# Efficient Graph Management Based On Bitmap Indices

19-2-11

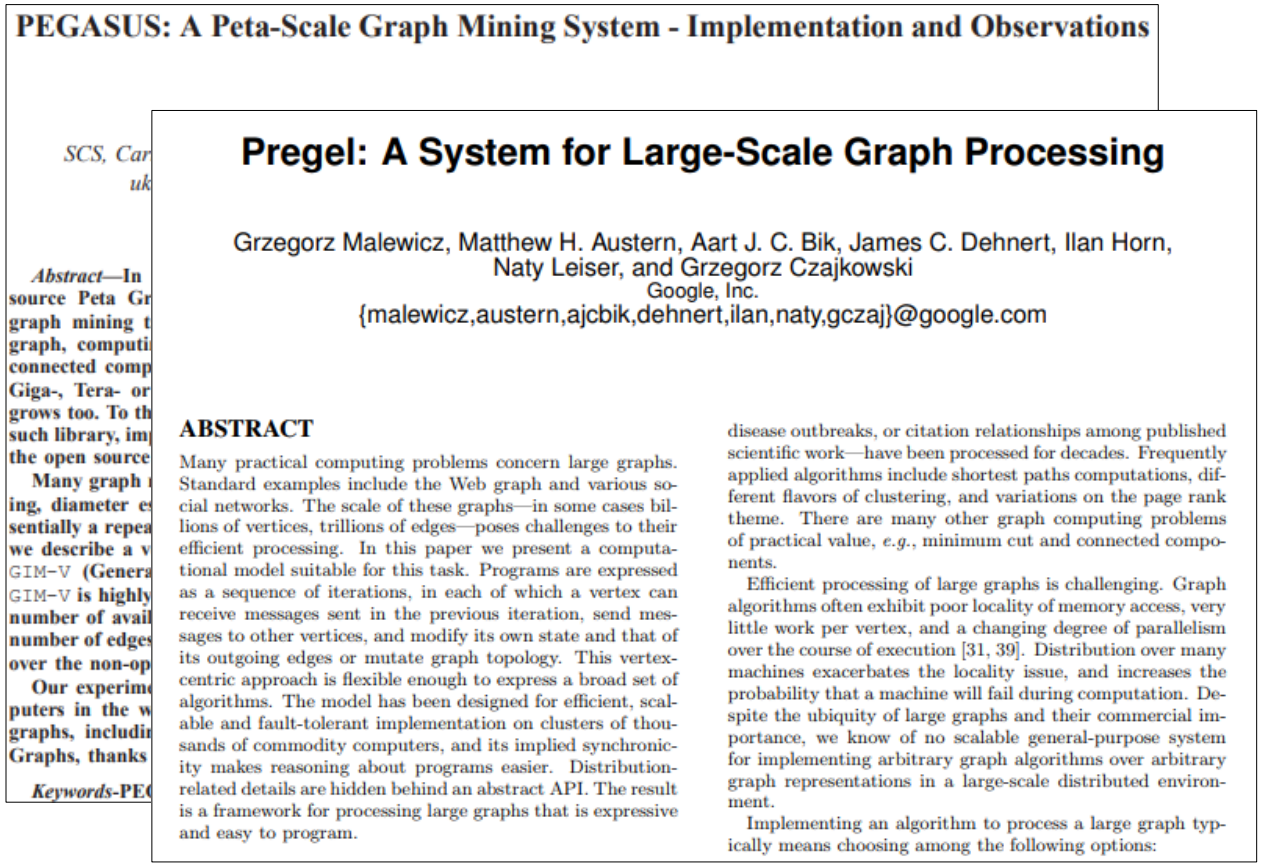
**Norbert Martínez-Bazan et al.**

Presenter: Camilo Muñoz

UNIVERSITY OF WATERLOO

# Paper Background

- 2 approaches to graph analysis



PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations

SCS, Car
uk

*Abstract*—In
source Peta Gr
graph mining t
graph, computi
connected comp
Giga-, Tera- or
grows too. To th
such library, imp
the open source
Many graph
ing, diameter es
sentially a repea
we describe a v
GIM-V (Genera
GIM-V is highly
number of avail
number of edges
over the non-op
Our experime
puters in the w
graphs, includi
Graphs, thanks

