Industrial Science
Vol.1,Issue.9/Feb. 2015 **ISSN** : 2347-5420

Research Papers

BASIC STUDY ON THE DESIGN OF GRAPH DATABASE BENCHMARKS

Rahul Arun Deshmukh

MSc in Research in Computer and System Engineering Technische
Universitat Ilmenau, Germany

Abstract

TA Graph Database Benchmark is a collection of programs which is used to generate measurements to evaluate capability of systems like efficiency, etc. It can be considered as a useful tool to test the performance of a system. It also helps in deciding which system would be useful for a particular use case based on the availability and the demands. Graph Database Management systems (GDBs) are gaining popularity. They are used to analyze huge graph datasets that are naturally appearing in many application areas to model interrelated data. The objective of this paper is to raise a new topic of discussion in the benchmarking community and allow practitioners having a set of basic guidelines for GDB benchmarking. I strongly believe that GDBs will become an important player in the market field of data analysis, and with that, their performance and capabilities will also become important. For this reason, I discuss those aspects that are important from our perspective, i.e. the characteristics of the graphs to be included in the benchmark, the characteristics of the queries that are important in graph analysis applications and the evaluation workbench.

KEYWORDS:

Graph Database Benchmark, Graph Database Management, Graph Database Model, graph operation, graph characteristics.

INTRODUCTION

The analysis and storage of data in the form of a graph has increased in therecent years. Analyzing the characteristics of social networks, the use of theInternet, or the interactions among proteins has put graph processing in the eyeof the storm. The amount of data managed in most of those cases is huge, andthe complexity of the algorithms needed for the analysis as well, leading to aclear need in the market: the Graph Database Management System (GDB).A GDB is a step forward in the management and analysis of data. As statedby Angles and Gutierrez [1]: “Graph database models can be defined as thosein which data structures for the schema and instances are modeled as graphs orgeneralizations of them, and data manipulation is expressed by graph-orientedoperations and type constructors”. Graph databases emphasize the queries thatcompute results related to the structure of the links in the graphs rather thanon the entities themselves: for example detecting link patterns, path analysis,authority relations, etc. However, managing large graphs is a complex issue, andobtaining the best suited analysis algorithms is difficult.There are a certainly growing number of initiatives to implement and commercializeGDBs, like Neo4j [2], HyperGraphDB [3], Infogrid [4] or DEX [5] andmany RDF solutions such as Jena [6] or AllegroGraph [7]. There are other initiativesto create graph querying languages that allow for a simplified view of querying to the user like SPARQL [8] and Gremlin [9]. This shows that the

communityis very active proposing new technologies, and sets an adequate scenarioto reflect on which is the adequate benchmark for a GDB.The main objective of this paper is to open the discussion on GDB benchmarking.

Thus, I describe and discuss important aspects to be considered forbenchmarking. I started describing and analyzing the type of applicationswhere it is necessary the use of GDBs. In particular, I review the applicationof GDBs in Social Network Analysis (SNA), proteomics, recommendation systems,travel planning and routing, which gives a catalog of representative areaswhere huge graph datasets are appearing.I believe that the set of applications mentioned are representative of themarketplace for GDBs. Thus, based on those, I discuss the characteristics of the graphs that appear in such applications and how they could influence benchmarking.

I also survey different types of operations, why they are importantand how they can be categorized in order to produce a wide coverage of issueswithin a benchmark. Finally, I discuss on the evaluation setup of thebenchmark, where issues like the experimental process to follow and the typeof measures to be taken are considered. Despite the diversity of applications,I found that the different areas have common features, and I believe that thedesign of a benchmark based on SNA would become a representative candidateof general GDB applications.

The paper is organized as follows. I start by setting up the state of the artin Section 2. Then, in Section 3, I analyze a broad spectrum of applicationsdemanding for massive graph management. From these scenarios, I extractgeneric characteristics of graphs and queries, that will be important in order todesign a graph database benchmark. Also, I propose a query categorizationand I remark the relevant aspects that I should take into account for theexperimental settings of a benchmark. Finally, I draw some conclusions.

2.GRAPH ORIENTED BENCHMARKS

Popular database benchmarks, such as TPC-C or TPC-H [10], focus on evaluatingrelational database queries that are typical of a business application. Thesebenchmarks emphasize queries with joins, projections, selections, aggregationsand sorting operations. However, since GDBs aim at different types of queries,these widespread benchmarks are not adequate for evaluating their performance.

