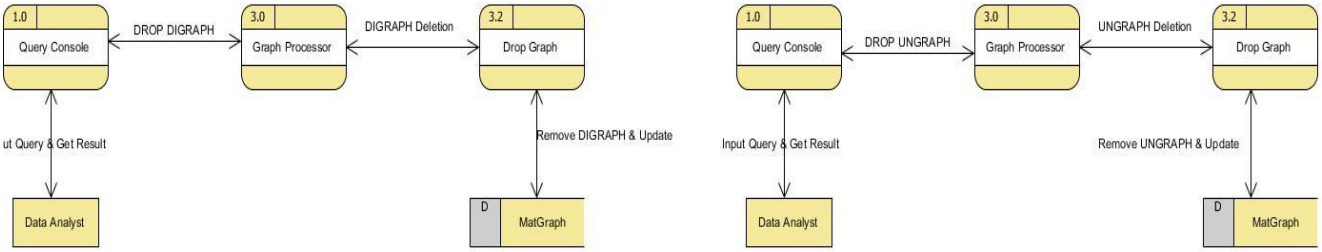


Figure 8.2 Graph drop – Data flow

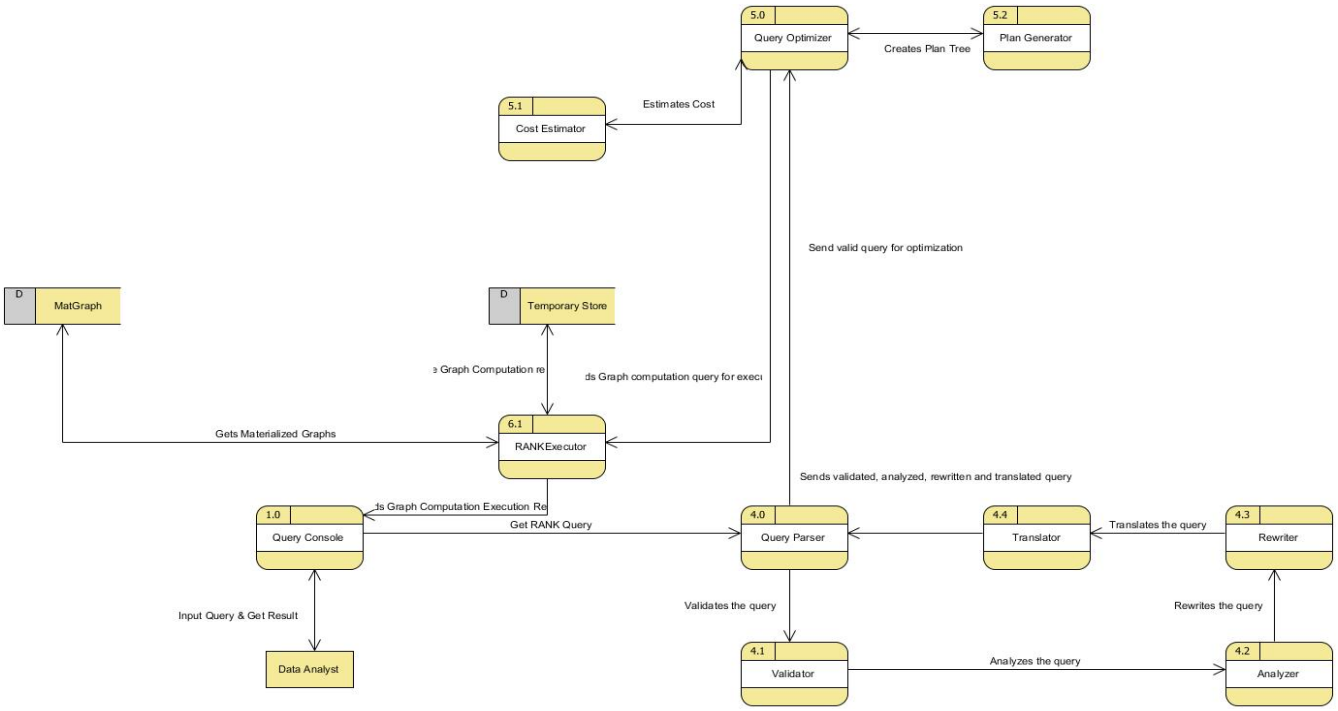


Figure 8.3 Rank implementation – Data flow

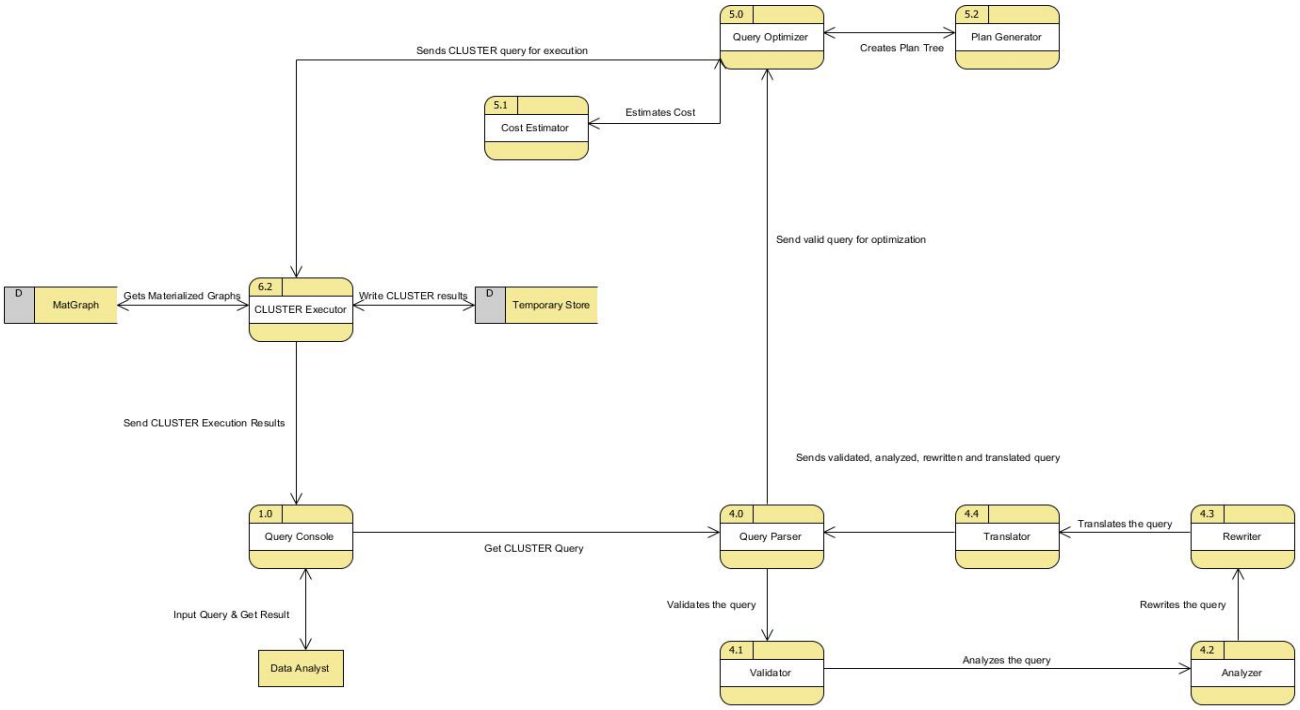


