BENCHMARKING ESSENTIAL GRAPH QUERIES

Renzo Angles

Assistant Professor, Universidad de Talca, Chile
Postdoc, Vrije Universiteit Amsterdam

1st Workshop on Graph-based Technologies and Applications (Graph-TA)
Barcelona, February 19, 2013

AGENDA

1. INTRODUCTION
2. THE BENCHMARK
3. PRELIMINARY EXPERIMENTAL RESULTS
4. CONCLUSIONS

AGENDA

1. INTRODUCTION
2. THE BENCHMARK
3. PRELIMINARY EXPERIMENTAL RESULTS
4. CONCLUSIONS

Motivation

- Increasing amount of graph data (e.g., social networks)
- How to store graph data?
  - Graph (oriented) databases
  - RDF Triple stores (RDF databases)
  - NOSQL databases?
- What is the most suitable graph database?
  - Theoretical comparison (complexity and expressive power)
  - Empirical comparison (performance, usability, etc.)
  - Benchmarks for GDBs (there is not a standard one)
  - The application domain is very important

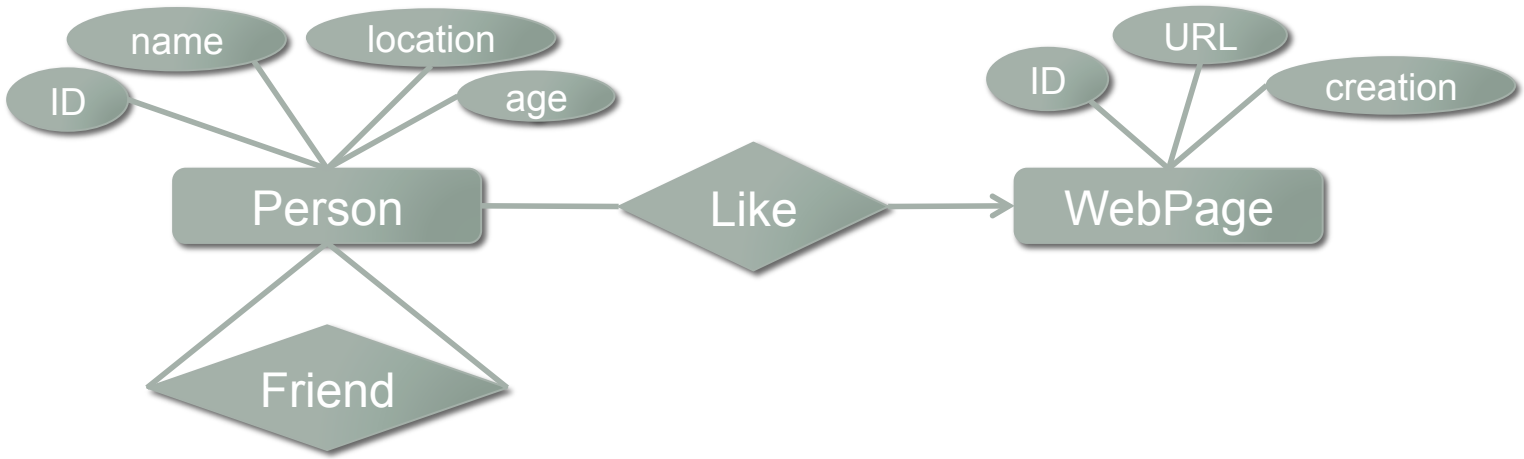
Our work

- Development of a benchmark for graph databases
  - Graph data based on a social network use case
  - Oriented to evaluate essential graph queries
  - Tools: data generator + test-driver + data converter
- Experiments
  - Small datasets: 1K, 100K, 500K, 1M nodes (done)
  - Very large datasets: > 10M nodes (current work)
- Experience on using several graph/RDF databases

AGENDA

1. INTRODUCTION
2. THE BENCHMARK
3. PRELIMINARY EXPERIMENTAL RESULTS
4. CONCLUSIONS

Benchmark Data Model



Social Network Data based on Facebook

Benchmark Query Mix

- Objective: real-life graph queries in a social network
- Approach: exploration of Facebook
- Selection: essential graph queries
- Result:
  - Attribute searching (Get people with a given name)
  - Node/edge adjacency (Get people that likes a given Web page)
  - Fixed-length paths (Get the friends of the friends of a given person)
  - Reachability (Is there a “friend” connection between two people?)
  - Pattern matching (Get the common friends between two people)
  - Aggregates (Get the number of friends of a given person)

Benchmark query mix

- (Q1) Get people having a given name
- (Q2) Get people that likes a given Web page W
- (Q3) Get the Web pages that a given person P likes
- (Q4) Get the name of a person with a given ID
- (Q5) Get the friends of the friends of a given person P
- (Q6) Get Web pages liked by the friends of a given person
- (Q7) Get people that likes a Web page which a person P likes
- (Q8) Is there a “friend” connection (path) between two people
- (Q9) Get the shortest path between two people
- (Q10) Get the common friends between two people
- (Q11) Get the common web pages that two people like
- (Q12) Get the number of friends of a given person P

Expressing graph queries

	Get the friends of a person identified by id 10
Dex	long person_id = dex_graph.findObject(pid, dexvalue.setLong(10)): dex_graph.neighbors(person_id, friend, EdgesDirection.Outgoing);
InfiniteGraph	Person person = this.findPersonById(10); Iterator<VertexHandle> it = person.getNeighbors().iterator(); while (it.hasNext()) { … }
Neo4j	START p=node:peopleIdIndex(id=10) MATCH p-[:friend]->f RETURN f
OrientDB	SELECT FROM ographvertex WHERE in[label='friend'].out in (select rid from index:personIdx where key = 10)
SPARQL	SELECT ?friend WHERE { ?person <http://sn.org/voc/person#id> 10 . ?person <http://sn.org/voc/friend> ?friend }

AGENDA

1. INTRODUCTION
2. THE BENCHMARK
3. PRELIMINARY EXPERIMENTAL RESULTS
4. CURRENT EXPERIMENTS
5. CONCLUSIONS

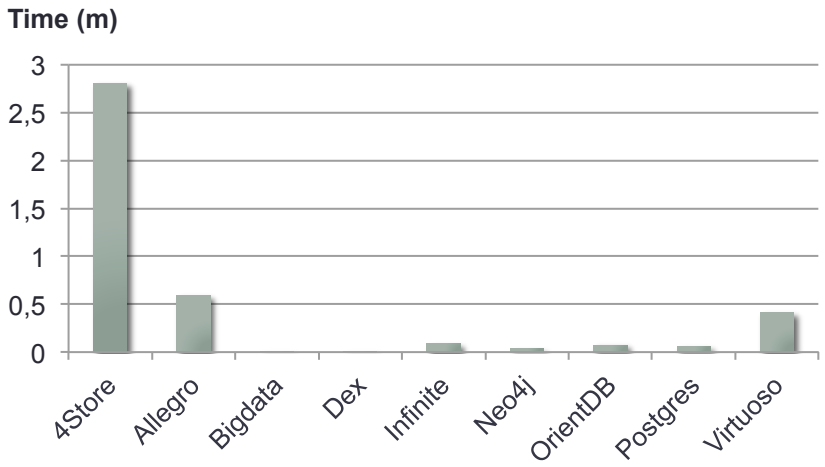
Datasets

Dataset	Nodes	Edges	RDF Triples	Tuples (SQL)
G1	1000	5.874	8695	6.874
G2	100.000	1.002.216	1.283.727	1.102.216
G3	500.000	5.735.332	7.142.423	6.235.332
G4	1.000.000	12.094.498	14.909.745	13.094.498

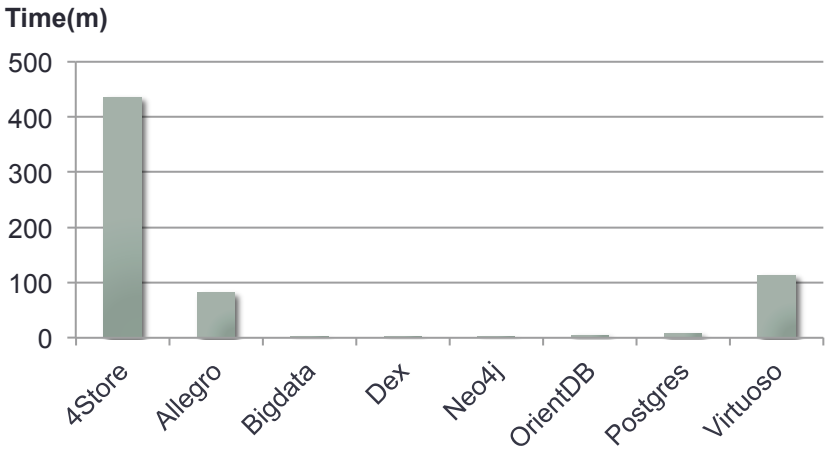
Running on a HP Proliant, Intel Xeon X3430 2.40GHz, 8 GB RAM, Debian 64 bits