Object oriented databases (OODB) share some similarities with GDBs. Thedata of a OODB also conforms a graph structure, where the entities that arerepresented as objects draw relationships among them. The OO1 benchmark [11],one of the earliest proposals, is a very simple benchmark that emphasizes threebasic operations for OODB: (a) lookup, which finds the set of objects for a given object identifier; (b) traversal, which performs a 7-hop operation starting from arandom node; and (c) insertion, which adds a set of objects and relations to thedatabase. OO1 defines a dataset that only contains one type of objects with afixed number of outgoing edges per object. Since the links mostly go to objectswith a similar document identifier, the graphs are very regular. Another popular benchmark for OODB is the OO7 proposed by Carey etal [12]. In OO7, the database contains three types of objects, which are organizedas a tree of depth seven. The connectivity of the database is also very regularbecause objects have a fixed number of relations. The benchmark is made up bya rich set of queries that can be clustered into two groups: (a) traversal queries,which scan one type of objects and then access the nodes connected to themin the tree, and (b) general queries, which mainly perform selections of objectsaccording to certain characteristics.

I observe that although OODB benchmarks create graphs, the graphs havea very different structure from typical graphs in graph analysis applications. AsI review in detail in Section 3.2, graphs in GDBs are very irregular: the degree ofthe nodes exhibit a large variance, nodes are clustered in communities and graphshave small diameters. Furthermore, the applications that interact with GDBsare mainly interested in analyzing the graph structure, i.e. the relationships,instead of the attributes in the objects. For example, operations such as findingthe shortest path connecting two objects or finding patterns (e.g. a clique) arecommon GDB operations that are not considered in OODB.XML databases also follow a model which relates entities. An XML databaseis a collection of typed data and attributes organized as a tree. One of the mostwell known benchmarks for XML databases is XMARK [13], which models an

auction site. The queries in the benchmark cover many aspects: selections, sortedaccess, tree

path location, aggregation, etc. Nevertheless, XML only models trees,which are a limited subclass of graphs.

In the recent years, the knowledge management community has made effortsto design a standard for representing the relations between metadata elements,which has derived in the introduction of the Resource Description Framework(RDF). Along with RDF, the community has designed a query language calledSPARQL, which is an SQL extension to describe relationship patterns amongentities. In order to test the performance of these knowledge bases, Guo et al.proposed a benchmark of 14 SPARQL queries in [14], which is known as LUBM.

To our knowledge, the only benchmark proposed for the evaluation of graphlibraries is the HPC Scalable Graph Analysis Benchmark v1.0 [15]. The benchmarkis compound by four separated operations on a graph that follows a powerlaw distribution: (a) insert the graph database as a bulk load; (b) retrieve theset of edges with maximum weight; (c) perform a k-hops operation; and (d)calculate the betweenless centrality of a graph, whose performance is measuredas the number of edges traversed per second. However, this benchmark doesnot evaluate some features expected from a GDB such as object labeling or attributemanagement. For an implementation and a discussion of the benchmark over different graph databases see [16].

3.BENCHMARKS CONSIDERATIONS

In this section, I discuss several aspects that may affect the design of a GDBbenchmark. First, I examine a set of motivating scenarios where graph databases are useful. Second, I describe the most commonly used graph types. From this,I explore common queries run in these scenarios and propose a categorizationdepending on their interaction with the graph data. Finally, I review someexperimental considerations for the design of GDB benchmarks.

1.1Analysis of Motivating Applications

Over the past years, there has been an increasing interest in multiple disciplineson datasets that can be represented and analyzed as a network or graph. Inthese networks, the nodes represent the entities and the edges the interactionor relationships between them. For example, the use of Social Network Analysis(SNA) has grown to currently become one of the most successful tools to investigate the structure of organizations and social groups, focusing on uncoveringthe structure of the interactions between people [17]. SNA techniques have beeneffectively used in several areas of interest like social interaction and network evolutionanalysis, counter-terrorism and covert networks, or even viral marketing.The Web 2.0 [18] and the increasing use of Internet applications that facilitateinteractive collaboration and information sharing has caused the appearance ofmany social networks of different kinds, like Facebook and LinkedIn for socialinteraction, or Flickr for multimedia sharing. Other web portals that containhumaninteractions can also be considered social networks, like in bibliographiccatalogs such as Scopus, ACM or IEEE, where the scientific community is sharinginformation and establishing de facto relationships. In all these cases, thereis an increasing interest in the analysis of the underlying networks, to obtain abetter knowledge of the patterns and the topological properties. This may beused to improve service to users or even to provide more profit to the informationproviders in the form of direct advertising or personalized services.

The rapid growth of the World Wide Web, has caused new graph structureddata to be archived and analyzed, such as hypertext and semi-structureddata [19]. Also, with RDF, users are allowed to explicitly describe semantic resourcesin the form of graphs [20]. In this context, and others such as finding web service's connection patterns, algorithms and applications for graph andsubgraph matching in data graphs are becoming increasingly important [21].

