Graph Data Warehouse: Steps to Integrating Graph Databases into the Traditional Conceptual Structure of a Data Warehouse

Work in Progress

Yunkai Liu, Ph.D. and Theresa M. Vitolo, Ph.D.
Dept. of Computer and Information Science
Gannon University
Erie, PA, United States
liu006@gannon.edu, vitolo001@gannon.edu

***Abstract*— As an important kind of NOSQL databases, graph database management systems (GDBMSs) have improved greatly in recent years. However, the development of related linked queries is still limited. This paper introduces an approach for matching frequently-used management tasks of GDBMS with concepts in Structured Query Language (SQL). An application programming interface (API) based on the Neo4j GDBMS was developed to enable functionalities needed with graph databases. Besides the API, a user-friendly graphical user interface (GUI) was developed enabling the management and demonstration of graph datasets. Further, the concept "graph cube" is proposed as a design for integrating graphs with tables. The prototype combining these components constitutes the fundamental elements of a graph data warehouse. The proposed research initiates an effort to develop the potential of graph information systems (GRIS) and to define associated technical aspects.**

Database Management Systems; NOSQL Database; Graph Database; Data Warehouses; Linked Queries; Cube; Join

I. FROM SQL DATABASE TO GRAPH DATABASE

With the continual growth of web-scale data, unique database management systems have gained prominence and are sought as components in the next generation of information systems. Classified as NOSQL (Not Only SQL), the databases are designed to represent special data such as might arise with web-interactions or to offer functionalities for manipulating such data [1, 2].

Graph databases [3], a kind of NOSQL databases, employ a graph as their data model. Graph database management systems (GDBMSs), such as Neo4j, usually provide a more efficient graph transversal framework with massive scalability (up to billions of nodes, edges, and related properties) [4].

However, GDBMSs do not have capabilities for easily constructed, linked queries -- a very essential part of any database management system [5]. Most current graph databases only provide functions for developers. Neo4j does provide Cypher, but its query functionalities are limited. For example, some typical data manipulation operations, such as insert and update, are not provided in Cypher. The relational and SQL data world has an engrained expectation for a standardized command structure allowing easy manipulation of values with the data representation.

A. The Innovative Approach

NOSQL databases could benefit from the inclusion of standard SQL operations. The conceptualization of such a combination of SQL terms with graph databases [6, 7] is expressed in Table I. In the table, the behavior to be expressed by the command is indicated. At this time, the implementation of the functionality is pending.

TABLE I. QUERY DESGIN OF PROPOSED API

Query Categories	Query Functions	Descriptions
Graph Data Manipulation Language (GDML)	SELECT	*To retrieve node(s), an edge, or a sub-graph based on specific node(s);*
	INSERT	*To add node(s) or edge(s) into a graph*
	UPDATE	*To remove node(s) or edge(s) from a graph*
	DELETE	*To change the values of attributes of node(s) or edge(s)*
Graph Data Definition Language (GDDL)	CREATE	*To create a graph*
	DROP	*To delete a graph*
	UPDATE	*To add or remove one or several attributes for all nodes*
Graph Mining Language (GML)	SHORTEST PATH	*To find one or all shortest paths between two nodes*
	NEIGHBORS	*To find all connected nodes*
	COMMON NEIGHBORS	*To find the common neighbors (nodes) of two nodes*
	MOTIFS	*To find the most-frequently-occurring-subgraph(s)*
Graph Structure Statistics Language (GSSL)	COUNTALL	*To retrieve the number of nodes or edges*
	D_SEQ	*To retrieve the degree sequences (sorted)*
	DIAMETER	*To retrieve the length of the longest shortest path*
Other Queries	CONNECT	*To connect a specific graph database*
	SAVE	*To save current graph data into a database*
	PAINT	*To assign locations to nodes, based on specific layout parameters*

The work began with the development of a Java-based application programming interface (API) accessing a Neo4j graph database and initiates the functionalities of Table I. In

978-0-7695-5006-0/13 $26.00 © 2013 IEEE
DOI 10.1109/BigData.Congress.2013.72

addition, a graphical user interface (GUI) was created to provide a more direct and friendly visualization of the graph data [8]; (see Fig. 1). Visualization functions, such as drag-and-drop and highlight, are handled by the interface.