Data loading test

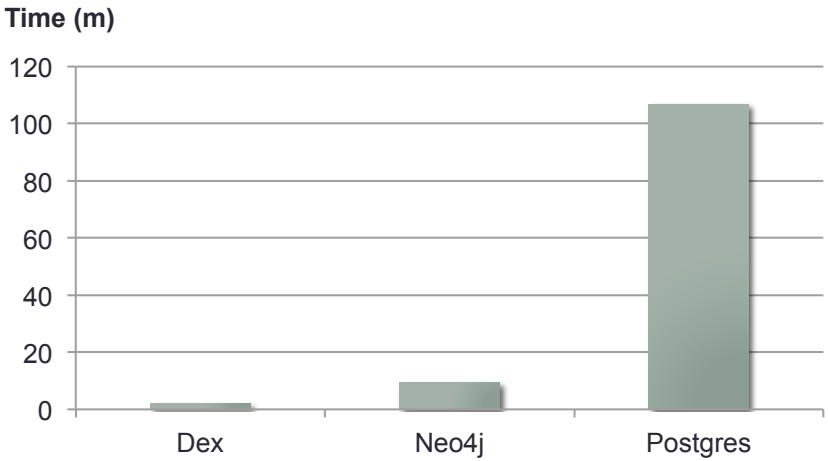
N=1.000 E=5.874



N=100.000 E=1.002.216



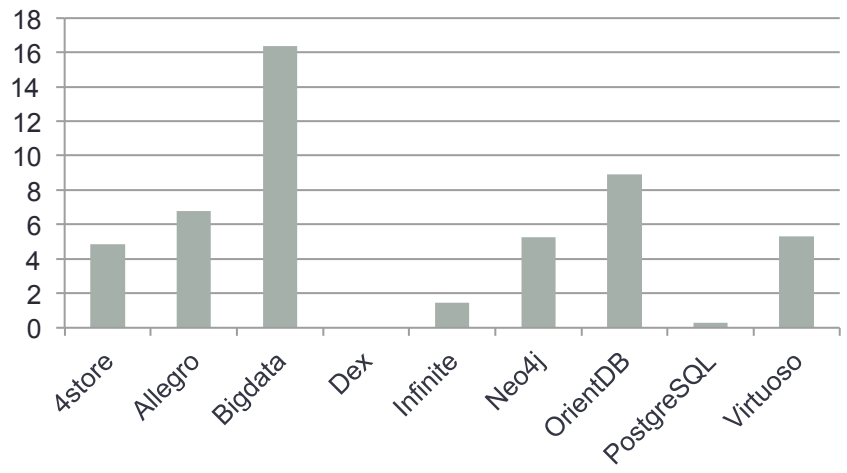
N=500.000 E=5.735.332



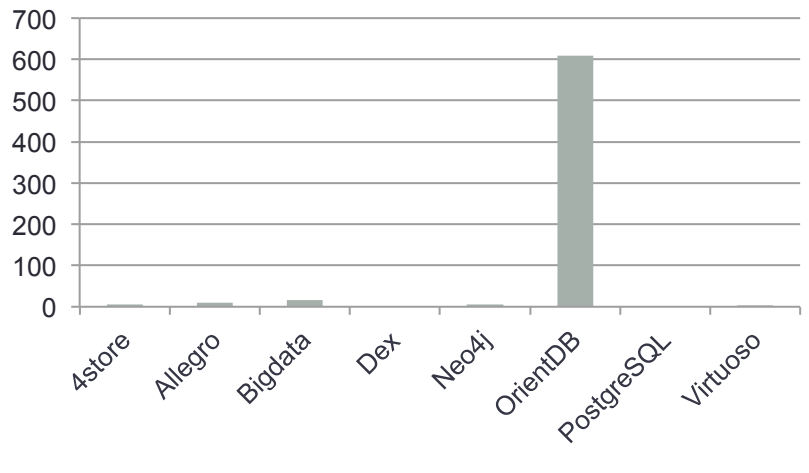
N=1.000.000 E=12.094.498

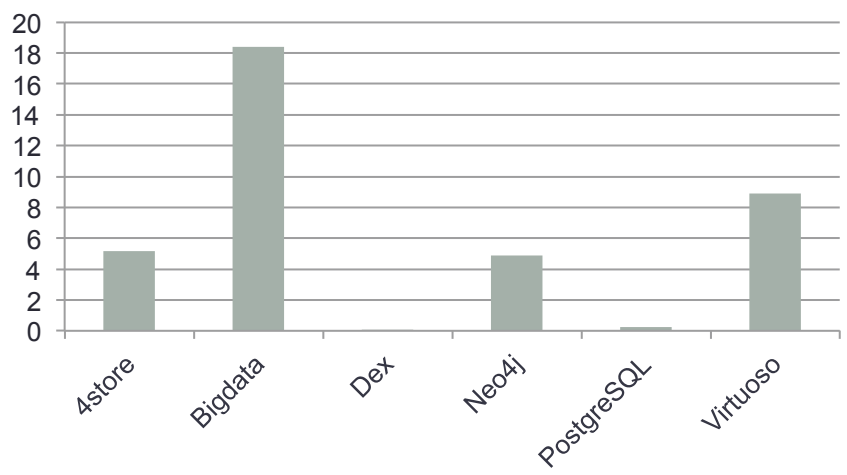
(Q1) Get people having a given name (attribute searching)

N=1.000 E=5.874

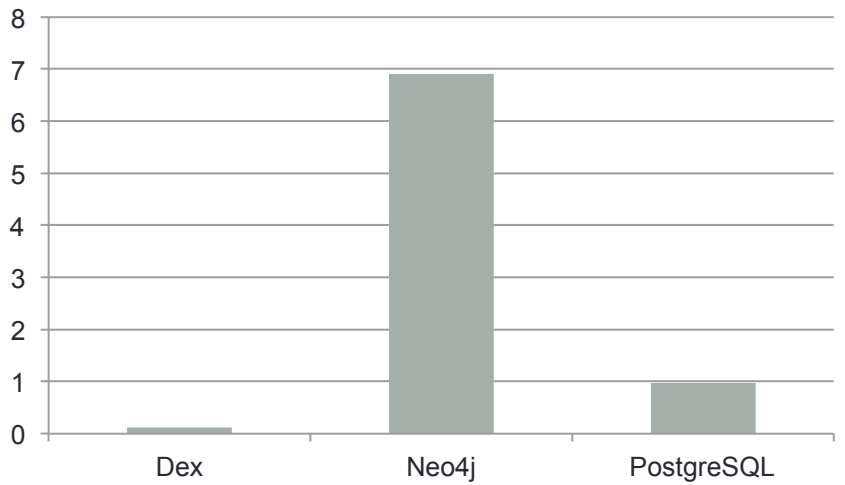


N=100.000 E=1.002.216





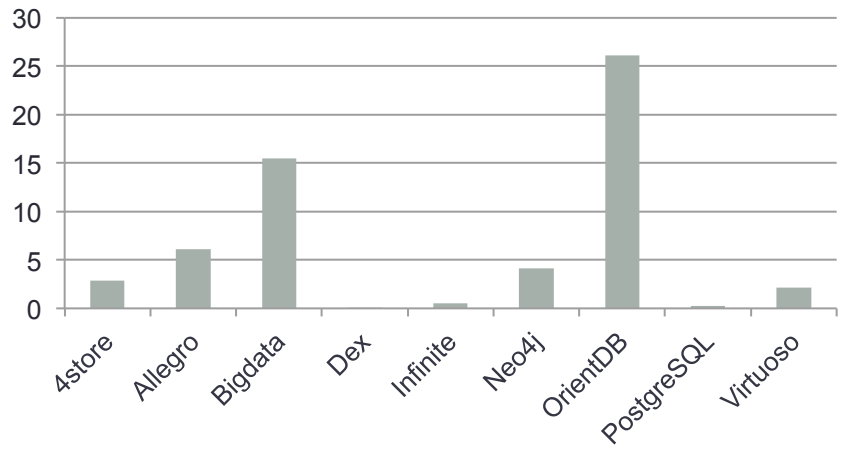
N=500.000 E=5.735.332



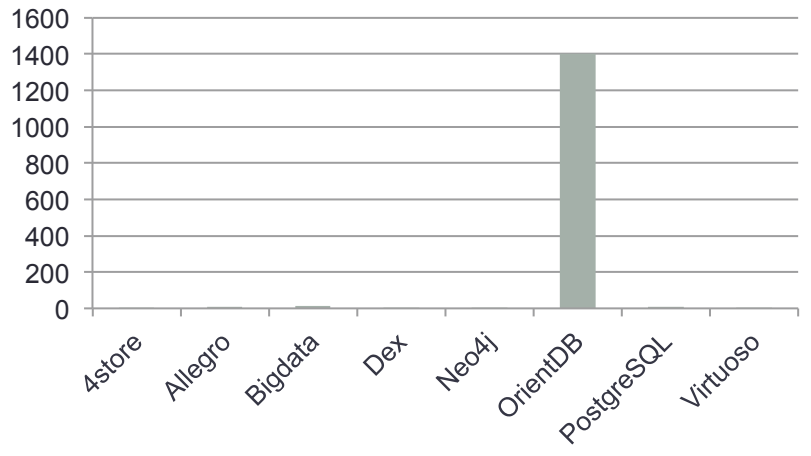
N=1.000.000 E=12.094.498

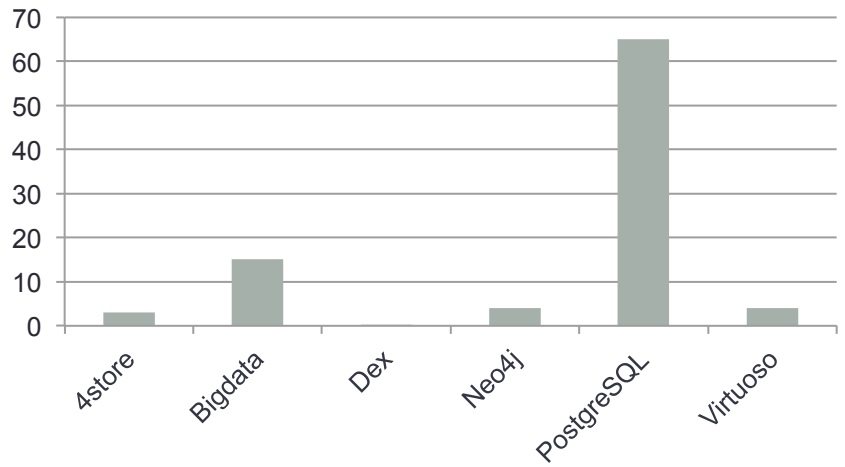
(Q2) Get people that likes a given Web page W