Pattern matching algorithms are also used to find relationships in social networks[22], find research collaboration partners, or mine the connections amongresearch paper publications in archived bibliography datasets.The use of graph theory combined with computing analysis has attracted theinterest of the graph mining community [23]. Some of the classical analysis in thisarea are the determination of an actor's centrality, to identify key players, leadersor relevant people in a community; the grouping of individuals in communitiesor affinity groups, to provide specific services or to improve

their connectivity;the identification of weaknesses or singular points in the network, for securityor counter-terrorism; or the study of social roles and collaborations, to get theposition of an actor in the society or to connect people to others to improve theirrelationships network in professional environments. Another area of interest is in proteomics or genetic interactions, where thelarge-scale study of proteins is considered the next step in the study of biologicalsystems. The fact that most proteins act in collaboration with other proteins isthe basis for proteomics to reveal which of those are potentially associated witha certain disease. In protein-protein interaction networks, nodes represent proteins,and edges between nodes represent physical interactions between proteins.

Several protein interaction databases are shared between hundreds of researchlaboratories and companies around the world, such as BioGRID [24] or the ProteinData Bank (PDB) [25]. With this data, some of the most usual analysisin genetic interactions are the study of the complexity of the network topology,the node analysis of centrality and role, the identification of articulation pointsor bridges between groups of proteins, or the clustering of proteins based ontheir relationship or neighborhoods. Many scientific software tools are availablefor the analysis and the visualization of data, like Navigator [26] for networkanalysis, visualization and graphing.

Network analysis has also become essential for recommendation systems. Inthis case, the goal is to present information that could be interesting to the usersbased on previous knowledge extracted fromtheir social environment. Recommendationsystems are prior to computers, but the Internet has exploded againthe use of recommendation for different purposes, such as on-line sales and catalogslike Amazon, or digital music services in iTune. Even PageRank [27] inGoogle can be considered a recommendation engine, and the analysis of hubsand authorities to rate Web pages in HITS [28] is an exploration of the networkof relationships of a web page with its hyperlinks. Although relational databaseshave been the storage system of choice in many commercial recommendation enginesfor collaborative filtering, like Strands [29] for social networks and e-Commerce,lately, new approaches have appeared using graph representation andexploration like conceptual graphs [30] or more recently DirectedEdge [31]. Anotherarea of network analysis where graphs may be large is travel planningand routing, where the system tries to find the most efficient path between twopoints according to some constraints or recommendations given by the user, likein Amadeus [32], or real time analysis of traffic networks in large cities. In thesecases, the data is naturally represented as graphs where vertices stand for thelocation and the edges are the routes with lengths and costs. Then, queries aremainly navigational operations between neighbors or network topology analysis.

1.2Graph Description

GDBs store entities as nodes, which have relations among them that are set asedges. However, depending on the particular application, the graph model maydiffer, showing a different degree of complexity. In addition to the plain storageof nodes and edges, I detail the main features required by some applications:

❖Attributes:

In addition to nodes and edges, graph databases store informationassociated to these nodes and edges. The associated information is typicallystring or numerical values, which indicate the features of the entity or relation.For the particular case of edges, some graphs include numerical attributes that quantify the relation, which is usually interpreted as the length, weight, cost orintensity of the relation. Moreover, many applications set a unique identifier foreach node and edge of the graph that labels each graph element.

❖Directed:

Depending on the problems the relation between two nodes may besymmetric or not. If the relation is symmetric, the edge might be traversed fromany of the adjacent nodes to the opposite one. If the relation is not symmetric,edges differentiate the head and the tail. The tail of the edge is the node fromwhich the edge starts, and the head of the edge is the node which the edgepoints to. Undirected

graphs are a particular case of directed graphs, since anundirected edge can be represented as two directed edges, each one in a reversedirection of the other.

Node and edge labeling: Some applications differentiate different labels (ortypes) of nodes and edges. Labeling has an important impact because someapplications require distinguishing between different kinds of relations.

For example,a social network may accept either “positive” or “negative” friendshiprelations [33].

❖ Multigraphs:

Multigraphs differ from graphs in that two nodes can be connected by multiple edges.Multigraphs appear commonly when graphs have typededges because often two nodes are related by different categories. For example,in a mobile telephone network that represents the cell phones as the nodes andthe telephone calls as the edges, two nodes will have multiple connections if theyhave called more than once.

❖ Hypergraphs:

A hypergraph is a generalization of the concept of a graph, inwhich the edges are substituted by hyperedges. In contrast to regular edges, anhyperedge connects an arbitrary number of nodes. Hypergraphs are used,Forexample, for building artificial intelligence models [34].