*Keywords*-PE

Pregel: A System for Large-Scale Graph Processing

Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn,
Naty Leiser, and Grzegorz Czajkowski
Google, Inc.
{malewicz,austern,ajcbik,dehnert,ilan,naty,gczaj}@google.com

**ABSTRACT**

Many practical computing problems concern large graphs. Standard examples include the Web graph and various social networks. The scale of these graphs—in some cases billions of vertices, trillions of edges—poses challenges to their efficient processing. In this paper we present a computational model suitable for this task. Programs are expressed as a sequence of iterations, in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. This vertex-centric approach is flexible enough to express a broad set of algorithms. The model has been designed for efficient, scalable and fault-tolerant implementation on clusters of thousands of commodity computers, and its implied synchronicity makes reasoning about programs easier. Distribution-related details are hidden behind an abstract API. The result is a framework for processing large graphs that is expressive and easy to program.

disease outbreaks, or citation relationships among published scientific work—have been processed for decades. Frequently applied algorithms include shortest paths computations, different flavors of clustering, and variations on the page rank theme. There are many other graph computing problems of practical value, *e.g.*, minimum cut and connected components.

Efficient processing of large graphs is challenging. Graph algorithms often exhibit poor locality of memory access, very little work per vertex, and a changing degree of parallelism over the course of execution [31, 39]. Distribution over many machines exacerbates the locality issue, and increases the probability that a machine will fail during computation. Despite the ubiquity of large graphs and their commercial importance, we know of no scalable general-purpose system for implementing arbitrary graph algorithms over arbitrary graph representations in a large-scale distributed environment.

Implementing an algorithm to process a large graph typically means choosing among the following options:

neo4j
Java Driver 1.5.0-beta03

HyperGraphDB

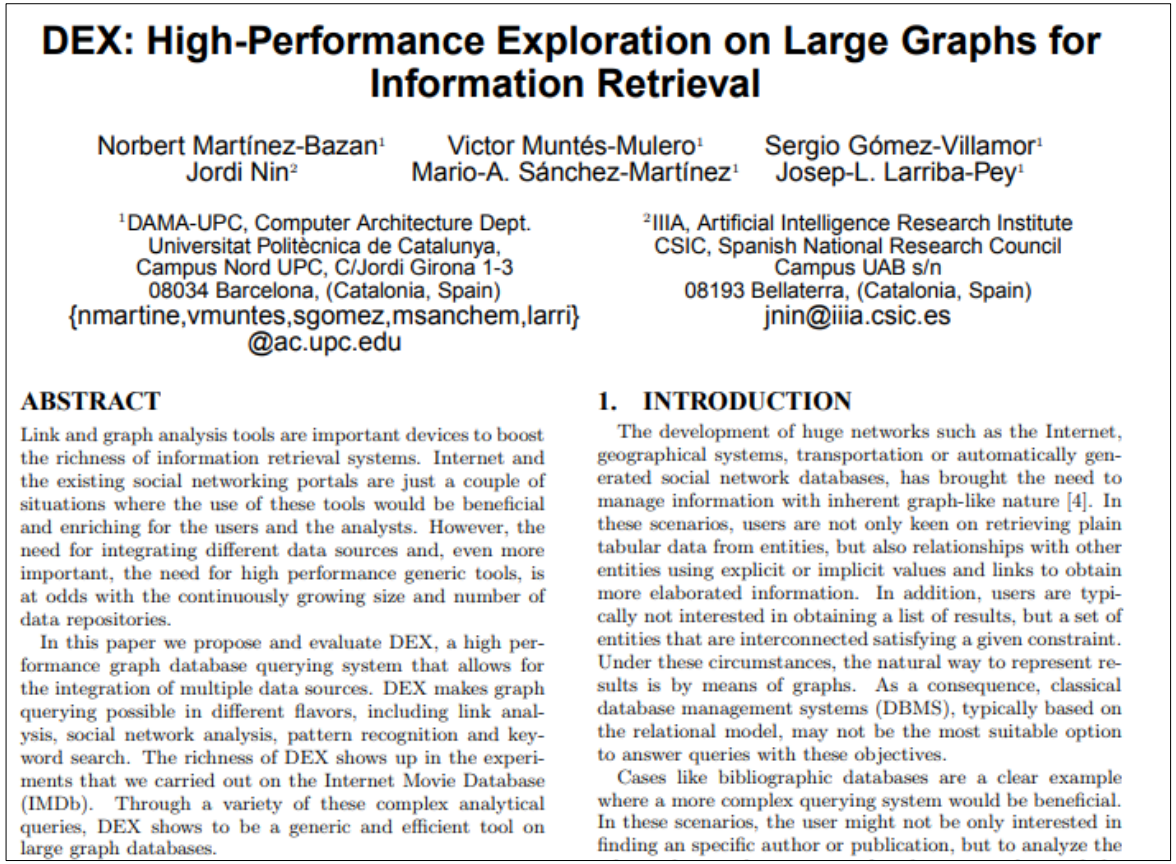
OrientDB®



UNIVERSITY OF WATERLOO

# Introduction

- Use of bitmaps to reduce cost of graph operations: Performance ↑
- 2 key aspects → less space and efficient operations



**DEX: High-Performance Exploration on Large Graphs for Information Retrieval**


Norbert Martínez-Bazan[1] Victor Muntés-Mulero[1] Sergio Gómez-Villamor[1]
Jordi Nin[2] Mario-A. Sánchez-Martínez[1] Josep-L. Larriba-Pey[1]

[1]DAMA-UPC, Computer Architecture Dept.
Universitat Politècnica de Catalunya,
Campus Nord UPC, C/Jordi Girona 1-3
08034 Barcelona, (Catalonia, Spain)
{nmartine,vmuntes,sgomez,msanchem,larri}
@ac.upc.edu

[2]IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB s/n
08193 Bellaterra, (Catalonia, Spain)
jnin@iiia.csic.es



**ABSTRACT**

Link and graph analysis tools are important devices to boost the richness of information retrieval systems. Internet and the existing social networking portals are just a couple of situations where the use of these tools would be beneficial and enriching for the users and the analysts. However, the need for integrating different data sources and, even more important, the need for high performance generic tools, is at odds with the continuously growing size and number of data repositories.

In this paper we propose and evaluate DEX, a high performance graph database querying system that allows for the integration of multiple data sources. DEX makes graph querying possible in different flavors, including link analysis, social network analysis, pattern recognition and keyword search. The richness of DEX shows up in the experiments that we carried out on the Internet Movie Database (IMDb). Through a variety of these complex analytical queries, DEX shows to be a generic and efficient tool on large graph databases.


**1. INTRODUCTION**

The development of huge networks such as the Internet, geographical systems, transportation or automatically generated social network databases, has brought the need to manage information with inherent graph-like nature [4]. In these scenarios, users are not only keen on retrieving plain tabular data from entities, but also relationships with other entities using explicit or implicit values and links to obtain more elaborated information. In addition, users are typically not interested in obtaining a list of results, but a set of entities that are interconnected satisfying a given constraint. Under these circumstances, the natural way to represent results is by means of graphs. As a consequence, classical database management systems (DBMS), typically based on the relational model, may not be the most suitable option to answer queries with these objectives.