Figure 8.4 Cluster implementation – Data flow

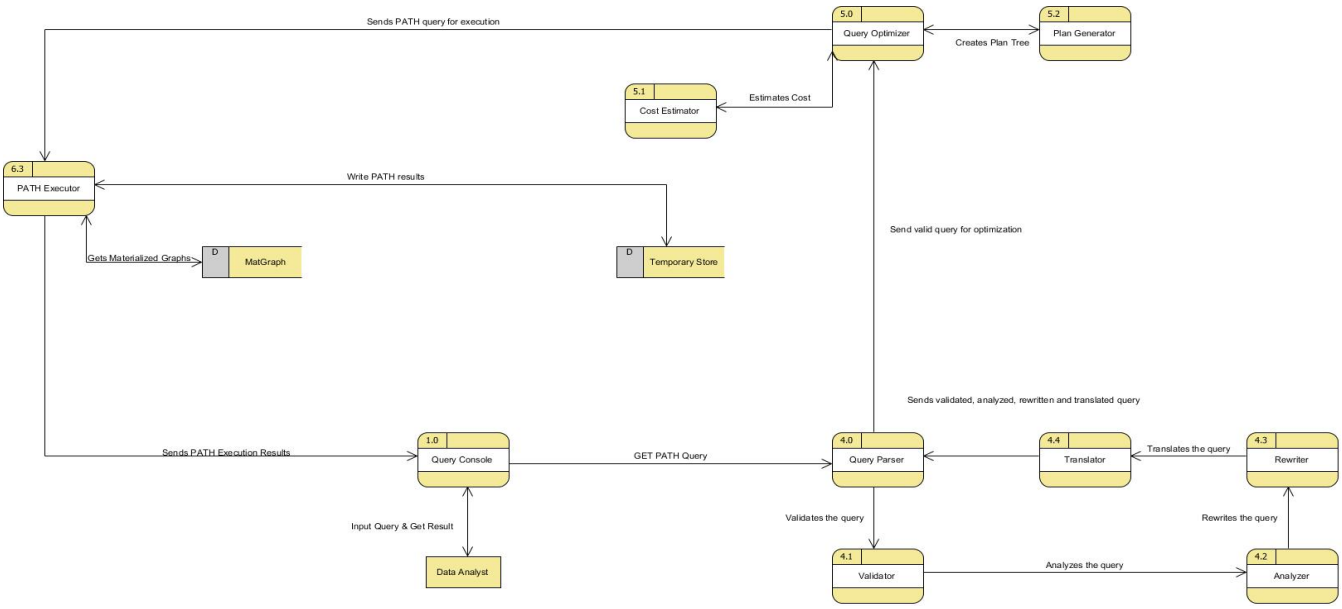


Figure 8.5 Path implementation – Data flow

8.1.2 Package Diagram

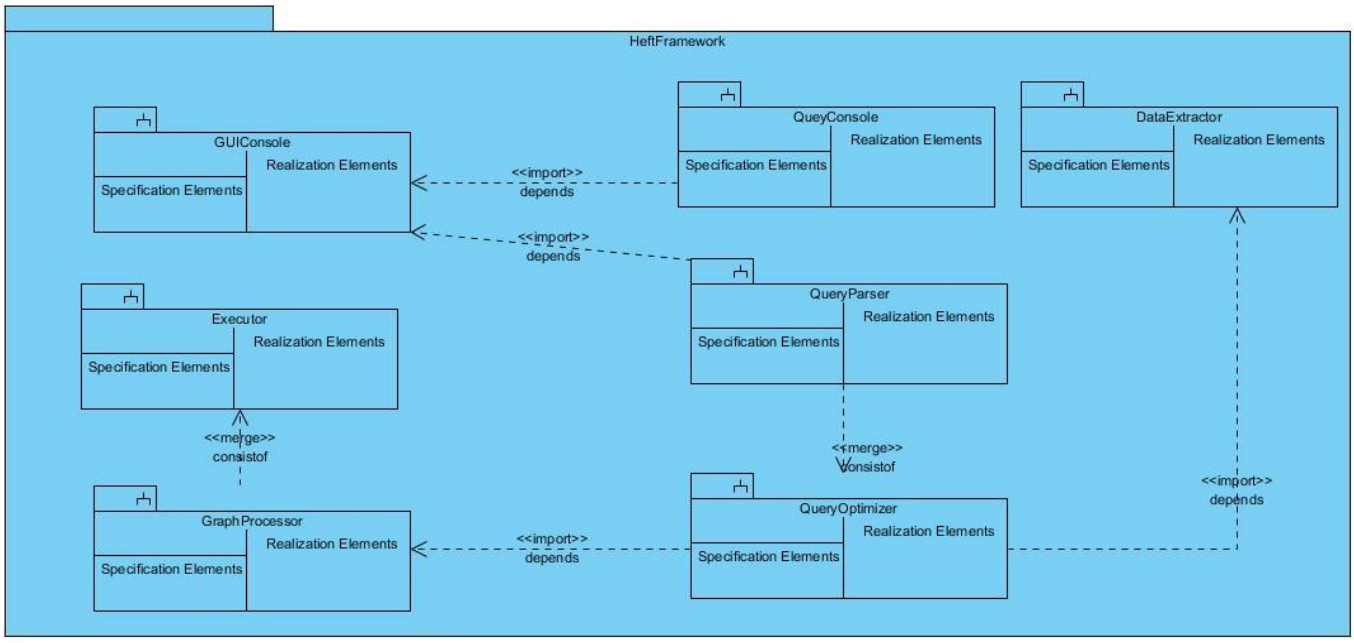


