Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 41~48
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp41-48

Comparing MongoDB spatial indexes for proximity queries

Mahmoud Hammadi, Charef Abdallah Bensalloua
CSTL Lab, University of Mostaganem-Abdelhamid Ibn Badis, Mostaganem, Algeria

Article Info

Article history:

Received Apr 29, 2026
Revised Aug 27, 2026
Accepted Sep 10, 2026

Keywords:

Comparative evaluation
MongoDB
Performance
Proximity queries
Spatial indexing

ABSTRACT

In this paper, we empirically compare MongoDB's three main geographical indexes: 2d, 2dsphere, and Geohash. We look at different data volumes and geographic distributions. Our goal is to see how each index behaves in practice. We assess the effects of spatial density (dense urban, suburban, and rural regions) on index reaction time, memory use, and construction time using a repeatable technique based on real-world OpenStreetMap datasets. The findings demonstrate that while Geohash provides useful flexibility in less populated regions, the 2dsphere index is best suited for dense metropolitan environments because of its geometric precision. For historical applications that span tiny, flat regions, the 2d index is still useful. Along with the code and data to reproduce the test, we offer useful suggestions for architects and developers.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Mahmoud Hammadi
CSTL Lab, University of Mostaganem-Abdelhamid Ibn Badis
Mostaganem, Algeria
Email: mahmoud.hammadi.etu@univ-mosta.dz

1. INTRODUCTION

The proliferation of location-based services in logistics, internet of things (IoT), social networks, and mobile applications has created an unprecedented demand for efficient geospatial data management [1], [2]. Modern applications increasingly rely on proximity queries-operations that find nearby points of interest within specified geographic regions-making spatial indexing a critical performance determinant in database systems [3], [4].

MongoDB, a widely adopted NoSQL document-oriented database, provides three primary spatial indexing mechanisms: 2d (planar grid), 2dsphere (spherical geometry), and Geohash (hierarchical encoding). Each index offers distinct theoretical advantages and limitations, yet practical guidance for index selection under varying geographic conditions remains limited [4]. Existing recommendations are predominantly based on theoretical properties rather than empirical validation using real-world datasets and realistic query workloads [5], [6].

Previous benchmark frameworks, such as GeoYCSB [7], have contributed valuable methodologies for evaluating geospatial NoSQL databases. However, these studies typically focus on comparing different database systems rather than providing granular, scenario-specific guidance for index selection within a single database platform. Furthermore, prior MongDB performance evaluations [8], [9] often rely on symmetric datasets or small-scale experiments that do not capture the diversity of real-world geographic distribution, from dense urban centers to sparsely populated rural regions.

The research gap addressed in this paper is the absence of systematic, empirical guidance for selecting among MongDB's spatial indexes based on geographic context. While theoretical comparisons exist, practitioners lack evidence-based recommendations that account for spatial density, query type, and performance trade-offs, including response time, memory consumption, and index build cost.

The main contributions of this work are:

a. A reproducible experimental methodology for comparing MongoDB spatial indexes using real OpenStreetMap datasets across three distinct geographic scenarios (dense urban, suburban, and rural).
b. Empirical performance measurements identifying the optimal index for each scenario-query combination, with quantified improvement percentages.
c. Practical, evidence-based recommendations for developers and architects selecting spatial indexes in MongoDB deployments.
d. An openly available experimental framework and dataset specifications enabling replication and extension by the research community.

The remainder of the paper is organized as follows: Section 2 details the experimental methodology, including the hardware and software environment, dataset characteristics, query types, evaluation metrics, and the reproducible experimental protocol employed. Section 3 presents the empirical results, covering query performance measurements, index storage costs, build times, and comparative analysis across all scenarios, followed by a discussion of observed performance patterns and practical recommendations for practitioners. Finally, Section 4 concludes the paper with a summary of key findings, acknowledges the limitations of the current study, and outlines promising directions for future research in spatial indexing for NoSQL databases.