1.3Graph Characterization:

Real graphs are typically very different fromgraphs following the Erd¨os-Renyimodel (random graphs) [35]. Leskovec et al. analyzedover 100 real-world networks in [36] in the following fields: social networks,information/citation networks, collaboration networks, web graphs, Internet networks,bipartite networks, biological networks, low dimensional networks, actornetworks, and product-purchaser networks. The size of these networks variesfrom a few hundred nodes to millions of nodes, and from hundreds of edges tomore than one hundred million. I note that although they might seem huge,the graph data sets of some current applications are significantly larger:

For example - Flickr accounts more than 4 billion photographs that can be tagged andrated [37], and Facebook is publishing more than 25 billion pieces of contenteach month. For these large graphs, one of the most interesting aspects is thatin general most graph metrics (such as the node degree or the edge weight) followpower law distributions [36, 38, 39], and hence some areas of the graph aresignificantly denser than others.

With respect to the characterization of graphs, I summarize some propertiesthat often appear in these huge graphs [23]: (a) they contain a large connectedcomponent that accounts for a significant percentage of the nodes; (b) they aresparse, which means that the number of edges is far away from the maximum number of edges; (c) the degree distribution follows a power law distribution(i.e scale-free networks), where a few nodes have a number of connections thatgreatly exceeds the average, usually known as hubs; (d) the average diameterof each connected components from the graph is small, in other words, from agiven node there is a short path to reach the majority of the remaining nodes inthe connected component (which is also called the small world property); and(e) the nodes are grouped in communities where the density of edges among themembers of the community is larger than the edges going outside the community.Discussion: In this section, I see that graph applications represent theirdatasets following graphs with different degrees of complexity. Nevertheless, Iobserve that the structure of the graphs datasets follow power law characterizationsand properties, which makes it possible to create generic graphs representativeof multiple applications for benchmarking purposes.

According to the previously described applications, I also identify threeaspects that a generic GDB should be able to compute (and thus be included ina benchmark):

(a) labeled graphs, which enable to identify the nodes and edgesof the graph;

(b) directed graphs, which set the relation and its direction;

(c)attribute support, which are used by applications such as for setting the weightof the edges.

1.4Graph Operations

In this subsection, I present several types of operations used in the areas presentedbefore. The analysis of this section will be useful to learn different aspectsthat will be used to fix criteria, in the following subsection, to design relevantoperations for a future benchmark. Table 1 lists some of these graph operations,organized by the type of access that is performed on the graph.
First, I define a set of generic operations. These operations are not typicalin a single specific domain, but common operations that may be necessary inany context. This set of operations allows us to
i.Get atomic information fromthe graph such as getting a node, getting the value of an attribute of an edge, orgetting the neighbor nodes of a specific node, and
ii.Create, delete and transformany graph. Any complex query or transformation of the graph will necessarilyuse these operations.
Afterwards, I extend these operations to other higher level actions typicallyperformed in the scenarios presented before. I group these operations intodifferent types:

❖Traversals.

Traversals are operations that start from a single node and explorerecursively the neighborhood until a final condition, such as the depth orvisiting a target node, is reached. For instance, I consider the operation of calculatinga shortest path, which is the shortest sequence of edges (or the smallestaddition of edge weights in the case of weighted graphs) that connects two nodes.
In a directed graph the direction is restricted to outgoing edges from the tailto the head. Note that shortest paths may be constrained by the value of somenode or edge attributes, as in the case of finding the shortest route from two