N=1.000 E=5.874

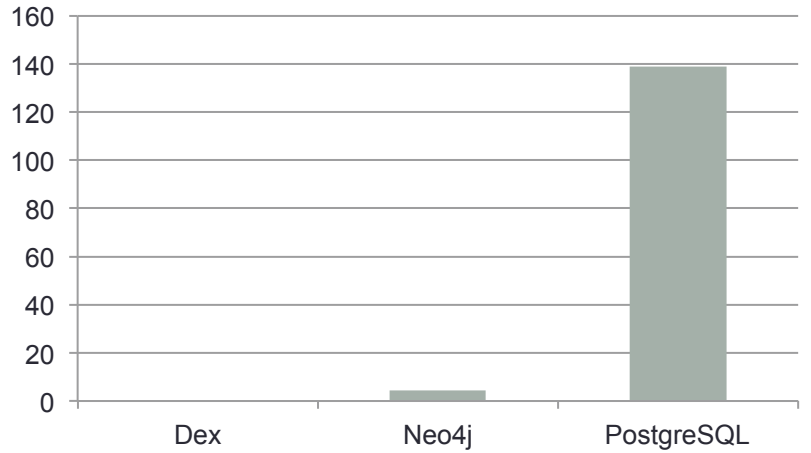


N=100.000 E=1.002.216





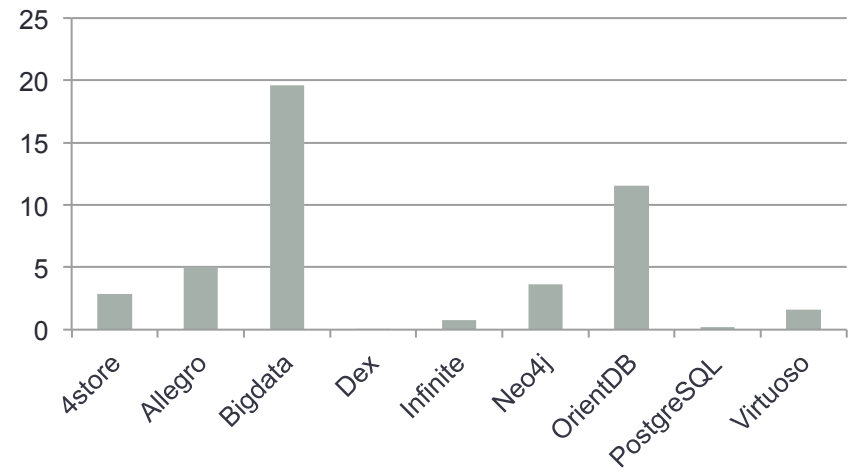
N=500.000 E=5.735.332



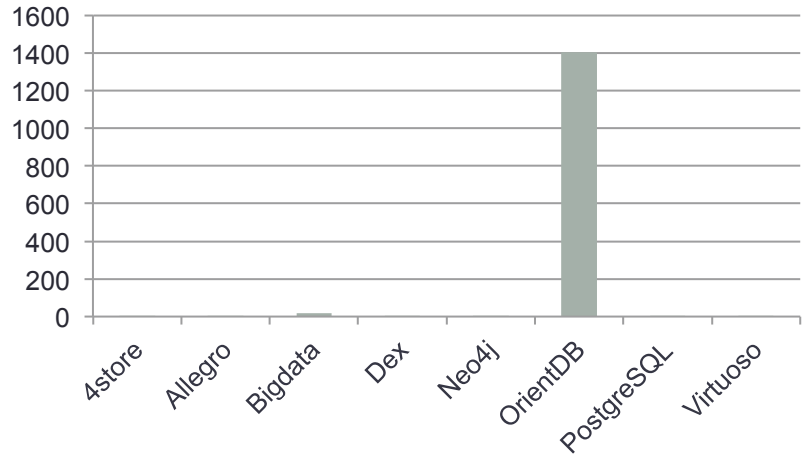
N=1.000.000 E=12.094.498

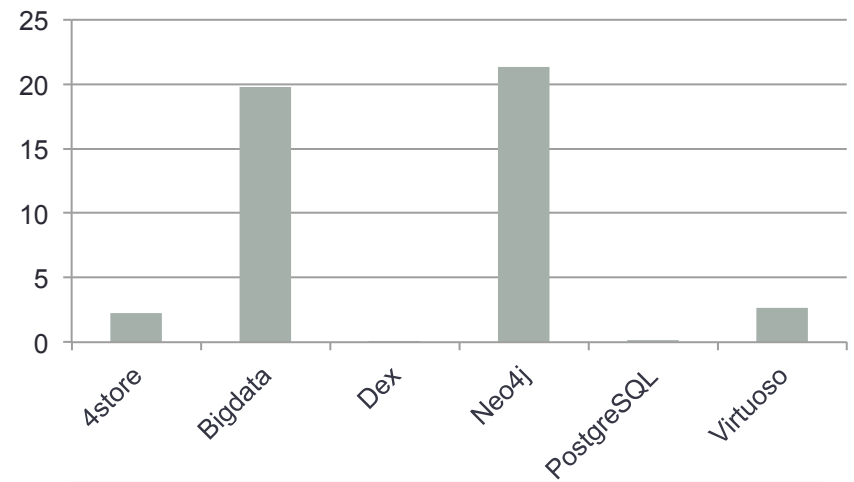
(Q3) Get the Web pages that a given person P likes (adjacency)