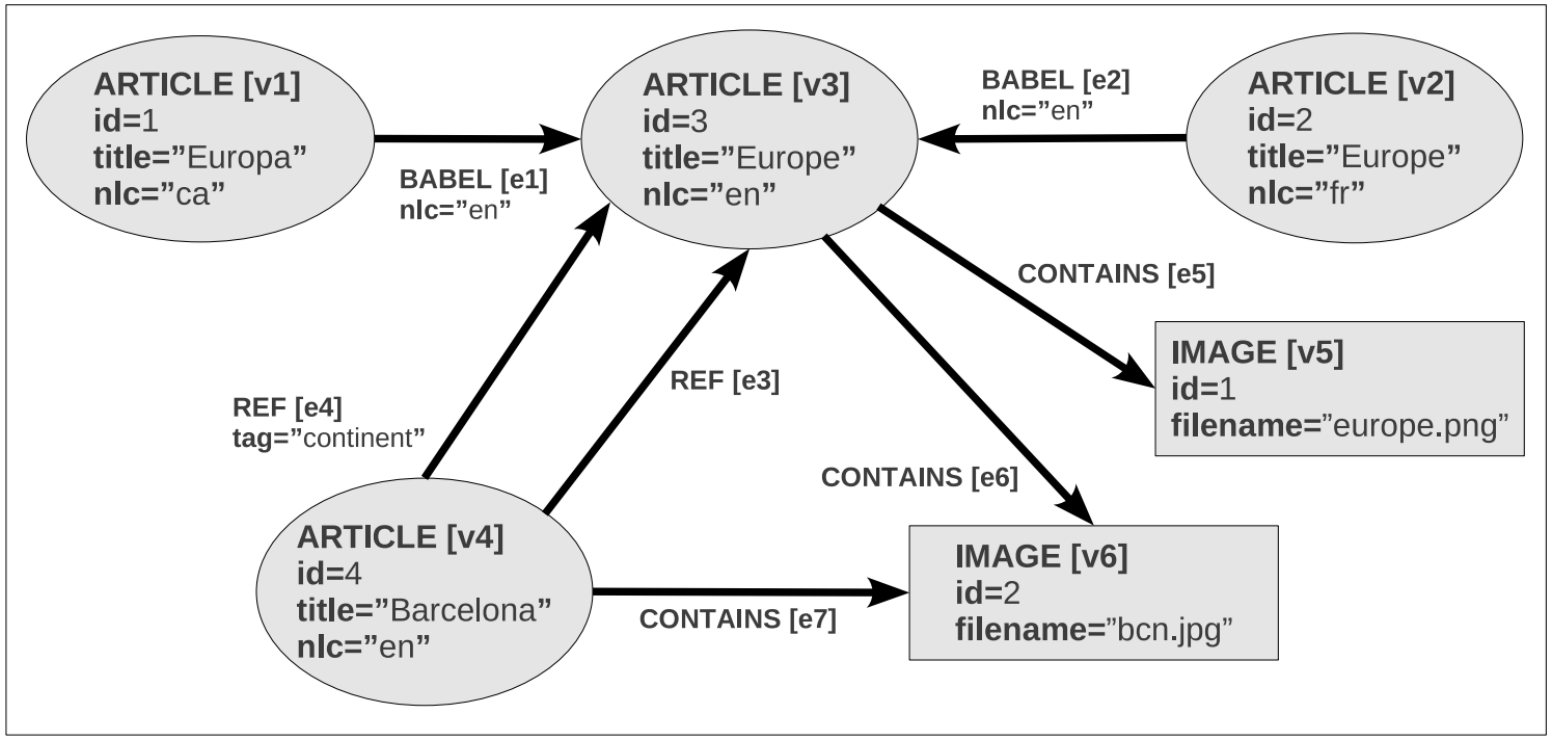
Cases like bibliographic databases are a clear example where a more complex querying system would be beneficial. In these scenarios, the user might not be only interested in finding an specific author or publication, but to analyze the



UNIVERSITY OF WATERLOO

# Graph Model

- Labeled and directed attributed multigraph
- Multiple lists of pairs of values → <vertex/edge ID, value>