Figure 8.6 Graphmk package diagram

8.1.3 ER Diagram

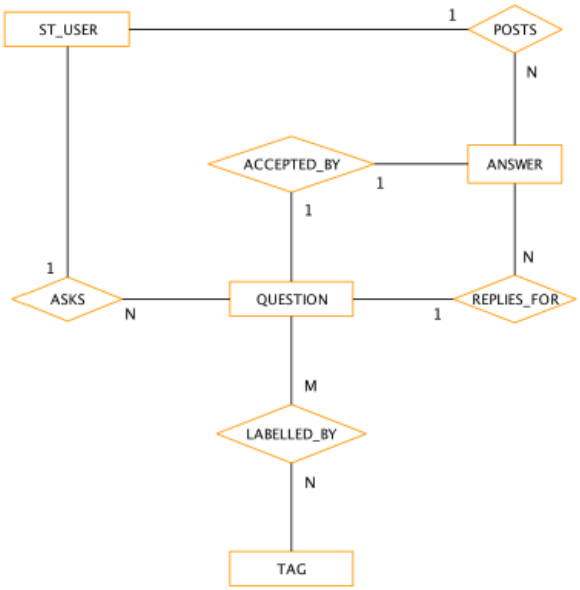


Figure 8.7 Example Stack overflow – ER diagram

8.2 Implementation Appendix

