**Comparison:**

1. [A Comparison of Current Graph Database Models - 2012](https://www.researchgate.net/publication/261076480_A_Comparison_of_Current_Graph_Database_Models)

* Review including general features, data modeling features, and support for essential graph queries
* Graph databases having the following components were considered:  
  -> external interfaces, database languages, query optimizer, database engine, storage engine, transaction engine, management and operation features (tuning, backup, recovery, etc)
* The evaluation presented is oriented to evaluate the data model provided by each graph database, in terms of data structures, query language and integrity constraints
* Graph stores were also considered
* Comparison done on: AllegroGraph, DEX, Filament, G-Store, HypergraphDB, InfiniteGraph, Neo4J, Sones, VertexDB
* Overall the paper compares only features of different graph databases and graph stores, not a performance evaluation.

1. [Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB - 2018](https://pdfs.semanticscholar.org/bcca/5c7e9180c4890936343d79e812cb16cbefb6.pdf)

* Analysis of Graph Databases: AllegroGraph, ArrangoDB, InfiniteGraph, Neo4J, OrientDB
* Selected important features such as flexible schema, query language, sharding and scalability
* A section on each of the 5 graph databases highlighting their features, advantages and limitations
* Comparison of features include flexible schema, query language, sharding, backups, multimodal, multi-architecture, scalability, cloud ready
* Some features mentioned, but a performance evaluation not done, and left as future work

**Performance Evaluation:**

1. [Scalability and Performance Evaluation of Graph Database Systems: A Comparative Study of Neo4j, JanusGraph, Memgraph, NebulaGraph, and TigerGraph - 2023](https://ieeexplore.ieee.org/document/10391694)

* Neo4j, JanusGraph, MemGraph, NebulaGraph, TigerGraph
* Query response time, data loading time, memory usage
* Scalability assessment to measure database’s ability to handle increasing size
* Linked Database Benchmark Council’s Social Network Benchmark used
* Overview given with database, its architecture, query language, and key features

1. [EVALUATION OF GRAPH DATABASES PERFORMANCE THROUGH INDEXING TECHNIQUES](https://www.academia.edu/16307078/EVALUATION_OF_GRAPH_DATABASES_PERFORMANCE_THROUGH_INDEXING_TECHNIQUES)

* Performance evaluation of Neo4J and OrientDB through indexing techniques
* Twitter social network of 5000 nodes used
* Two scenarios considered, retrieving a node with an index, and without an index
* Response time mainly analyzed
* Somehow took granularity pattern leveraging the fact that most indexing tech use graphs under the hood to create a natural index for the data models

1. [A Performance Evaluation of Open Source Graph Databases - 2014](http://www.stingergraph.com/data/uploads/papers/ppaa2014.pdf)

* 12 open source graph dbs used, four fundamental graph algorithms on a network of up-to 256 million edges used for comparison
* Single Source Shortest Path (SSSP), Shiloach-Vishkin (SV) Connected Components algorithm, Page Rank (PR), edge insertions and deletions.
* R-MAT generator used for data
* Initial graph construction time, total memory used, update rate in edges per second, time to completion for shortest path, SV, PR taken

1. [Experimental Evaluation of Graph Databases: JanusGraph, Nebula Graph, Neo4j, and TigerGraph - 2023](https://www.mdpi.com/2076-3417/13/9/5770)

* Comparison done on JanusGraph, Nebula Graph, Neo4j, and TigerGraph
* Performance evaluated using Linked Data Benchmark Council’s Social Network Benchmark (LDBC SNB)
* Analysis on execution time of queries, loading time of nodes, RAM and CPU usage for each db
* Details, characteristics, features, limitations of each db mentioned
* LDBC SNB a good benchmark, graphs represent natural phenomena between social interactions, designed to evaluate performance based on real world scenarios
* Schema provided
* Datasets of varying sizes used and compared on
* Execution of benchmark as follows:
  + Installation of graph db
  + Loading dataset and analyzing loading time
  + Running 29 benchmark queries, extracting execution time, CPU and RAM used
  + Running 29 queries 5 times each to check how engines handle the caching effect
* For future work, they aim to evaluate other open-source graph dbs using the LDBC SNB, and use higher scale factors

1. [Experimental Comparison of Graph Databases - 2013](https://www.academia.edu/10845356/Experimental_Comparison_of_Graph_Databases)