Group	Operation	Social Network	Protein Interaction	Recom-mendation	Routing	Analytical	Cascaded	Scale	Attr.	Result
Generic operations										
General Atomic / Local Information Extraction	Get node/edge	+	+	+	+	Yes	No	Neigh.	No	Set
	Get attribute of node/edge	+	+	+	+	Yes	No	Neigh.	No	Set
	Get neighborhood	+	+	+	+	Yes	No	Neigh.	No	Set
	Node degree	+	+	+	+	Yes	No	Neigh.	No	Agr.
General Atomic Transformations	Add/Delete node/edge	+	+	+	+	No	No	Neigh.	No	Set
	Add/Delete/Update attrib.	+	+	+	+	No	No	Neigh.	E/N	Set
Application dependent operations										
Traversals	(Constrained) Shortest Path	+	+		+	Yes	Yes	Glob.	Edge	Graph
	k-hops	+		+	+	Yes	Yes	G/N	No	Graph
Graph Analysis	Hop-Plot	+				Yes	No	Glob.	No	Agr.
	Diameter	+	+			Yes	Yes	Glob.	Edge	Set
	Eccentricity	+				Yes	Yes	Glob.	Edge	Agr.
	Density	+	+			Yes	No	Glob.	No	Agr.
	Clustering coefficient	+				Yes	Yes	Glob.	No	Agr.
Components	Connected Components	+	+			Yes	Yes	Glob.	No	Graph
	Bridges	+	+		+	Yes	Yes	Glob.	No	Set
	Cohesion	+				Yes	Yes	Glob.	No	Set
Communities	Dendrogram	+				Yes	Yes	Glob.	No	Graph
	Max-flow min-cut	+				Yes	Yes	Glob.	Edge	Graph
	Clustering	+	+	+		Yes	Yes	Glob.	No	Graph
Centrality Measures	Degree Centrality	+		+		Yes	No	Glob.	No	Set
	Closeness Centrality	+		+		Yes	Yes	Glob.	No	Set
	Betweenness Centrality	+		+		Yes	Yes	Glob.	No	Set
Pattern Matching	Graph/Subgraph Matching	+	+			Yes	Yes	Neigh.	No	Graph
Graph Anonymization	k-degree Anonym.	+		+		Yes	No	Glob.	No	Graph
	k-neighborhood Anonym.	+		+		Yes	Yes	Glob.	No	Graph
Other Operations (Similarity, ranking,...)	Structural Equivalence	+				Yes	Yes	Glob.	No	Graph
	PageRank			+		Yes	No	Glob.	Node	Set

Table 1. Graph Op erations, Areas of Interest and Categorization

points, avoiding a certain type of road, for instance. This operation is also usedas a measure to calculate the information loss of graph anonymization methods.Another typical operations is

calculating k-hops, which returns all the nodes thatare at a distance of k edges from the root node. A particular case is when k = 1,also known as the neighbors of the node. The particular case of 1-hops is widelyused as part of other operations. For example to calculate the nearest neighbor-hood in recommender systems, to obtain a particular user's neighborhood withsimilar interest, or in web ranking using hubs and authorities.

❖ **Graph Analysis.**

Basic graph analysis includes the study of the topologyof graphs to analyze their complexity and to characterize graph objects. It isbasically used to verify some specific data distributions, to evaluate a potentialmatch of a specific pattern, or to get detailed information of the role of nodesand edges. In several situations graph analysis is the first step of the analyticalprocess and it is widely used in SNA and protein interaction analysis. Amongthis I may calculate the hop-plot (a metric to measure the rate of increase ofthe neighborhood depending on the distance to a source node), the (effective)diameter, the density, or the clustering coefficient (to measure the degree oftransitivity of the graph), to give some examples.

❖ **Components.**

A connected component is a subset of the nodes of the graphwhere there exists a path between any pair of nodes. Thus, a node only belongsto a single connected component of the graph. Finding connected components isusually crucial in many operations, typically used in a preprocess phase. Also,some operations are helpful to study the vulnerability of a graph, or the probabilityto separate a connected component into two other components. Findingbridges, edges whose removal would imply separating a connected component, isimportant in many applications. Going further, the cohesion of the graph can becomputed by finding the minimum number of nodes that disconnect the groupif removed.

❖ **Communities.**

A community is generally considered to be a set of nodeswhere each node is closer to the other nodes within the community than to nodesoutside it. This effect has been found in many real-world graphs, especially socialnetworks. Operations related to the creation of a community may be buildingdendograms (communities formed through hierarchical clustering), finding theminimal-cut set of edges or other clustering techniques.Centrality Measures. A centrality measure aims at giving a rough indicationof the social importance of a node based on how well this node connects thenetwork. The most well-known centrality measures are degree, closeness andbetweenness centrality.

❖ **Pattern matching.**

Pattern recognition deals with algorithms which aim atrecognizing or describing input patterns. Graph matchings are usually categorizedinto exact or approximate. Exact matchings may include finding homomorphismsor (subgraph) isomorphisms. Approximate matchings may includeerror-correcting (subgraph) isomorphisms, distance-based matching, etc.

3.5 Query Categorization

The computational requirements of graph queries is not homogeneous. For example,some queries may traverse the full graph, while others may request theoutdegree of a single node. In order to build a balanced benchmark it must berepresentative of the different types of operations that can be issued by an applicationto the graph database. In this section, I build up a set of categoriesto classify the different operations that are issued to a graph database:

❖ **Transformation/Analysis:**

I distinguish between two types of operationsto access the database: transformations and analysis operations. The first groupcomprise operations that alter the graph database: bulk loads of a graph, add/removenodes or edges to the graphs, create new types of nodes/edges/attribute or modify the value of an attribute. The rest of queries are considered analysisqueries. Although an analysis query does not modify the graph, it may need toaccess to secondary storage because the graph or the temporary results generatedduring the query resolution are too large to fit in memory.