8.2.1 Benchmarking GraphMk

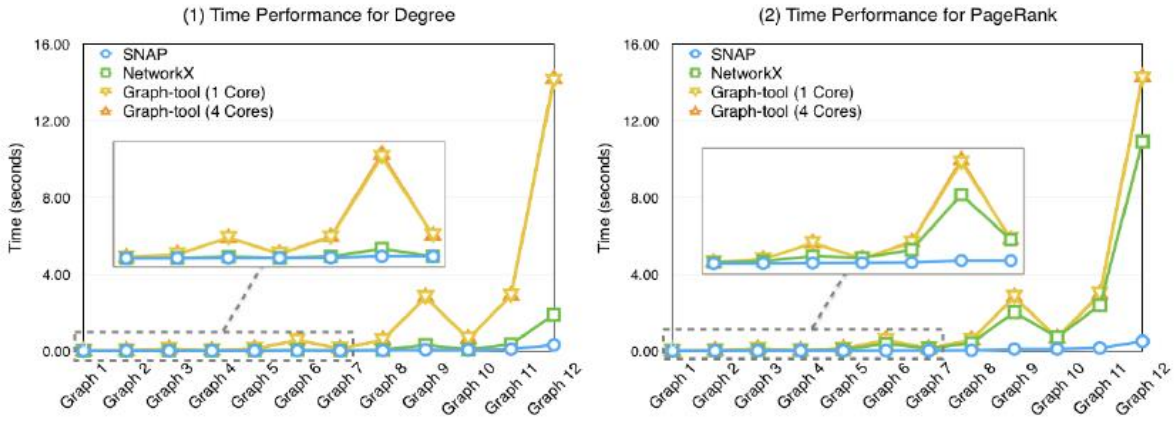


Figure 8.8 Rank execution – time utilization

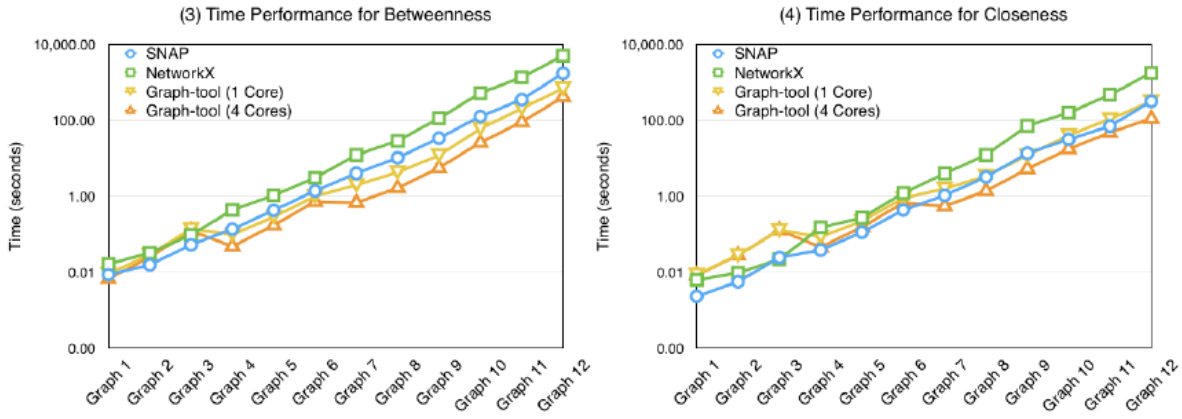


Figure 8.9 Closeness – time utilization