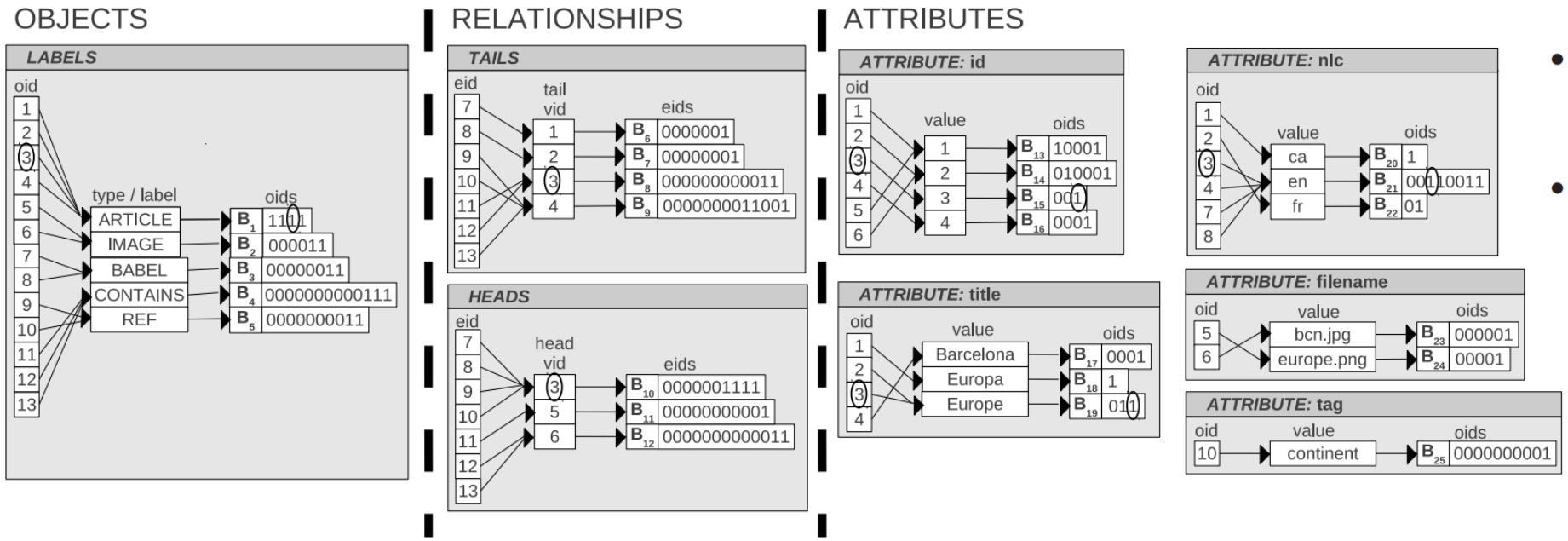
# Graph Representation

Table 1: Wikipedia transformation into value sets

| Collection | Graph $G$ | Transformed Graph $G$ |
|---|---|---|
| $L$ | $(v_1, \texttt{ARTICLE}), (v_2, \texttt{ARTICLE}), (v_3, \texttt{ARTICLE}),$ $(v_4, \texttt{ARTICLE}), (v_5, \texttt{IMAGE}), (v_6, \texttt{IMAGE}), (e_1, \texttt{BABEL}),$ $(e_2, \texttt{BABEL}), (e_3, \texttt{REF}), (e_4, \texttt{REF}), (e_5, \texttt{CONTAINS}),$ $(e_6, \texttt{CONTAINS}), (e_7, \texttt{CONTAINS})$ | $(\texttt{ARTICLE}, \{v_1, v_2, v_3, v_4\}), (\texttt{BABEL}, \{e_1, e_2\}),$ $(\texttt{CONTAINS}, \{e_5, e_6, e_7\}), (\texttt{IMAGE}, \{v_5, v_6\}),$ $(\texttt{REF}, \{e_3, e_4\})$ |
| $T$ | $(e_1, v_1), (e_2, v_2), (e_3, v_4), (e_4, v_4),$ $(e_5, v_3), (e_6, v_3), (e_7, v_4)$ | $(v_1, \{e_1\}), (v_2, \{e_2\}), (v_3, \{e_5, e_6\}),$ $(v_4, \{e_3, e_4, e_7\})$ |
| $H$ | $(e_1, v_3), (e_2, v_3), (e_3, v_3), (e_4, v_3),$ $(e_5, v_5), (e_6, v_6), (e_7, v_6)$ | $(v_3, \{e_1, e_2, e_3, e_4\}), (v_5, \{e_5\}),$ $(v_6, \{e_6, e_7\})$ |
| $A_{id}$ | $(v_1, 1), (v_2, 2), (v_3, 3), (v_4, 4), (v_5, 1), (v_6, 2)$ | $(1, \{v_1, v_5\}), (2, \{v_2, v_6\}), (3, \{v_3\}), (4, \{v_4\})$ |
| $A_{title}$ | $(v_1, \texttt{Europa}), (v_2, \texttt{Europe}),$ $(v_3, \texttt{Europe}), (v_4, \texttt{Barcelona})$ | $(\texttt{Barcelona}, \{v_4\}), (\texttt{Europa}, \{v_1\}), (\texttt{Europe}, \{v_2, v_3\})$ |
| $A_{nlc}$ | $(v_1, \texttt{ca}), (v_2, \texttt{fr}), (v_3, \texttt{en}), (v_4, \texttt{en}), (e_1, \texttt{en}), (e_2, \texttt{en})$ | $(\texttt{ca}, \{v_1\}), (\texttt{en}, \{v_3, v_4, e_1, e_2\}), (\texttt{fr}, \{v_2\})$ |
| $A_{filename}$ | $(v_5, \texttt{europe.png}), (v_6, \texttt{bcn.jpg})$ | $(\texttt{bcn.jpg}, \{v_6\}), (\texttt{europe.png}, \{v_5\})$ |
| $A_{tag}$ | $(e_4, \texttt{continent})$ | $(\texttt{continent}, \{e_4\})$ |