N=1.000 E=5.874

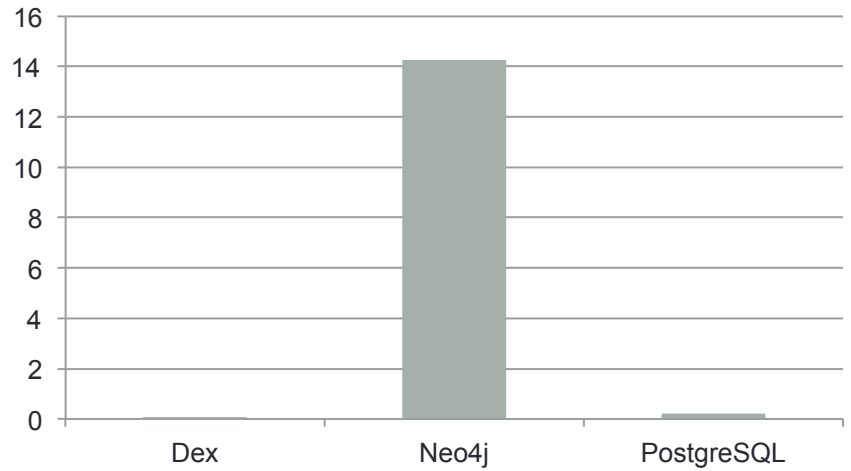


N=100.000 E=1.002.216



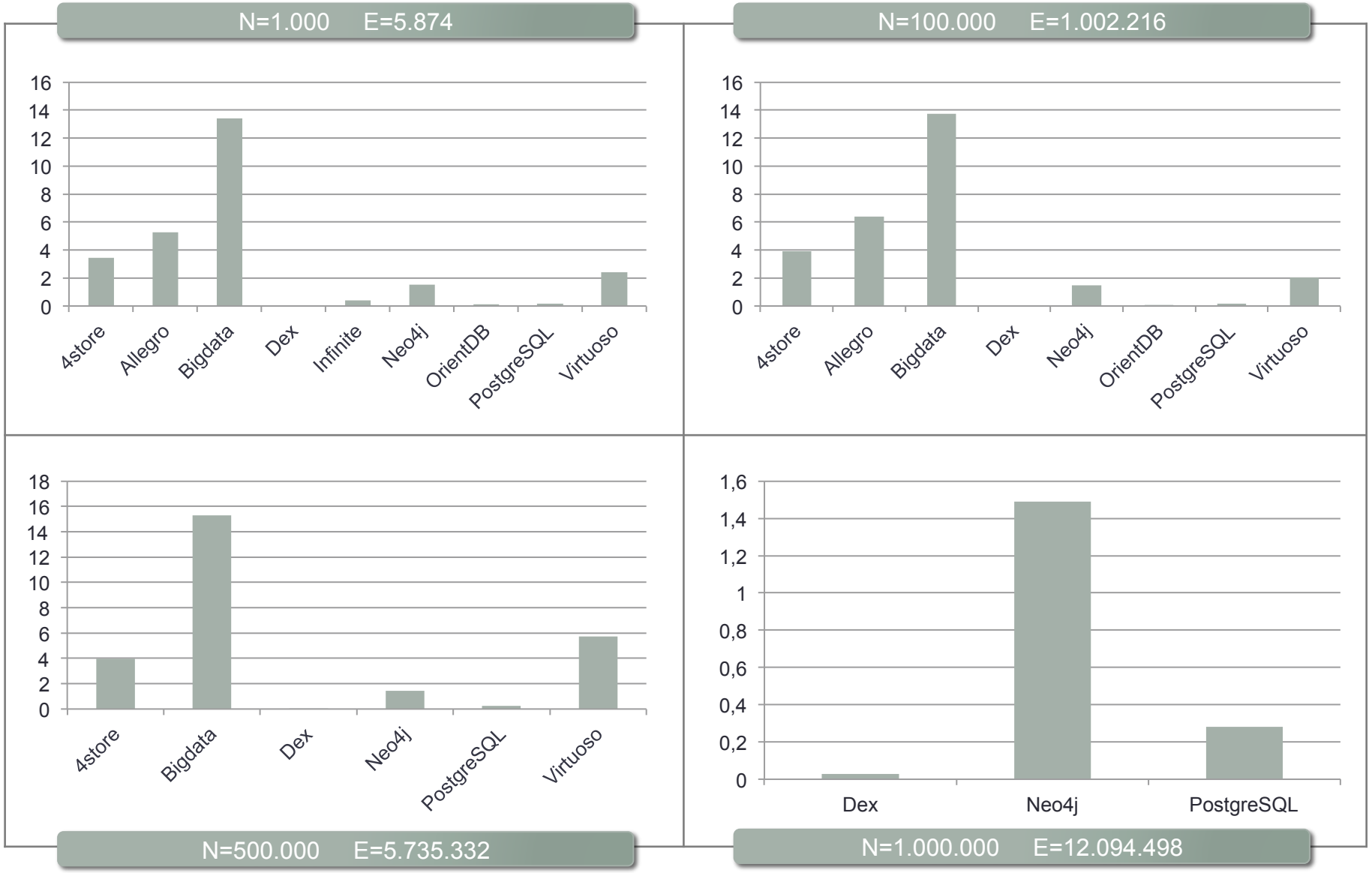


N=500.000 E=5.735.332



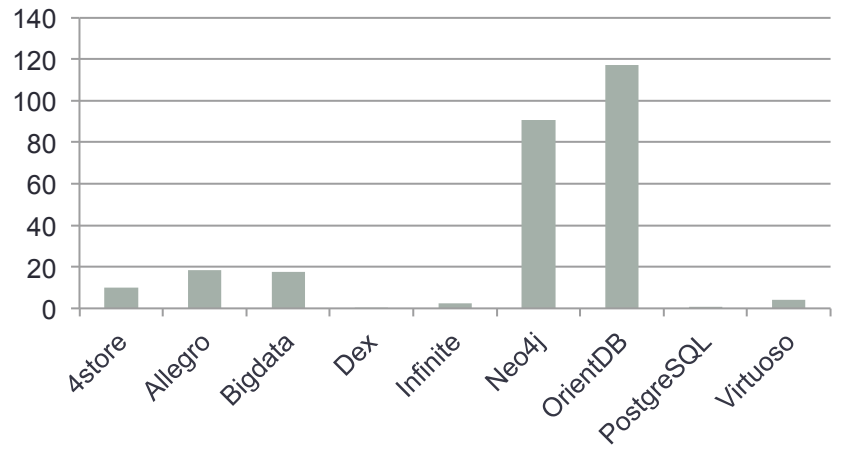
N=1.000.000 E=12.094.498

(Q4) Get the name of a person with a given ID

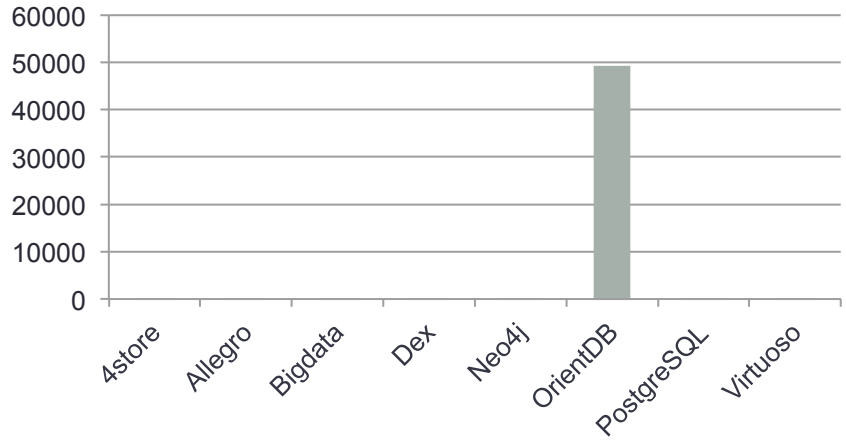


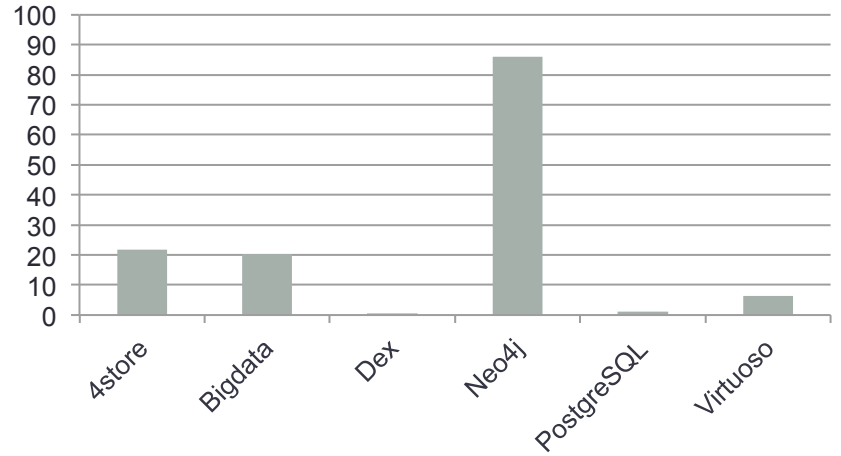
(Q5) Get the friends of the friends of a given person P (path)