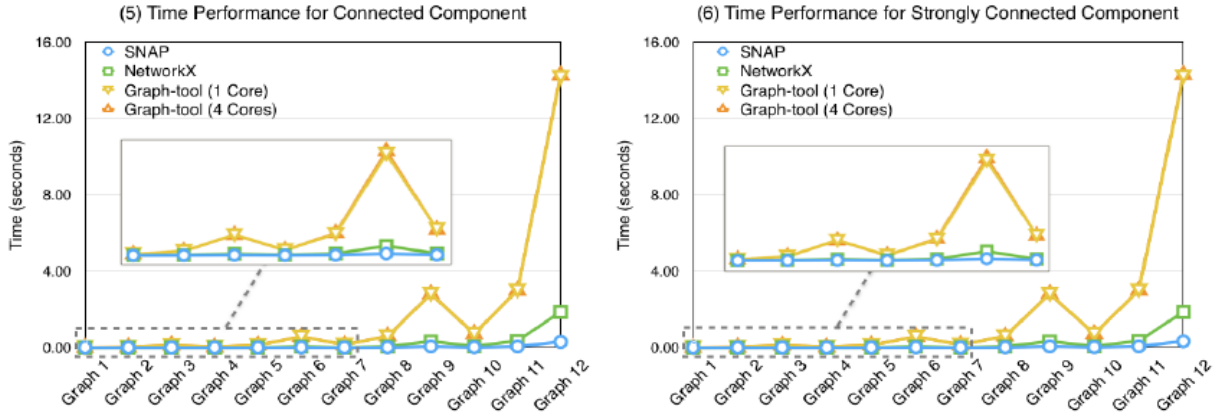


Figure 8.10 Connected components – time utilization

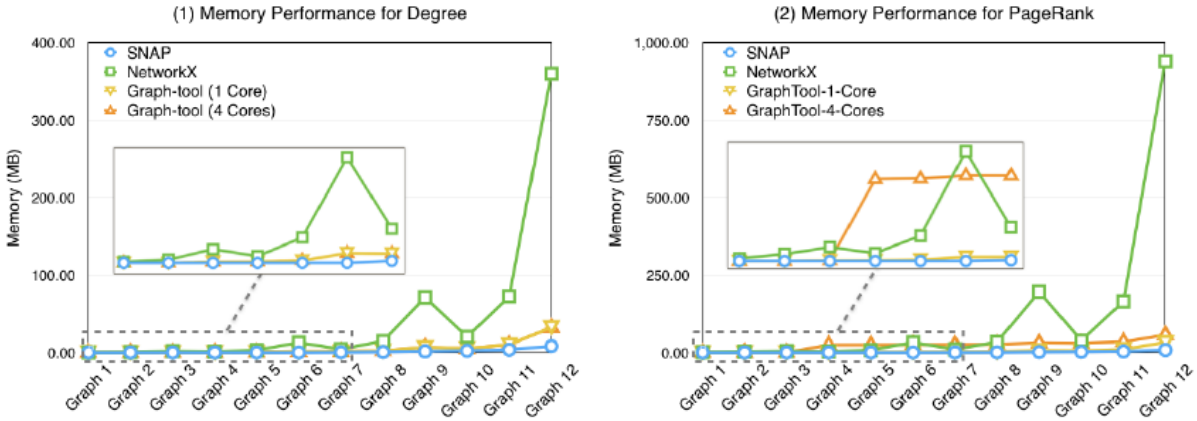


Figure 8.11 Rank execution – memory utilization