2. METHOD

2.1. Experimental environment

The experiments were conducted on a dedicated server with the following specifications:

- Processor: Intel Xeon E5-2680 v4 (2.40 GHz, 14 cores).
- Memory: 64 GB DDR4 RAM.
- Storage: 1 TB NVMs SSD (read speed 3500 MB/s).
- Operating System: Ubuntu 22.04.4 LTS.
- Database: MongoDB 7.0.8 (standalone deployement).
- Database configuration: WiredTiger storage engine, cache size 24 GB, default read/write concerns.
- Client: Python 3.10 with PyMongo 4.6, pandas 2.2, matplotlib 3.8, seaborn 0.13.
- Data Extraction: OSMnx 1.9 and Overpass API [10].

2.2. Datasets

Three representative datasets were extracted from OpenStreetMap [11] to capture distinct geographic distributions:

a. Dense urban: 50,000 points of interest (POIs) within Paris, France (48.8566° N, 2.3522° E) representing high spatial density with heterogeneous POI types including restaurants, shops, and public facilities.
b. Suburban: 100.000 POIs across the Île-de-France region, encompassing Paris and its surrounding suburbs, representing medium spatial density with mixed land use.
c. Dispersed rural: 30.000 POIs in the Centre-Val de Loire region, representing low spatial density with dispersed settlements and agricultural land use.

OpenStreetMap was selected as the data source due to its global coverage, proven data quality, and extensive use in scholarly research [12], [13]. For each dataset, POI coordinates were extracted and stored in MongoDB as GeoJSON Point object wth the following structure: {type: “Point”, coordinates: [longitude, latitude]}.

2.3. Tested queries

Three querY types were defined to represent common geospatial query patterns in real-world applications [14]-[16].

- Q1 (nearest neighbor): Find the 10 nearest neighbours ($near) around a randomly selected query point.
- Q2 (radius search): Find all points within a 1 km radius ($geoWithin + $centerSphere).
- Q3 (polygon query): Find all points within a polygon representing a district or municipality boundary.

2.4. Evaluation metrics

Performance was evaluated using three metrics aligned with database benchmarking standards [16], [17]:

- Average response time (MS): mean query execution time.
- Index memory usage (MB): storage footprint in memory.
- Index built time (s): time required to construct the index.

2.5. Experimental protocol

For each dataset and index type, the following protocol was executed:

a. Data loading: POI data was loaded into a MongoDB collection using batch insertion (1,000 documents per batch).
b. Index creation: each spatial index was created using MongoDB's createIndex() method with default parameters. For Geohash, the default precision level (12 characters, approximately 0.07 meters) was used.
c. Query execution: 1,000 random queries of each type were executed. Query points were generated uniformly within the bounding box of each dataset. The same set of 1,000 query locations was reused across all indexes to ensure fair comparison.
d. Metrics recording: response times were recorded for each query execution. Index memory usage was measured using MongoDB's stats() command after index creation. Build time was measured from index creation initiation to completion.
e. Statistical analysis: the experiment was repeated three times for each configuration. Reported values represent means across the three runs, with standard deviations calculated for key results.

2.6. Data preprocessing

POI data extracted from OpenStreetMap underwent the following preprocessing steps:

a. Removal of duplicate entries based on unique OSM identifiers.
b. Filtering to include only POIs with valid coordinates.
c. Conversion of coordinate systems to WGS84 (EPSG:4326).
d. Validation of GeoJSON geometry format.

3. RESULTS AND DISCUSSION

3.1. Query performance measurement

The experimental results reveal distinct performance patterns across index types and geographic scenarios. Table 1 reports detailed query response times for each scenario and index type. The results demonstrate a clear superiority of the 2dsphere index for dense urban environments, particularly for Q1 and Q3. Geohash performs competitively in suburban and rural areas for Q1 and Q2 with stable response times. The 2d index remain competitively only for small, flat areas, as simulated here. Table 2 summarizes the optimal index selection for each scenario-query combination.