N=1.000 E=5.874

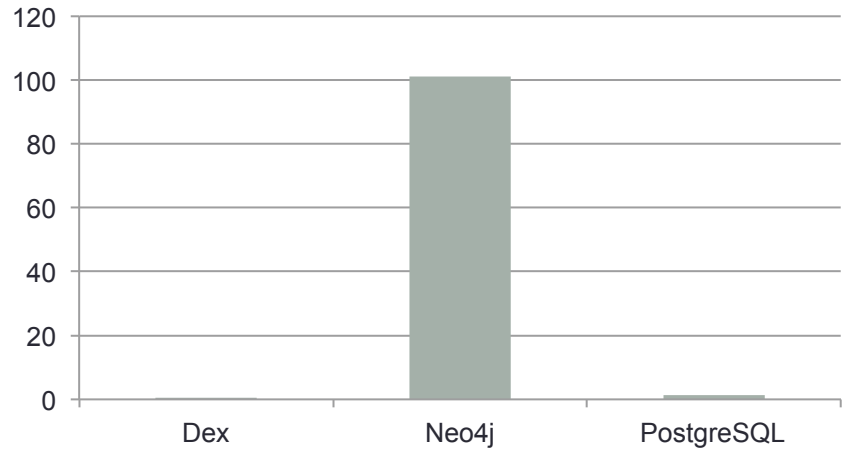


N=100.000 E=1.002.216





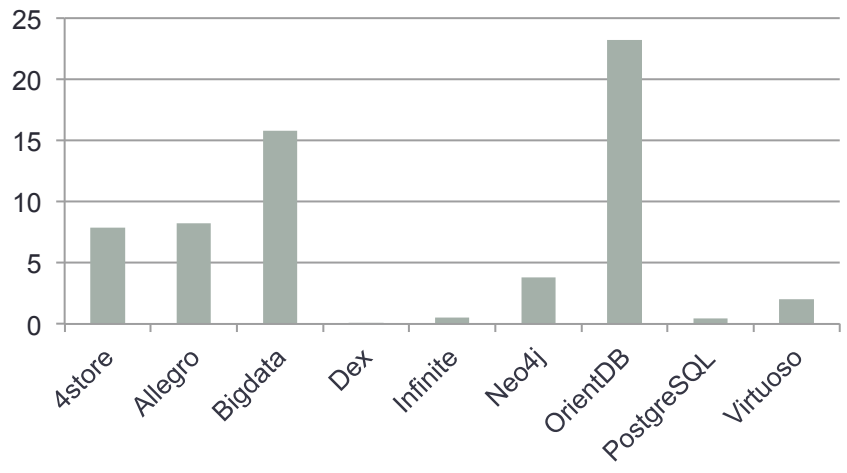
N=500.000 E=5.735.332



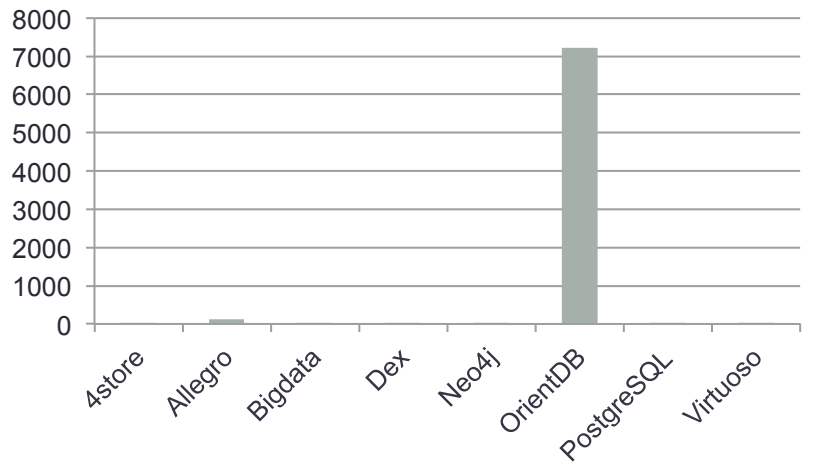
N=1.000.000 E=12.094.498

(Q6) Get the Web pages likes by the friends of a given person

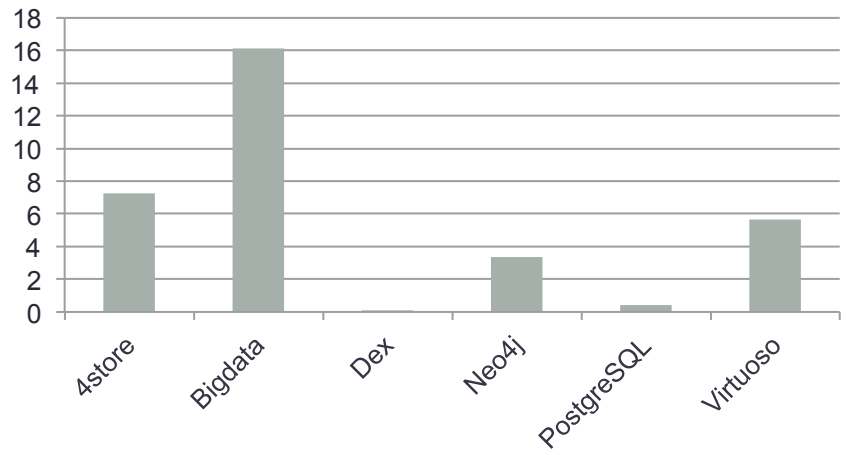
N=1.000 E=5.874



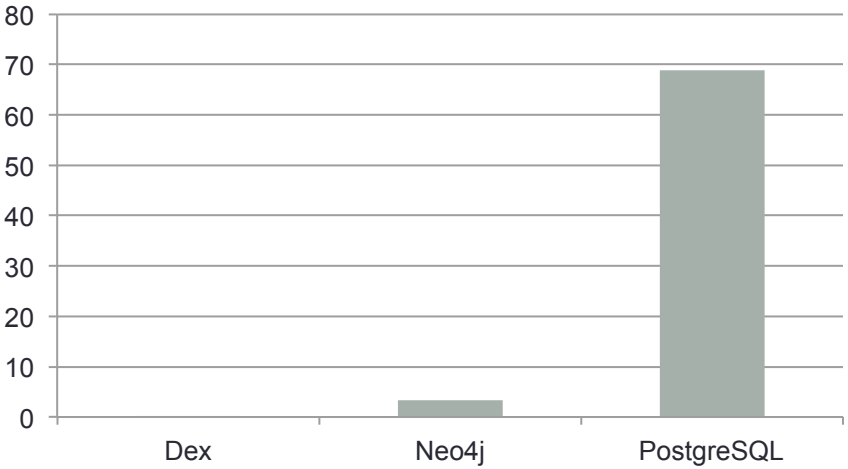
N=100.000 E=1.002.216



N=500.000 E=5.735.332

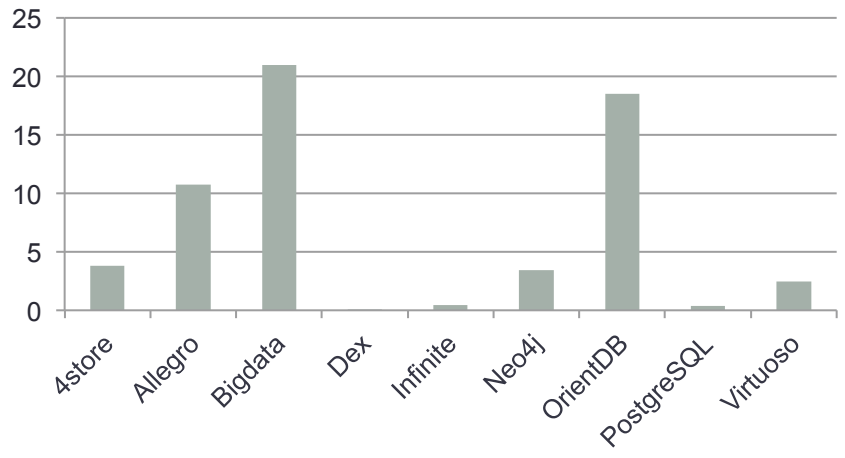


N=1.000.000 E=12.094.498

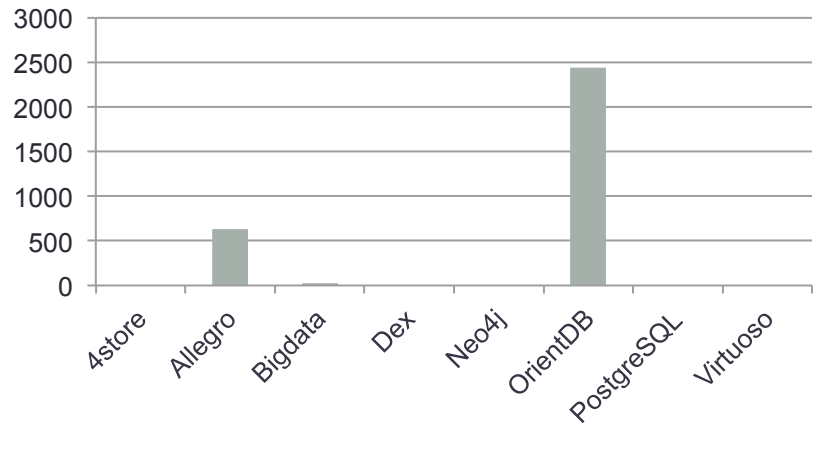


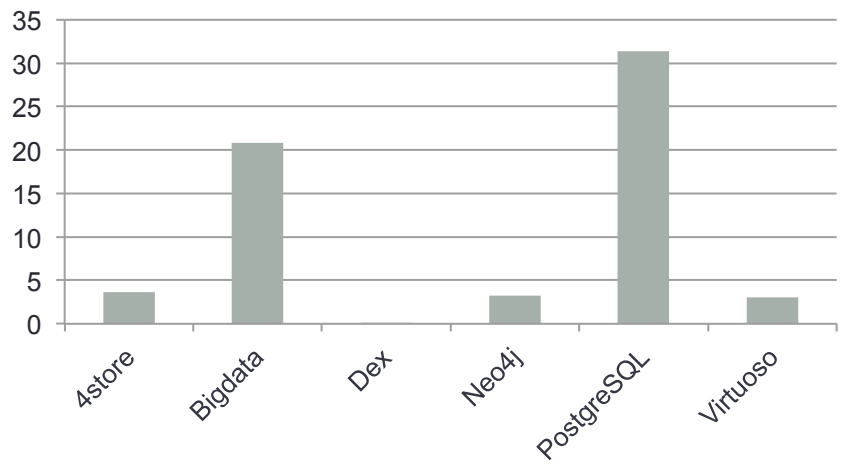
(Q7) Get people that likes a Web page which a person P likes