# Graph Representation

- Graph model sets ⟶ Value sets
- 5 operations on value sets → domain, objects, lookup, insert, and remove




- Number of articles:

$$|objects(\texttt{LABELS}, \texttt{'ARTICLE'})| = 4$$

- Articles in French or Catalan:

$$(objects(\texttt{NLC}, \texttt{'fr'}) \cup objects(\texttt{NLC}, \texttt{'ca'})) \cap objects(\texttt{LABELS}, \texttt{'ARTICLE'}) = \{1, 2\}$$

# Experimental Evaluation

- Dataset: January 2010 Wikipedia dump → 55 million articles, 254 languages, 2.1 million images and 321 million links.
- Multiple types of queries → k-hops and path traversals (Q1, Q2), graph pattern matching (Q3), aggregations and edge connectivity (Q4), and graph transformation (Q5)

# Performance analysis

| | MonetDb | MySQL | Neo4j* | DEX |
|---|---|---|---|---|
| Graph Size (GB) | **12.00** | 15.72 | 42.00 | 16.98 |
| Load (h) | *Error* | **1.36** | 8.99 | 2.89 |
| Q1 (s) | 4,801.6 | > *12 h.* | > *12 h.* | **120.5** |
| Q2 (s) | 3,788.4 | 13,841.6 | > *12 h.* | **205.4** |
| Q3 (s) | 458.9 | 33.0 | 481 | **10.8** |
| Q4 (s) | 279.3 | **45.0** | > *12 h.* | 144.9 |
| Q5 (s) | 267.4 | 930.3 | > *12 h.* | **140.9** |

# Performance in-memory

| | Matrix | MonetDb | MySQL | Neo4J | DEX |
|---|---|---|---|---|---|
| Memory (GB) | 20.00 | 20.00 | 12.00 | 45.00 | 20.00 |
| Graph Size (GB) | 15.09 | 12.00 | 15.92 | 42.00 | 16.98 |
| Load (h) | 0.32 | 0.74 | 1.15 | 8.99 | 2.25 |
| Q1 (s) | 12.97 | 106.14 | $> 12\ h.$ | $> 12\ h.$ | 118.93 |
| Q2 (s) | 51.86 | 120.50 | 8,896.00 | $> 12\ h.$ | 205.97 |
| Q3 (s) | 6.28 | 7.56 | 29.83 | 481.00 | 10.68 |
| Q4 (s) | 31.65 | 84.97 | 39.71 | $> 12\ h.$ | 146.77 |
| Q5 (s) | 76.15 | 48.34 | 909.24 | $> 12\ h.$ | 141.06 |