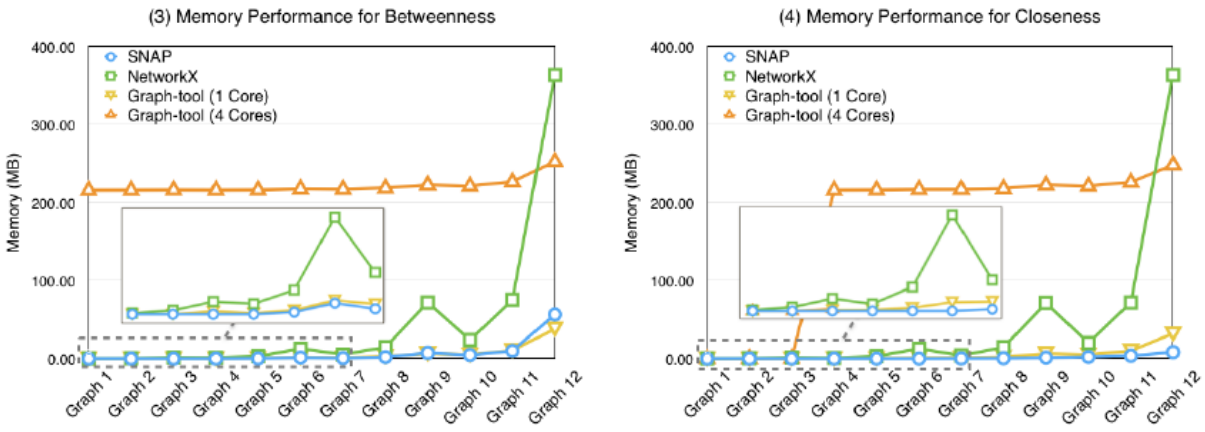


Figure 8.12 Closeness execution – memory utilization

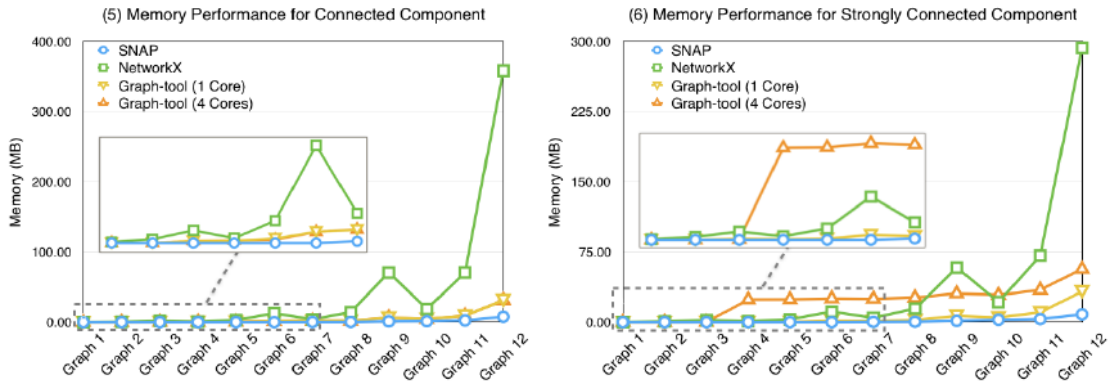


Figure 8.13 Cluster computation – memory utilization