❖ **Cascaded access:**

I differentiate two access patterns to the graph: cascadedand not cascaded. I say that an operation follows a cascaded pattern if thequery performs neighbor operations with a depth at least two. For example, a3-hops operation follows a cascaded pattern. Thus, a non cascaded operationmay access a node, an edge or the neighbours of a node. Besides, an operationthat does not request the neighbours of a node, though it may access the fullgraph, is a non cascaded operation. For instance, an operation that returns thenode with the largest value of an attribute accesses all nodes, but since it doesnot follow the graph structure is a non-cascaded operation.

❖ **Scale:**

I classify the queries depending on the number of nodes accessed. Idistinguish two types of queries: global and neighbourhood queries. The former type corresponds to queries that access the complete graph structure. In otherwords, I consider as global queries those that access to all the nodes or theedges of the graph. The latter queries only access to a portion of the graph.

❖ **Attributes:**

Graph databases do not only have to manage the structural informationof the graph, but also the data associated to the entities of the graph.Here, I classify the queries according to the attribute set that it accesses: edgeattribute set, node attribute set, mixed attribute set or no attributes accessed.

❖ **Result:**

I differentiate three different types of results: graphs, aggregatedresults, and sets. The most typical output for a graph database query is anothergraph, which is ordinarily a transformation, a selection or a projection of theoriginal graph, which includes nodes and edges. An example of this type of resultis getting the minimum spanning tree of a graph, or finding the minimum lengthpath that connects two nodes. The second type of results build up aggregates,whose most common application is to summarize properties of the graph. Forinstance, a histogram of the degree distribution of the nodes, or a histogram ofthe community size are computed as aggregations. Finally, a set is an output thatcontains either atomic entities or result sets that are not structured as graphs. For example, the selection of one node of a graph or finding the edges with thegreatest weight are set results.

❖ **Discussion:**

Queries in a benchmark must represent the workload of the realenvironment where the application is going to be executed, and thus should beadapted to the particular application profile to be tested. I have seen thatgraph operations are diverse, but many operations share similar operationalpatterns. In Table 1, I summarize these patterns categorizing the catalog ofpopular operations. In this table, I find that although the most basic operationsare neither structured nor affect large fractions of the graph, many applicationsuse large scale operations that traverse the graph. Furthermore, I find thatmost graph operations are accessing the information stored in the edges sincethe

attributes in the edges (and weights in particular) are modeling the relationsbetween entities, which is the main objective of a graph. I also observe thatgeneric GDB must be able to store temporal objects because they are necessaryfor most operations (e.g: storing a boolean to decide whether a node has beenvisited or counting the number of paths through a node). Finally, I see thatgeneric GDB must be able to manage different types of result sets because I findoperations in many different applications that return sets, aggregates and graphs.

3.6 Experimental setting

Experimental setup and measurement is one of the most important parts of abenchmark and it must be clearly defined and configured to allow a fair comparisonbetween multiple solutions in different environments. Early databasebenchmarks only focused in the response time of the queries. As the benchmarkshave evolved and platforms are more complex, the measurements havebecome more sophisticated and, in consequence, the experimental setup and thetesting process is more expensive and complicated. For example, early benchmarkslike Wisconsin only considered the response time using very simple scalefactors [41]. Later, TPC-C or TPC-H introduced a metric with the relationshipof the pricing with respect the maximum throughput [42], and more recentlyLUBM [14], for the benchmarking of RDF graphs, defined a combined metricbetween the query response time and the answer completeness and soundness inthe case of partial results and pattern matching.
The different concepts related to the experimental setup and configuration ofa graph database benchmark can be grouped in the following areas: configurationand setup, experimental process, and measures.

❖ Configuration and setup:

Modern benchmarks allow for the definition of thedataset size by means of a scale factor, which fixes the dataset size generatedfollowing precise data distributions. For graphs, the scale factor defines the numberof nodes and edges. Additionally, the graph must conform to scale the graphstructural properties for different scales such as the vertex degree and hop-plotdistributions, the diameter or the community structure [43].

❖ Measures:

Measures are closely related to the benchmark capabilities andto the audience. Some general measures that can be applied to graphs are thefollowing:
(a) the load time, which measures the elapsed time for loading andpreprocess the dataset;
(b) the size of the graph;
(c) the query response timethat accounts for the time elapsed between the query is issued until the resultsare output;
(d) the throughput that measures the number of queries completedin an interval of time;
(e) the price of the computing site including hardware,license and maintenance costs if applicable; or
(f) the power consumption thatmeasures the energy requirements of the computing equipment and gives anindirect measure of its cooling requirements.