* Research on possibilities and limitations of gdbs and conducting an experimental comparison of selected gdbs
* BlueBench used as benchmarking tool
* DEX, InfiniteGraph, OrientDB, Neo4J, Titan
* Graph loading, traversal, shortest path, non-traversing queries (find edges by property and find vertices by property), data manipulation queries

**Benchmarking:**

1. [Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark - 2010](https://www.ac.upc.edu/app/research-reports/html/RR/2010/45.pdf)

* Four graph dbs compared: Neo4j, Jena, HypergraphDB, DEX
* Implement the full HPC Scalable Graph Analysis Benchmark (HPC-SGAB) (developed in 2009 by another paper)
* Present the first full implementation of HPC-SGAB
* DEX and Neo4J most efficient current GDB implementations
* Benchmarks discussed:

1. Data Generation: performance of operations on directed and weighted graphs. Data generated using R-MAT algorithm (able to build graphs of any particular size and edge density). Data was only generated using R-MAT, and not timed or considered, rather the time taken for generated data to be loaded was considered. The general format of data was: <Start Vertex, End Vertex, Weight>
2. Kernel Description: a kernel is an operation whose performance is tested. There are 4 kernels, first kernel loads the data, other kernels use the image to compute queries.

* Benchmarks implemented using the Java interface of each DB, with the latest versions
* DB size also considered, for the above benchmarks, time taken was considered

1. [Empirical Comparison of Graph Databases - 2014](https://www.odbms.org/wp-content/uploads/2014/05/an-empirical-comparison-of-graph-databases.pdf)

* Distributed graph database comparison framework for 4 major graph databases: Neo4j, Orient, Titan, DEX
* Tinkerpop developed *Blueprints*, a Java API for graph databases.
* Using *Blueprints*, they develop GDB (Graph Database Benchmark) for comparison: user defines a benchmark containing a list of databases to compare followed by a series of operations called workloads to realize on each database
* Workloads:
  + Load workload: start a graph db, load it with a dataset, loading time measured every 10000 graph elements loaded in db
  + Traversal workload: perform a particular traversal on the graph, shortest path or neighborhood exploration
  + Intensive workload, parallel clients synchronized to send a specific number of basic requests: GET by ID, GET by property, GET and UPDATE, GET and ADD an edge
* Dataset available here: [Dataset](https://graphbenchmark.com/)
* GDB available here: [https://github.com/kuzeko/graph-databases-testsuite](https://github.com/kuzeko/graph-databases-testsuite/tree/V2)
* Concerning loading times, DEX performed better and OrientDB the worst
* Neo4J performed best on traversal workloads and Titan Cassandra the worst, regardless of the number of hops. They used mean, median, quartile ranges and outliers for this assessment.
* Considering read-only intensive workloads, Neo4j, DEX, Titan-BDB and OrientDB had a similar performance.
* For read-write intensive workloads, Neo4j Titan-BDB and OrientDB’s performance degrades sharply, while DEX and Titan-Cassandra are better.
* GDB, a new tool was developed for the purpose of performance testing with the above benchmarks.

1. [Benchmarking Traversal Operations over Graph Databases - 2015](https://sci-hub.st/10.1109/icdew.2012.47)

* Discuss need to compare diff dbs, challenges of developing fair benchmarking methodologies
* Focus on benchmarking of traversal operations, design of graph traversal benchmark along with results.
* Efficient support for traversal operations, measure capabilities like query traversals (eg BFS), graph analysis operations (connected components, centrality measures, and community detection)
* Aim at performance in memory, using data of varying sizes
* Blueprints adopted as the interface their graph databases
* Used network generators for data, LFR-Benchmark generator
* DBs compared: Neo4J, DEX, OrientDB, Native RDF repository, SGDB
* Load Times, computation time of BFS, computation time of connected components using in-out edges done

**Other Literature:**

1. [Analyzing The Encountered Problems and Possible Solutions of Converting Relational Databases to Graph Databases - 2022](https://dergipark.org.tr/tr/download/article-file/1766229)
2. [MillenniumDB: A Persistent, Open-Source, Graph Database](https://arxiv.org/pdf/2111.01540)
3. [A Comparison of Graph Database and a Relational Database](https://www.researchgate.net/publication/220996559_A_comparison_of_a_graph_database_and_a_relational_database_A_data_provenance_perspective)
4. [Querying RDF Data from a Graph Database Perspective](https://link.springer.com/chapter/10.1007/11431053_24)