N=1.000 E=5.874

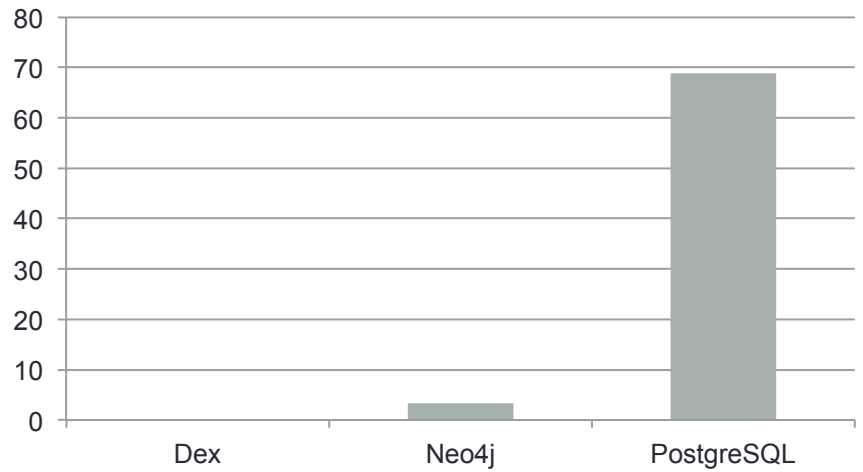


N=100.000 E=1.002.216





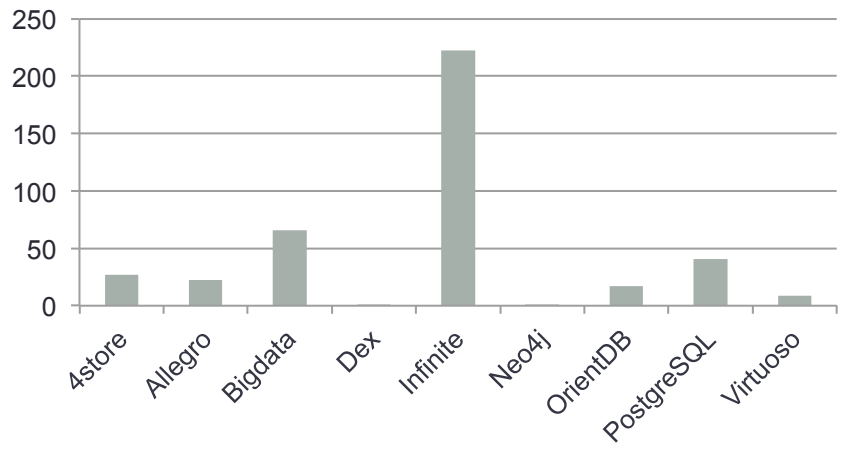
N=500.000 E=5.735.332



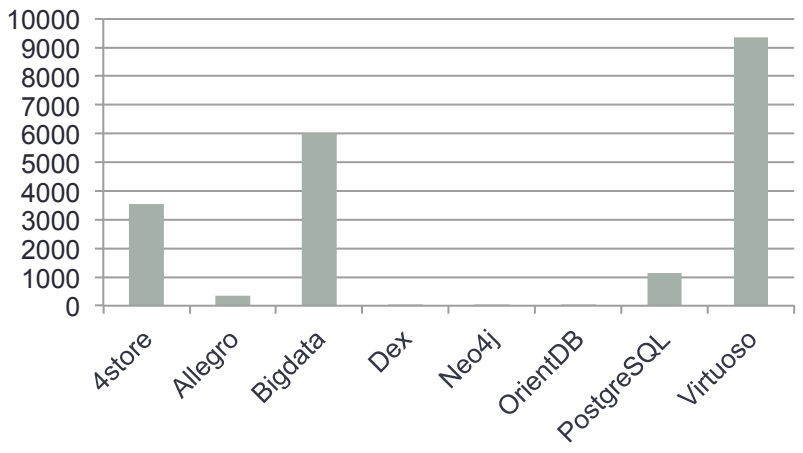
N=1.000.000 E=12.094.498

(Q8) Is there a “friend” connection (path) between two people

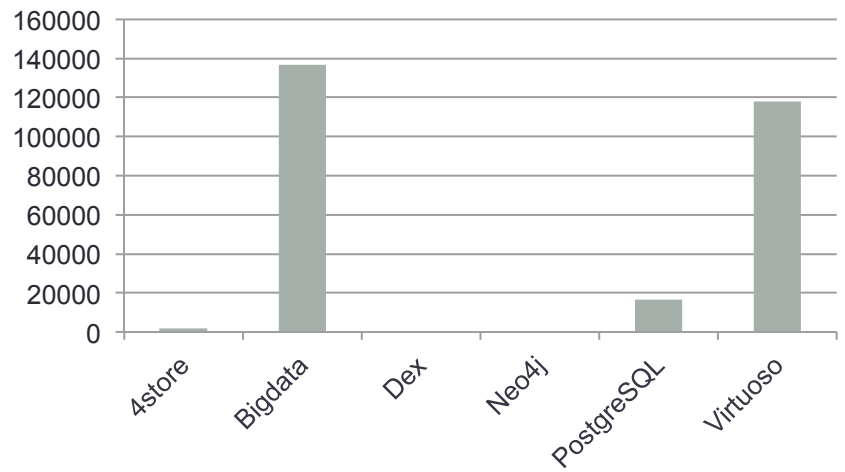
N=1.000 E=5.874



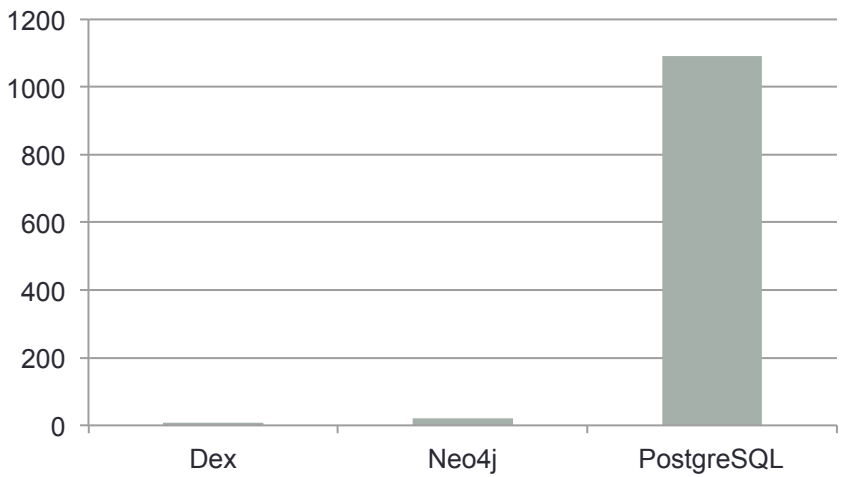
N=100.000 E=1.002.216



N=500.000 E=5.735.332

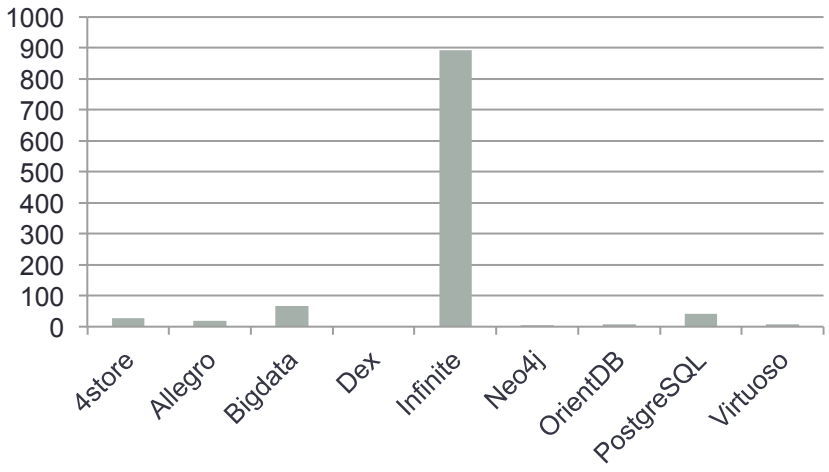


N=1.000.000 E=12.094.498

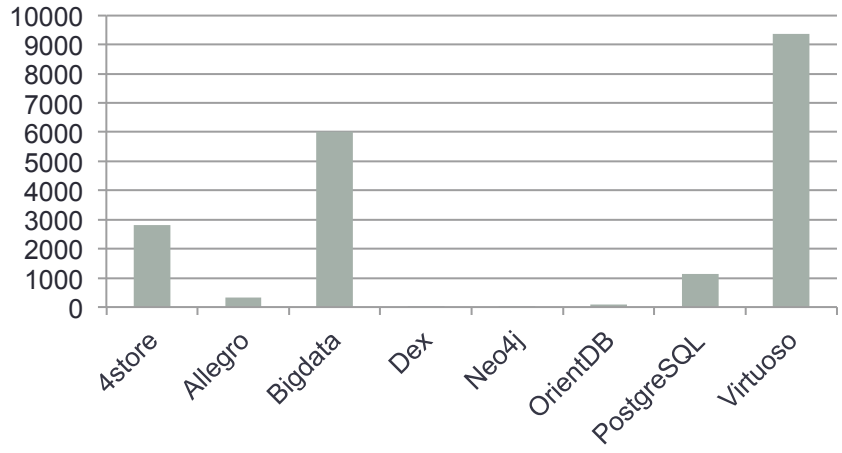


(Q9) Get the shortest path between two people

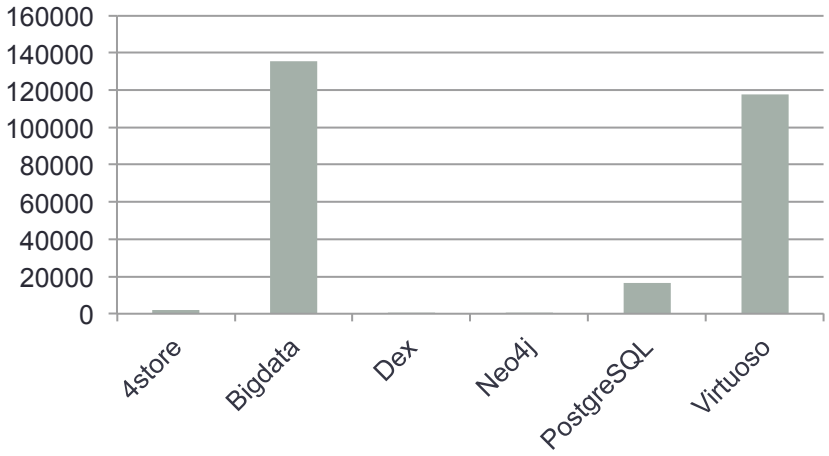
N=1.000 E=5.874



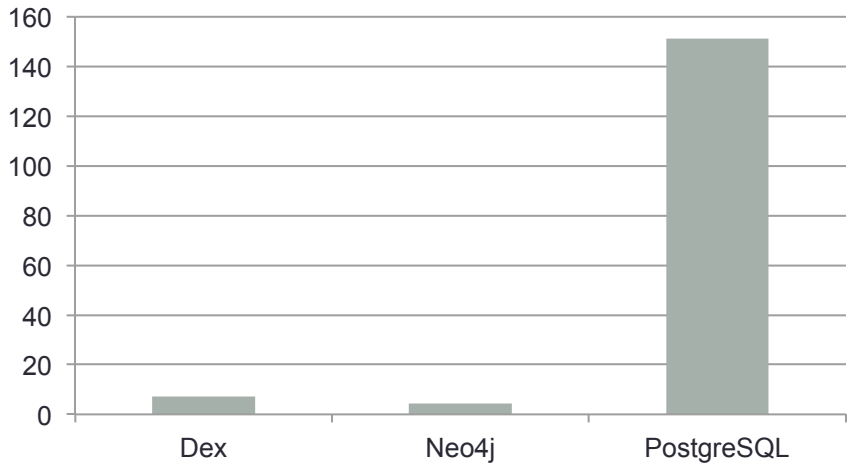
N=100.000 E=1.002.216



N=500.000 E=5.735.332

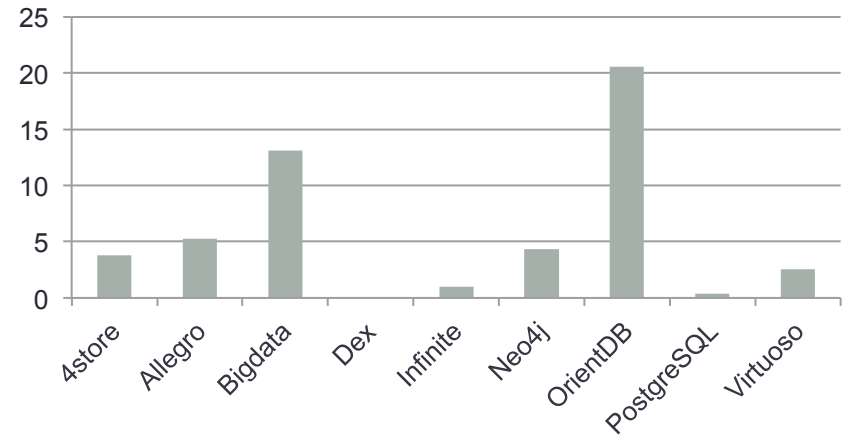


N=1.000.000 E=12.094.498

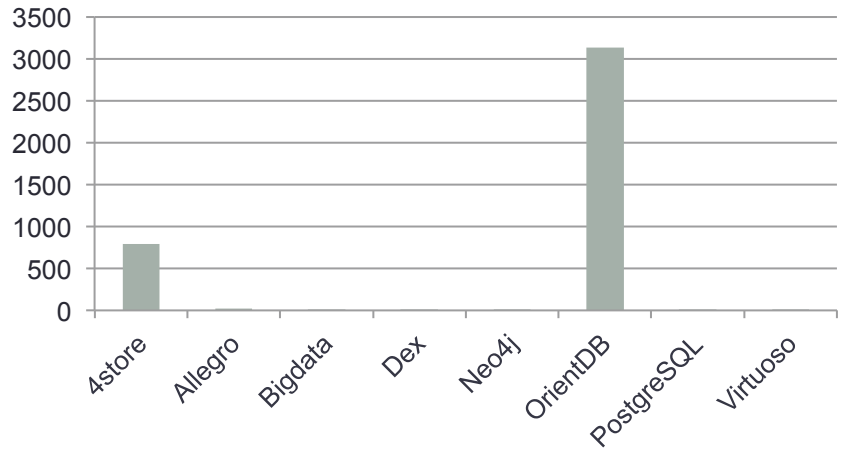


(Q10) Get the common friends between two people (graph pattern)