In order to compare such metricsamong different platforms, it is common to introduce normalized metrics such asthe price/throughput or the power/throughput that enable an easier comparisonof benchmarking results between, for example, a supercomputer and a laptophardware setup.

In the area of graph benchmarking, some specialized metrics have been proposed.The HPC benchmark [15] defines the number of traversals per second(TEPS) as the average number of edges explored during a traversal per second.The TEPS gives a measure of the average effort to navigate through the relationshipswith respect to the size of the graph. For pattern matching, when itis difficult to find all the results in a deterministic way like in knowledge bases,Guo et al. have proposed to measure the query completeness (or recall) or howfar is the result respect to all the possible answers [14]. If the

GDB is able toreturn results as soon as they appear, it might be interesting to obtain a plot ofthe GDB's recall with respect to time.

❖ **Discussion:**

There are many experimental settings and measures that can beselected for a GDB benchmark. The final selection depends on the benchmarkcharacteristics, the goals and the audience. Thus, an industrial-oriented benchmarkwill probably focus on the throughput and cost of complex queries on very large graphs. However, research oriented benchmarking may be interested in theperformance of more specific operations for different hardware configurations.In contrast to other scenarios where ACID properties are mandatory, I believe that many applications of GDB benchmarks allow for more relaxed consistencybehaviors.

4. CONCLUSION

In this paper, I have analyzed important aspects for the design of a GDBbenchmark. First of all, there is a significant core of applications that benefitsignificantly from their management as a graph, like social network analysis, proteininteraction, recommendation and routing among others. These applicationsjustify by themselves the existence and evolution of GDBs, and at the same time,justify the existence and evolution of a GDB benchmark. Its correct design andimplementation implies the following aspects to be considered:

- The characteristics of the graph to be used in a benchmark are important.Considering the inclusion of directed graphs with attributes, with differentnode and edge types in the context of multigraphs would be important.
- The characteristics of the graph like the distribution of edges per node, attributevalues per edge, etc. depend on the application and should be appliedbased on the different studies appeared in the literature. Nevertheless, mosthuge graph datasets follow power law distributions.
- Although not necessarily all the operations appearing in our analysis needto be considered for a benchmark, both analytical and transformation operationsshould be present.
- The cascaded nature of many graph operations is important, and a benchmarkshould include a good selection of operations with and without thiscascaded nature.
- While there are operations that cover a traversal of all the database, othersjust affect a few of their components. Such feature should be evaluated takinginto consideration the metrics to be used, in order to balance the importanceof each case.
- Depending on the application, some operations just evaluate the structureof the graph, while others take the attributes in the nodes and specially inthe edges, to be evaluated. A good combination of queries with both characteristicswould be of interest for the proper deployment of a benchmark.
- The nature of the result is important because GDBs are capable of offeringa good assortment of answers. In particular, operations returning setsof nodes, graphs or aggregational answers would be recommended for a balancedbenchmark.
- There are other aspects that influence the fairness of a benchmark. Thoseare the configuration of the GDB (partitioning, indexing of attributes, dataredundancy and reorganization), the way the experimental process is undertaken(warm-up of the database, query generation and the observational procedure) and the metrics to be considered (load time, repository size,query response time, throughput obtained and the cost of the deployment).

However, those aspects are not totally influenced by the GDB environment. Just to finalize, it would be very important to find an application coveringas many of the aspects that I have evaluated in this paper as possible. Ibelieve that social network analysis is very significant because it covers almostall the operation types in graph databases, it is easy to understand by the finaluser and carries a lot of natural queries with answers that can be conceptuallyunderstandable. Any other application with such characteristics would be of useand beneficial for a GDB benchmark.