Table 1. Query response times by scenario and index type

Scenario	Index	Q1 Nearest (ms)	Q2 Radius (ms)	Q3 Polygon (ms)
Dense urban	2d	45.2±3.1	52.8±4.2	68.5±5.3
Dense urban	2dsphere	12.3±1.2	14.7±1.5	18.2±1.8
Dense urban	Geohash	18.5±2.1	22.3±2.4	28.7±3.0
Suburban	2d	38.7±3.5	44.3±4.0	58.9±5.1
Suburban	2dsphere	15.1±1.4	16.8±1.6	19.5±2.0
Suburban	Geohash	14.2±1.3	15.9±1.5	24.1±2.3
Rural	2d	35.1±3.2	40.2±3.8	54.3±4.8
Rural	2dsphere	18.9±1.7	21.3±2.0	23.7±2.2
Rural	Geohash	12.7±1.1	13.8±1.2	22.5±2.1

Note: Values are means ± standard deviations across three experimental runs (n=3, 1.000 queries per run)

Table 2. Best index by scenario and query type

Scenario	Q1 (nearest)	Q2 (1 km radius)	Q3 (polygon)
Dense urban	2dsphere	2dsphere	2dsphere
Suburban	Geohash	2dsphere	2dsphere
Rural	Geohash	Geohash	2dsphere

3.2. Index storage and index cost

Table 3 presents index memory usage by scenario, while Table 4 reports index build times. Memory usage increases linearly with data volume across all index types. The 2dsphere index is the most resource-intensive, consuming approximately 2.3 times the memory of the 2d index and 1.5 times that of the Geohash. Construction time varies significantly across index types. The 2d index builds rapidly (under 2 seconds for all datasets), while 2dsphere requires substantially longer build times (approximately 5-9 seconds for the largest dataset). Geohash build times fall between these extremes.

Table 3. Index memory usage by scenario

Scenario	2d memory (MB)	2dsphere memory (MB)	Geohash memory (MB)
Dense urban (50 k)	12.3±0.5	28.5±1.2	18.7±0.8
Suburban (100 k)	24.1±1.0	56.8±2.3	37.2±1.5
Rural (30 k)	7.2±0.3	17.1±0.7	11.3±0.5

Table 4. Index build time by scenario

Scenario	2d memory (MB)	2dsphere memory (MB)	Geohash memory (MB)
Dense urban (50 k)	0.8±0.1	4.2±0.3	2.1±0.2
Suburban (100 k)	1.5±0.2	8.7±0.5	4.3±0.3
Rural (30 k)	0.5±0.1	2.5±0.2	1.3±0.1

Table 5 presents the percentage improvement of the best-performing index compared to the worst-performing index for each scenario and query type. These improvement percentages underscore the practical significance of appropriate index selection. In dense urban environments, selecting 2dsphere over the alternative indexes can yield up to 73.4% reduction in response time. Even in rural scenarios, where differences are less pronounced, improvements exceed 58% for polygon queries.

Table 5. Performance improvement of best index vs. worst index by scenario (%)

Scenario	Query type	Improvement (%)
Dense urban	Q1 Nearest	72.8
Dense urban	Q2 Radius	72.2
Dense urban	Q3 Polygon	73.4
Suburban	Q1 Nearest	63.3
Suburban	Q2 Radius	64.1
Suburban	Q3 Polygon	66.9
Rural	Q1 Nearest	63.8
Rural	Q2 Radius	65.7
Rural	Q3 Polygon	58.6

3.3. Analysis of performance patterns

The observed performance patterns can be explained through analysis of the underlying index mechanisms:

- Geometry accuracy vs. query precision: The 2dsphere index maintains high geometric accuracy through spherical calculations. In dense urban environments, the close proximity of points magnifies the importance of accurate distance calculation. The small distances typical in urban queries (often under 1 km) require precise spherical geometry to avoid misranking candidates. When 2dsphere calculates distances directly on the spheroid, it preserves correct ordering of nearest neighbours, reducing unnecessary candidate examination. The accuracy reduces the number of candidate documents that must be examined during query execution, yielding faster response time despite higher computational cost per operation.
- Hierarchical encoding efficiency: Geohash encodes locations hierarchically, with longer prefixes representing smaller geographic regions; in rural areas, the sparse distribution of points means that the initial high-level prefix filter eliminates most candidates quickly. The index traverses the prefix tree and examines only the branches containing potential candidates, effectively pruning the search space. This mechanism yields stable performance even when query regions cover larger geographic extents. When data is spatially dispersed, the prefix-based search quickly narrows the candidate set, minimizing the computational overhead. This observation aligns with theoretical expectations from space-filling curve literature [17] but provides empirical validation specific to MongoDB's implementation.
- Planar Limitations: The 2d index's planar approximation introduces distance distortion that increases with distance from the equator and with the size of the geographic extent. While this index is computationally efficient, the distortion leads to incorrect candidate ordering and additional processing to compensate for errors. In the suburban scenario, the larger geographic area amplifies the curvature effect, explaining why 2d's performance gap widens compared to urban settings. The 2d index's limited applicability confirms theoretical predictions but offers practical guidance for legacy system maintenance. Organizations with existing applications using flat map projections may continue using 2d indexes for small, localized areas, but new applications should favor 2dsphere or geohash.

- Data Density Effects: The dispersion of points benefits Geohash due to its hierarchical encoding. In rural areas, the prefix tree's branching structure limits the number of candidates at each level. Conversely, in dense urban areas, even small Geohash cells contain many points, reducing the effectiveness of prefix-based pruning and causing performance degradation relative to 2dsphere.
- The experimental results demonstrate that geographic context is a primary determinant of spatial index performance in MongoDB. The superiority of 2dsphere in dense urban environments stems from its ability to maintain accurate spherical distance calculations even when querying high-density point distributions.

3.4. Comparison with prior research

Our results align with the findings of Guo and Onstein [1], who identified Geohash as the most commonly adopted spatial indexing method in document-oriented databases. However, our work qualifies this observation by demonstrating that Geohash's advantages are context-dependent; 2dsphere outperforms Geohash in dense urban scenarios. The results support the conclusions regarding the effectiveness of document-oriented databases for geospatial processing. However, our study extends their analysis by isolating the impact of different indexing strategies within a single database system, providing more actionable guidance for practitioners. The observed memory usage patterns are consistent with theoretical expectations based on index structure complexity. 2dsphere's higher memory footprint reflects the additional information required for spherical geometry calculations, including projection metadata and spatial reference information. This trade-off between memory consumption and query performance is well documented in the database literature [18], [19].

These findings extend previous research by providing scenario-specific guidance that accounts for data density and query type. While prior work [9], [10] compared MongoDB's performance against other database systems, this study provides granular recommendations within MongoDB's index portfolio.

3.5. Practical recommendations

Based on the experimental findings, the following practical recommendations are offered:

3.5.1. Dense urban application

Favor the 2dsphere index for its accuracy and robustness. Spherical geometry calculations are essential to guarantee distance accuracy [18]. This recommendation applies particularly to applications involving:

- Localisation-based social networking.
- Ride-sharing and logistics.
- Urban navigation and mapping.
- Real estate and property search.

3.5.2. Rural or mixed application

Consider Geohash for its flexibility and stable performance. Its prefix search mechanism is particularly effective for dispersed data [20], [21]. Geohash is recommended for:

- Environment monitoring in agricultural areas.
- Wildlife tracking and conservation.
- Rural logistics and delivery.
- Telecom infrastructure mapping.

3.5.3. Legacy systems in small areas

The 2d index may suffice if accuracy constraints are relaxed and the geographic extent is limited to areas where curvature effects are negligible (approximately < 100 km²).