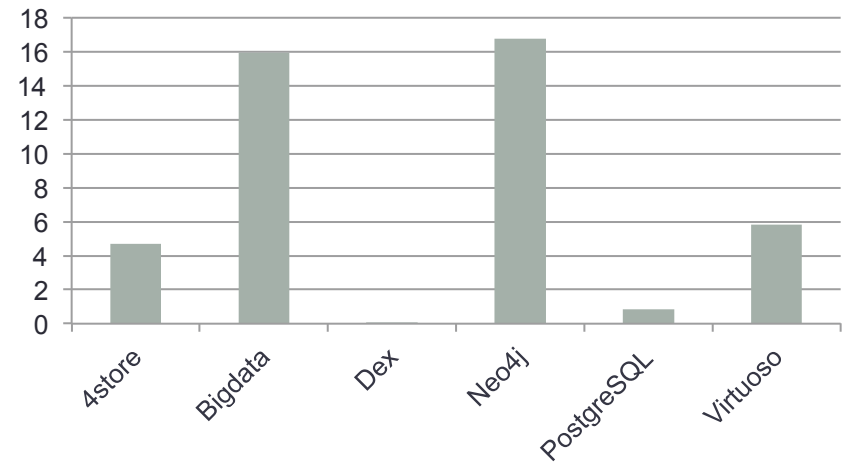
N=1.000 E=5.874



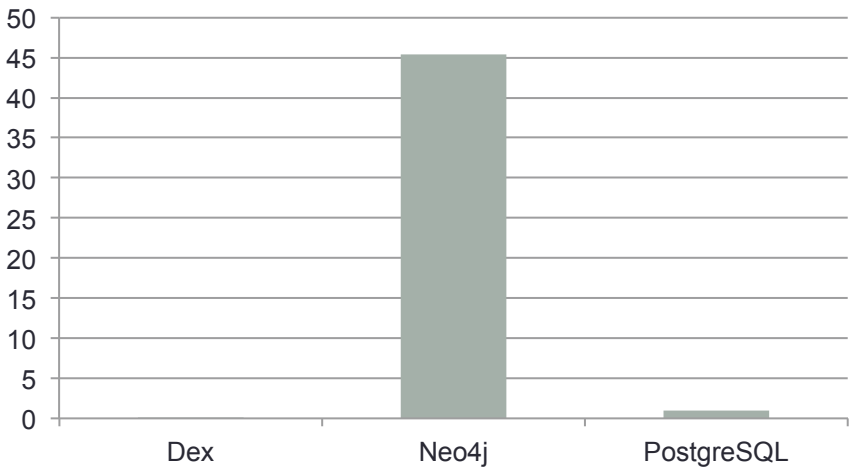
N=100.000 E=1.002.216



N=500.000 E=5.735.332

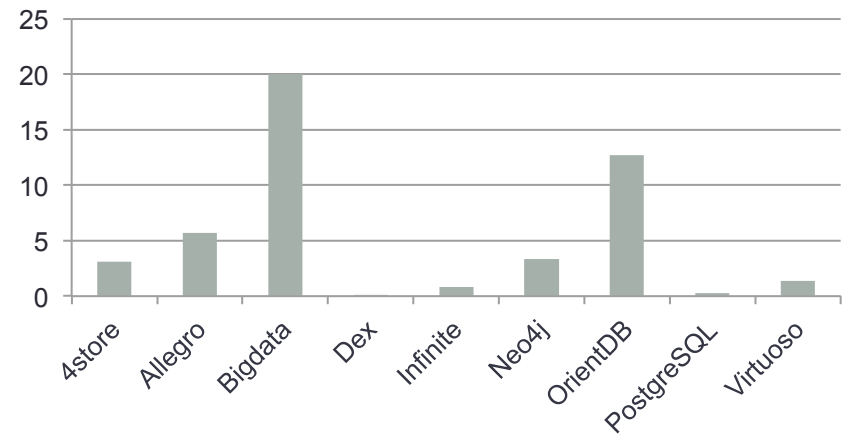


N=1.000.000 E=12.094.498

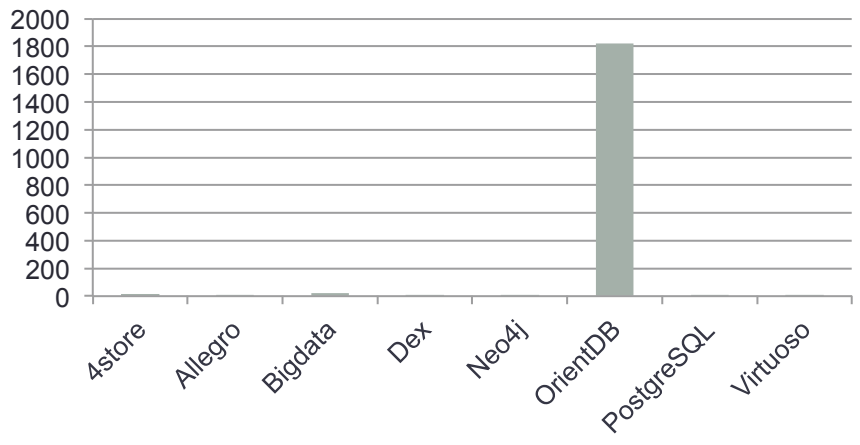


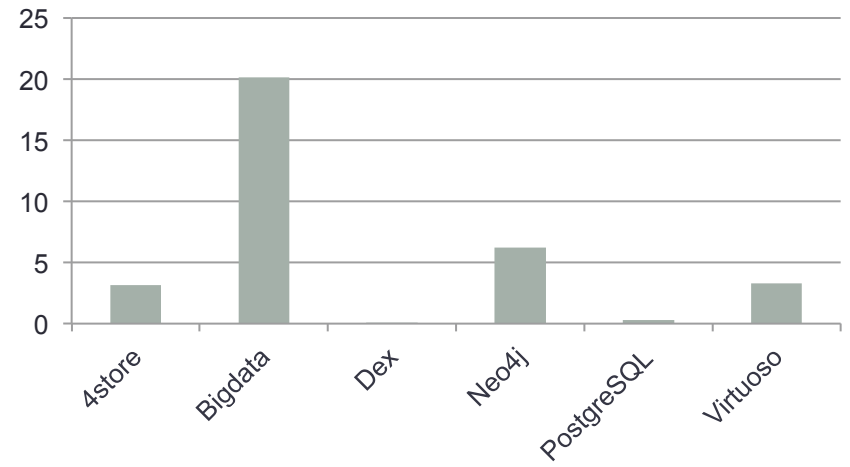
(Q11) Get the common web pages that two people like