REFERENCES

1.Angles, R., Guti´errez, C.: Survey of graph database models. ACM Comput. Surv.40(1) (2008)
2. Neo4j: The neo database. Available at http://dist.neo4j.org/neo-technologyintroduction.pdf (2006)
3.HypergraphDB: HypergraphDB website.http://www.kobrix.com/hgdb.jsp(Lastretrieved in March 2010)
4.Infogrid: Blog. http://infogrid.org/blog/2010/03/operations-on-a-graph-databasepart-(Last retrieved in March 2010).
5.Retrieval. In: CIKM. (2007) 573–582
6.Jena-RDF: Jena documentation. http://jena.sourceforge.net/documentation.html(Last retrieved in March 2010)
7.AllegroGraph: AllegroGraph website. http://www.franz.com/agraph/ (Last retrievedin May 2010)
8.E. Prud'hommeaux and A. Seaborne: SPARQL Query Language for RDF. W3C.Available at http://www.w3.org/TR/rdf-sparql-query/ (2008)
9.Gremlin website: Gremlin documentation. http://wiki.github.com/tinkerpop/gremlin/(Last retrieved in Jun 2010)
10.Transaction Processing Performance Council (TPC): TPC Benchmark. TPC website.http://www.tpc.org (Last retrieved in Jun 2010)
Cattell, R., Skeen, J.: Object operations benchmark. TODS 17(1) (1992) 1–31
11. Carey, M., DeWitt, D., Naughton, J.: The oo7 benchmark. In: SIGMOD Conference.(1993) 12–21
12.Schmidt, A., Waas, F., Kersten, M., Carey, M., Manolescu, I., Busse, R.: Xmark:A benchmark for xml data management. In: VLDB. (2002) 974–98
13.Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems.J. Web Sem. 3(2-3) (2005) 158–182
14. Bader, D., Feo, J., Gilbert, J., Kepner, J., Koetser, D., Loh, E., Madduri, K.,Mann, B., Meuse, T., Robinson, E.: HPC Scalable Graph Analysis Benchmarkv1.0. HPC Graph Analysis (February 2009)
15.Dominguez-Sal, D., Urb´on-Bayes, P., Gim´enez-Va˜n´o, A., G´omez-Villamor, S.,Mart´ınez-Baz´an, N., Larriba-Pey, J.L.: Survey of graph database performanceon the hpc scalable graph analysis benchmark. In: IWGD. (2010)
16.INSNA: International network for social network analysis. http://www.insna.org/
17.18. OReilly, T.: What is Web 2.0: Design patterns and business models for the nextgeneration of software. (2005)
18.Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: from relations tosemistructured data and XML. Morgan Kaufmann Publishers Inc. (2000)
19.Brickley, D., Guha, R.V.: Resource description framework (rdf) schema specification1.0. W3C Candidate Recommendation (2000)
20.Shasha, D., Wang, J., Giugno, R.: Algorithmics and applications of tree and graphsearching. In: PODS, New York, NY, USA, ACM (2002) 39–52
21.Anyanwu, K., Sheth, A.: _-queries: Enabling querying for semantic associationson the semantic web. In: WWW, ACM Press (2003) 690–699
22.Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms.ACM Computing Surveys (CSUR) 38(1) (2006) 2
23.BioGRID: General repository for interaction datasets. http://www.thebiogrid.org/
24.PDB: Rcsb protein data bank. http://www.rcsb.org/
25.NAViGaTOR. http://ophid.utoronto.ca/navigator/
26.Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine.Computer networks and ISDN systems 30(1-7) (1998) 107–117
27.Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM46(5) (1999) 604–632
28.Strands: e-commerce recommendation engine. http://recommender.strands.com/
29.Chein, M., Mugnier, M.: Conceptual graphs: fundamental notions. Revued'IntelligenceArtificielle 6 (1992) 365–406
30.DirectedEdge: a recommendation engine. Available athttp://www.directededge.com (Last retrieved

in Jun 2010)
31.Amadeus: Global travel distribution system. http://www.amadeus.net/
32.Leskovec, J., Huttenlocher, D., Kleinberg, J.: Signed networks in social media. In:CHI. (2010) 1361–1370
33.Goertzel, B.: OpenCog Prime: Design for a Thinking Machine. Online wikibook, at http://opencog.org/wiki /OpenCogPrime(2008)
34.. Erdos, P., Renyi, A.: On random graphs. Mathematicae 6(290-297) (1959) 156
35.Leskovec, J., Lang, L., Dasgupta, A., Mahoney, M.: Statistical properties of communitystructure in large social and information networks. In: WWW. (2008)695–704
36.Flickr: Four Billion. http://blog.flickr.net/en/2009/10/12/4000000000/ (Last retrievedin Jun 2010)
37.Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of theinternet topology. In: SIGCOMM. (1999) 251–262
38.McGlohon, M., Akoglu, L., Faloutsos, C.: Weighted graphs and disconnected components:patterns and a generator. In: KDD. (2008) 524–53
39.Bader, D., Madduri, K.: Parallel algorithms for evaluating centrality indices inreal-world networks. In: ICPP. (2006) 539–550
40.Bitton, D., DeWitt, D., Turbyfill, C.: Benchmarking database systems a systematicapproach. In: VLDB. (1983) 8–19
41.Transaction Processing Performance Council (TPC): TPC Benchmark H (2.11).TPC website. http://www.tpc.org/tpch/ (Last retrieved in Jun 2010)
42Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kroneckergraphs: An approach to modeling networks. Journal of Machine LearningResearch 11 (2010) 985–1042

Rahul Arun Deshmukh
MSc in Research in Computer and System Engineering Technische Universitat Ilmenau, Germany