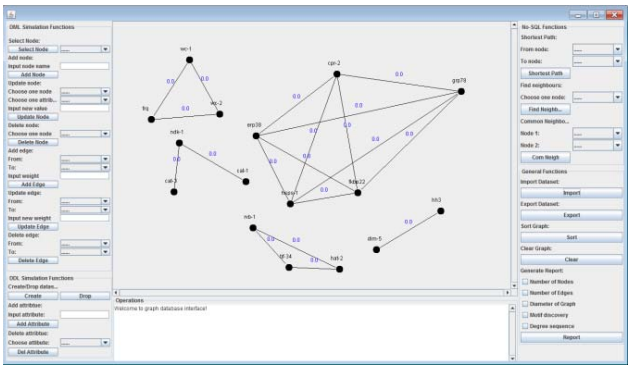

Figure 1. Screenshot of the GUI interface designed to assist data adminstrators or a stand-alone data user.

During the development of the interface, the concept of a "graph view" (GVIEW) was conceived and implemented. The GVIEW is a logical copy of graph data, residing in local libraries [7]. Its purposes are:

1. *To provide a local copy of data*. The concept is similar to "view" in RDBMS. With GVIEW, manipulations on the database can be more secure and faster. With GVIEW concurrent, multiple users of the graph can have individualized perspectives.
2. *To provide a representation of a graph or graph segment*. GVIEW is an expression of the graph data on the server. Extra information, such as the positions of each node from the last modification, is saved in GVIEW. High-end users whose work relies heavily on direct observation will find the option attractive, giving a convenient way to "save" or "remember" last visualized graphs.

II. Building the Graph Data Warehouse

Currently data warehouses are the organizational asset for knowledge discovery, business intelligence, and value-added awareness. Graph databases have the potential for knowledge building: connecting observations into inference structures, connected networks, whose properties can be represented, tested, and accepted as valid.

The two data models – graph and relational – have fundamentally different premises describing participation in a relationship. The GDBMSs employ physical adjacency while RDBMSs employ synthetic adjacency, inferring a relationship due to the key-field for instance. Subsequent steps of the project explore incorporating join-operations between relational data and graph data.

Extending graph data into the data warehouse paradigm, a new concept, termed a "graph cube" (GCUBE) would be developed. The major goal of a GCUBE in a data warehouse is to incorporate graphs as part of fact tables; see Fig. 2. The new graph cubes will require a "join" function. The value of defining GCUBEs is:

1. *To include aspects of graph data (named as "fact graph") into the mature data warehouse technologies*;
2. and, *to enhance the capabilities of data warehouse with functional capabilities of graph data.*

As professionals of data systems seek to incorporate graph databases into mainstream data processing, they will be seeking two core objectives: (1) to connect relational data with graph data and (2) to connect / aggregate multiple distinct graph data together. Fundamental to both of these actions will be a new join-operator.

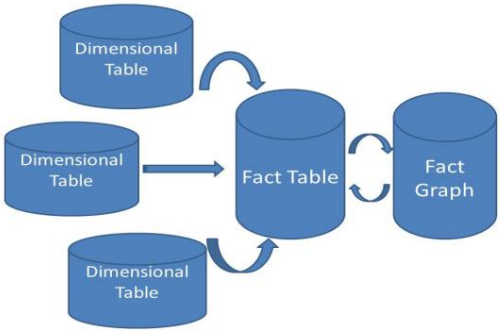

Figure 2. Star schema of graph cubes of a graph data warehouse.

III. Conclusions

The long-term goal of the project Graph Information Systems and Technologies (GRIST) is the study of graph information systems (GRIS) from theoretical levels into application areas. The growing prevalence of graph databases in the industrial and marketing arenas will require their connection with data warehouses. Achieving the goals of the work-in-progress will allow the wealth of social media data to be incorporated into decision-making.

References

[1] M. Stonebraker, "SQL databases v. NoSQL databases," *Comm. ACM*, vol. 53, pp. 10 – 11, Apr. 2010.

[2] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record*, vol. 39, pp. 12 – 27, Dec. 2010.

[3] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, pp. 1- 39, Feb. 2008.

[4] Neo Technology, Inc. (2013). *The Neo4j Manual v1.8* [Online] Available: http://docs.neo4j.org/chunked/stable/index.html.

[5] F. Holzschuher and R. Peinl, "Performamce of graph query languages: Comparson of Cypher, Gremlin and native access in Neo4j," *EDBT'13 Proc. Joint EDBT/ICDT 2013 Workshops*, pp. 195 – 204, 2013.

[6] Y. Liu, and L.Zhang, "Transferring biology data into graph model: A comprehensive graphic user interface to convert, manage and represent biological networks in graph database," *Proc. 2012 Great Lakes Bioinformatics Conf. (GLBIO-12)*, Apr. 2012.

[7] L. Zhang, "SQL and NO-SQL queries in graph database," Master's field study paper, Depart. Comput. and Inform. Sci., Gannon Univ., Dec. 2012.

[8] Q. Guan, "A comprehensive interface for graph database," Master's field study paper, Depart. Comput. and Inform. Sci., Gannon Univ., Dec.2012.