N=1.000 E=5.874

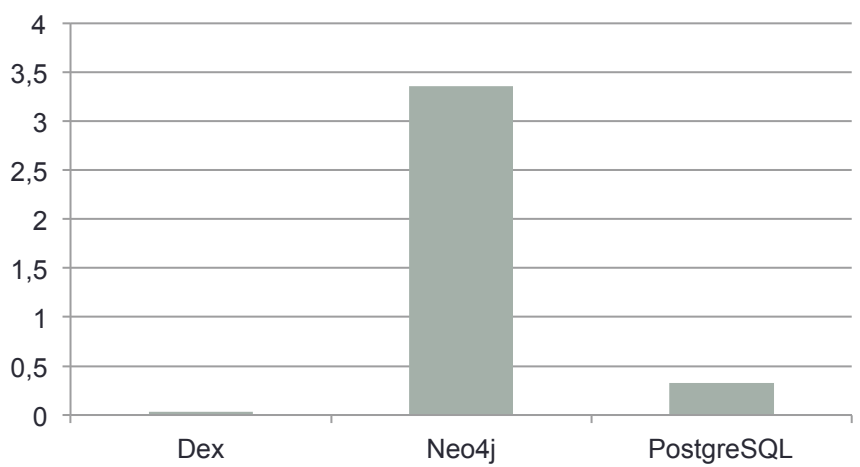


N=100.000 E=1.002.216





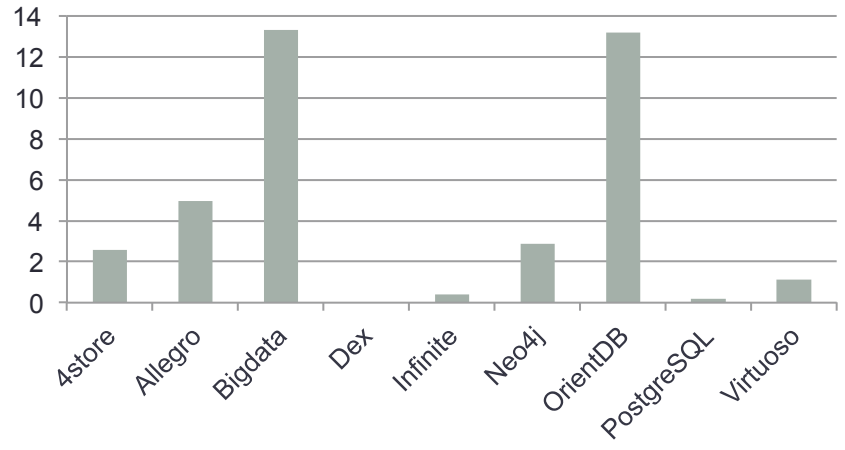
N=500.000 E=5.735.332



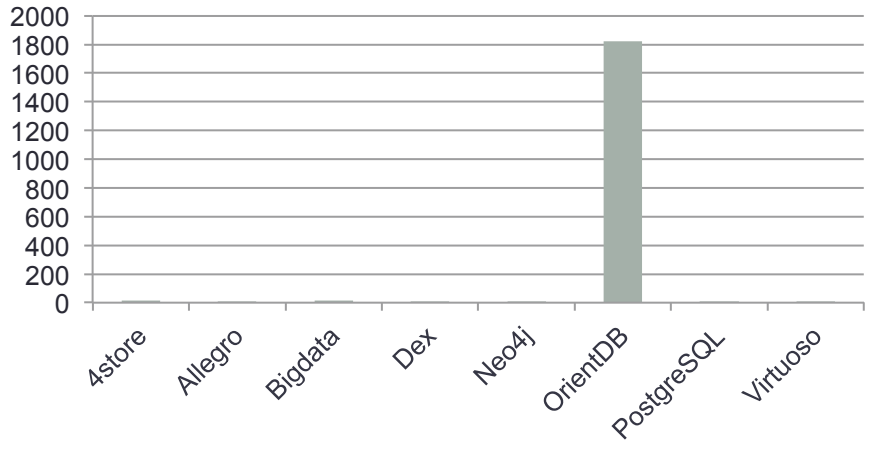
N=1.000.000 E=12.094.498

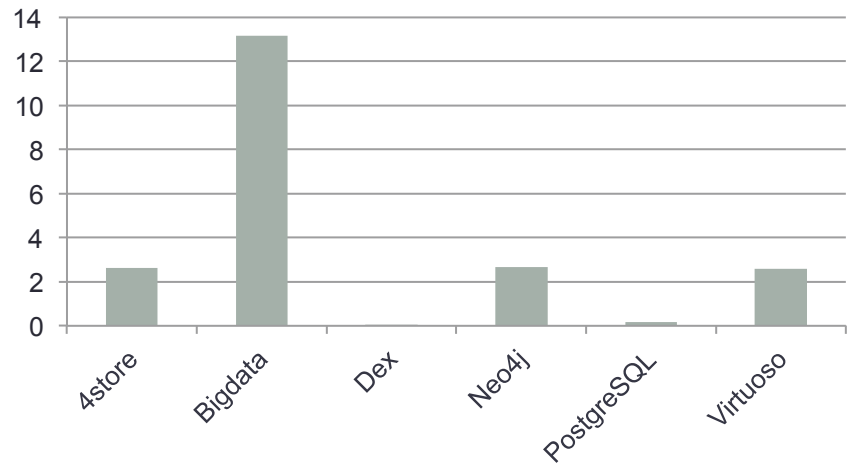
(Q12) Get the number of friends of a given person P (aggregation)

N=1.000 E=5.874

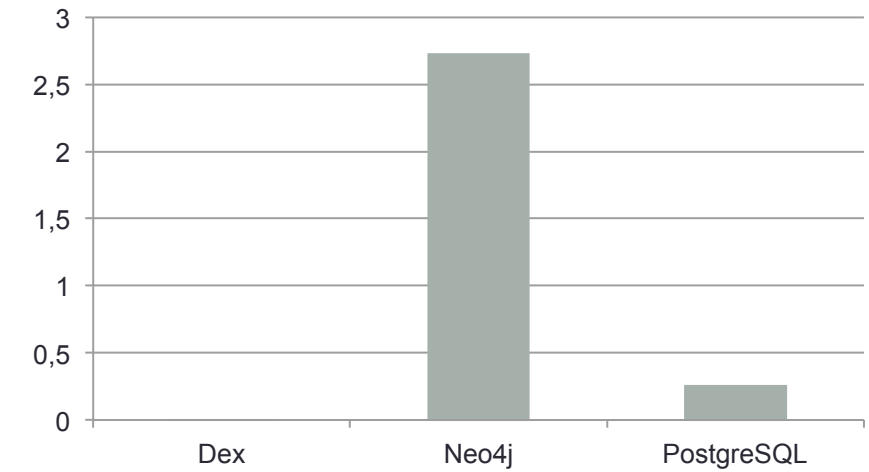


N=100.000 E=1.002.216





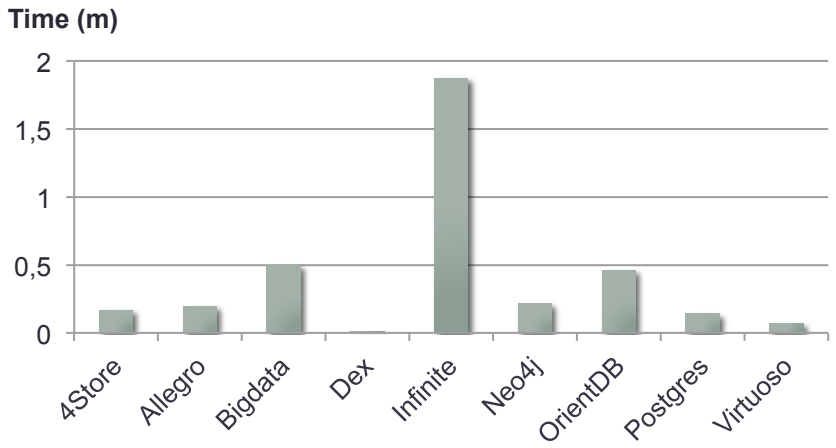
N=500.000 E=5.735.332



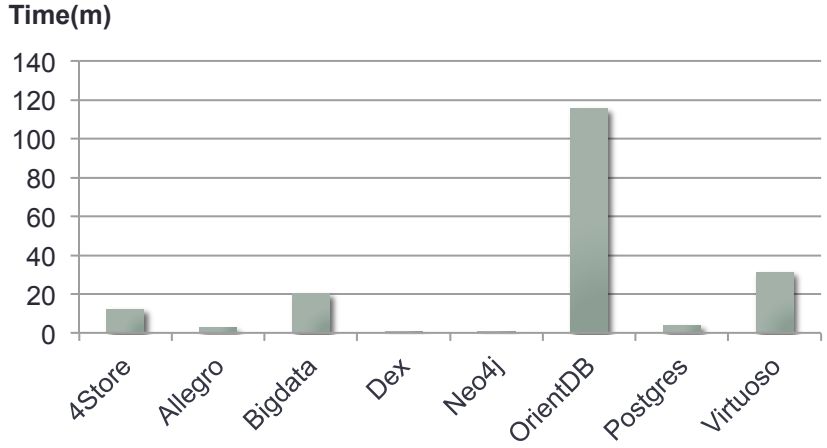
N=1.000.000 E=12.094.498

Total time for query mix (12 queries x 100 instances)

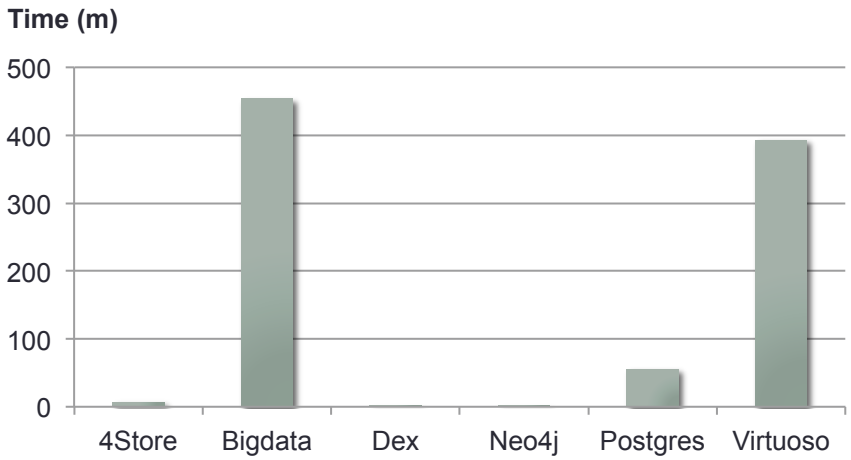
N=1.000 E=5.874



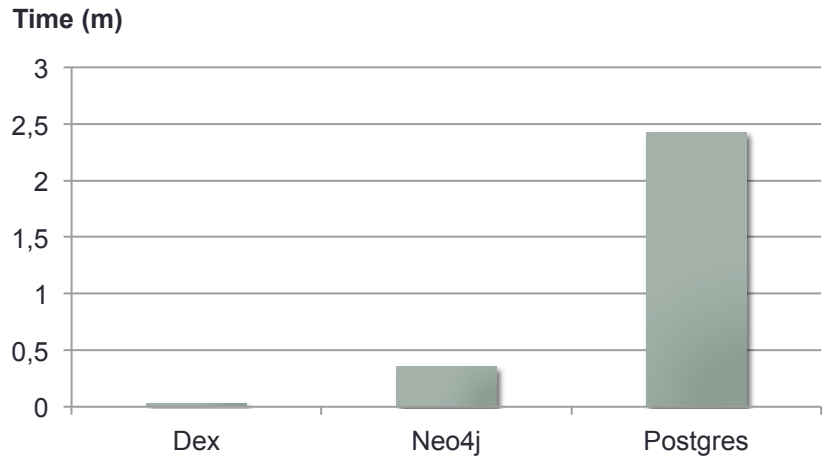
N=100.000 E=1.002.216



N=500.000 E=5.735.332



N=1.000.000 E=12.094.498



AGENDA

1. INTRODUCTION
2. THE BENCHMARK
3. PRELIMINARY EXPERIMENTAL RESULTS
4. CONCLUSIONS

Conclusions

- The comparison of current graph databases is not an easy task
  - There are several graph data models
  - There is not standard graph query language
  - Models and query languages are not formally defined
  - There is not a standard graph data format (to data export/import)

- We developed a benchmark for graph databases
  - Based on a social data network use-case
  - Oriented to evaluate essential graph queries

Conclusions

- Our experience
  - Most graph databases implement APIs for managing graph data
  - Several problems for installing and using the systems
  - Several problems during data loading (e.g., slow, codification)
  - Different query languages and/or query features (APIs)
  - Graph queries are well supported by graph databases
  - Most implementations fail by RAM use

- Current and future work:
  - Improvement of the benchmark
  - Evaluation of very large datasets: 10M, 100M , 1B … nodes
  - Inclusion of other databases (e.g., column-store, key/value, etc.)

Acknowledgments

R. Angles was supported by Fondecyt Project Nr. 11100364, Chile

Design and development of the graph data generator
Sebastián Arancibia, graduated student, Department of Computer Science, Universidad de Talca, Chile

Execution and reporting of benchmarking tests

Sergio Silva, undergraduate student, Department of Computer Science, Universidad de Talca, Chile