# Key Take away

Thanks to the bitmap representation less space is required and graph operations can be performed efficiently using binary logic

# DEX → Sparksee

*Sparsity

## Beyond Macrobenchmarks: Microbenchmark-based Graph Database Evaluation

Matteo Lissandrini
Aalborg University
matteo@cs.aau.dk

Martin Brugnara
University of Trento
mb@disi.unitn.eu

Yannis Velegrakis
University of Trento
velgias@disi.unitn.eu

**ABSTRACT**

Despite the increasing interest in graph databases their requirements and specifications are not yet fully understood by everyone, leading to a great deal of variation in the supported functionalities and the achieved performances. In this work, we provide a comprehensive study of the existing graph database systems. We introduce a novel microbenchmarking framework that provides insights on their performance that go beyond what macro-benchmarks can offer. The framework includes the largest set of queries and operators so far considered. The graph database systems are evaluated on synthetic and real data, from different domains, and at scales much larger than any previous work. The framework is materialized as an open-source suite and is easily extended to new datasets, systems, and queries[1].

**PVLDB Reference Format:**
Matteo Lissandrini, Martin Brugnara, and Yannis Velegrakis. Beyond Macrobenchmarks: Microbenchmark-based Graph Database Evaluation. *PVLDB*, 12(4): 390–403, 2018.
DOI: https://doi.org/10.14778/3297753.3297759

There are two categories of graph management systems that address two complementary yet distinct sets of functionalities. The first is that of *graph processing systems* [27, 42, 44], which analyze graphs to discover characteristic properties, e.g., average connectivity degree, density, and modularity. They also perform batch analytics at large scale, implementing computationally expensive graph algorithms, such as PageRank [54], SVD [23], strongly connected components identification [63], and core identification [12, 20]. Those are systems like GraphLab, Giraph, Graph Engine, and GraphX [67]. The second category is that of *graph databases* (GDB for short) [5]. Their focus is on storage and querying tasks where the priority is on high-throughput and transactional operations. Examples in this category are Neo4j [51], OrientDB [53], Sparksee [60] (formerly known as DEX), Titan [64] (recently renamed to JanusGraph), ArangoDB [11] and BlazeGraph [62]. To make this distinction clear, graph processing systems, can, in some sense, be seen as the graph world parallel to OLAP systems, while graph databases as the parallel to the OLTP systems.

GRAPH PROCESSING IN
MAIN-MEMORY COLUMN STORES

Zur Erlangung des akademischen Grades

**Doktor-Ingenieur (Dr.–Ing.)**

der Fakultät für Informatik
der Technischen Universität Dresden

vorgelegte

**Dissertation**

von

MARCUS PARADIES

geboren am 10. Februar 1987 in Ilmenau.

TECHNISCHE UNIVERSITÄT DRESDEN



UNIVERSITY OF WATERLOO

# DEX → Sparksee

| | L | C | R | | | U | D | | T | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Load | Insertions | Graph Statistics | Search by Property\Label | Search by Id | Updates | Delete Node | Other Deletions | Neighbors | Node | Edge-Labels | Degree Filter | BFS | Shortest Path |
| ArangoDB | | ✓ | ⚠ | ⚠ | | ✓ | ✓ | ✓ | | | | ⚠ | ⚠ | ⚠ |
| BlazeGraph | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ | ⚠ |
| Neo4J (v.1.9) | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Neo4J (v.3.0) | ✓ | | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| OrientDB | | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ | | | ✓ | |
| Sparksee | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | | | | | | ⚠ |
| Titan (v.0.5) | ⚠ | | ⚠ | ⚠ | | | | | | | | | | |
| Titan (v.1.0) | ⚠ | | ⚠ | ⚠ | | | | | | | | | | ⚠ |
| Sqlg | | | | ✓ | | ✓ | | ✓ | ⚠ | ⚠ | | ⚠ | ⚠ | ⚠ |

# Discussion

- Besides B+ trees, what alternative data structures could be used to implement the mappings?
- Nature of graph-based data can change based on the application domain
- Native Graph DBs and Hybrid Systems: What are the key considerations when selecting a graph-oriented system?