4. Resource-Constrained Environment: When memory is limited, the 2d index offers the smallest footprint, while Geohash provides a balanced compromise between memory usage and query performance.

3.6. Best practices and optimization

Based on current research [19], the following best practices should be observed:

a. Limit index count: Avoid over-indexing, which impairs write performance. Create spatial indexes only for fields used in geospatial queries.
b. Ensure indexes fit in RAM: All indexes should fit in available memory to prevent disk reads during query execution.

c. Use compound indexes carefully: When combining spatial and non-spatial filters, compound indexes may be beneficial but should be designed to match query patterns.
d. Monitor index usage: Use MongoDB's $indexStats and explain() to identify underutilized indexes and verify that queries are using appropriate indexes.
e. Consider sharding for large datasets: For datasets exceeding 1 million documents, geographic sharding based on the spatial field can improve scalability.
f. Adjust Geohash precision: For Geohash indexes, choose precision level based on query patterns. Higher precision (longer strings) improves accuracy but increases storage and reduces prefix pruning effectiveness.

3.7. Limitations and future research directions

3.7.1. Research limitations

This study has several limitations that should be acknowledged:

- Single-node deployment: The experiments were conducted on a standalone MongoDB instance; sharding and distributed performance were not assessed [19]. In a sharded deployment, data distribution across shards may affect performance patterns.
- Static datasets: The datasets represent snapshots and do not quantify the effects of frequent updates (inserts, updates, deletes). Write-intensive workloads may favor indexes with lower maintenance overhead.
- Geographic scope: The results are based on European datasets (France). Validation with datasets from other continents, particularly regions near the equator, would strengthen generalizability.
- Geometry types: The study focuses on point data. Index evaluation for complex geometries (LineString, MultiPolygon) was not performed [22].
- Geohash precision: The default precision level (12 characters) was used. Verifying the precision level may affect performance characteristics, particularly for Q2 and Q3 queries.
- Workloads representativeness: While the three query types represent common patterns, real-world workloads may include additional query types and mixed workloads.

3.7.2. Future research directions

Several promising directions for future research are identified:

- Distributed systems: Evaluation of spatial indexes in distributed MongoDB deployment with geographic sharding. Recent work on parallel spatial query processing [18] suggests that geographic data organization can significantly improve performance for big data.
- Real-time data streams: Investigation of index performance under continuous data updates, including insertion, deletion, and modification workloads.
- Hybrid indexes: Development and evaluation of hybrid indexing approaches combining multiple spatial indexing methods to leverage their respective strengths.
- Machine learning integration: Application of machine learning techniques for performance prediction and automated index selection [23], [24].
- Multimodal databases: Extension to databases supporting both spatial and graph data, enabling combined spatial and graph traversal queries [25], [26].

4. CONCLUSION

This paper has presented an empirical comparison of MongoDB's three spatial indexes-2d, 2dsphere, and Geohash- using real-world OpenStreetMap datasets across dense urban, suburban, and rural scenarios. The experimental methodology, based on reproducible protocols with 1,000 queries per configuration, provides systematic evidence for index selection guidance.

The key findings demonstrate that geographic context is a critical factor in index selection. In dense urban environments, the 2dsphere index significantly outperforms alternatives, achieving up to a 73.4% reduction in response time compared to the worst-performing index. In a rural scenario, Geohash provides the best performance for nearest neighbour and radius queries, with stable response times across verifying query regions. The 2d index remains usable only for legacy applications on limited flat areas, where its computational efficiency can be leveraged. These findings translate into practical recommendations for developers and architects. Applications operating in dense urban areas should prioritize 2dsphere despite its higher resource consumption. Rural and mixed applications should consider Geohash for its flexibility and stable performance. Resource-constrained environments may benefit from careful index selection considering the memory-query performance trade-offs.

The contributions of this work extend beyond simple performance comparisons. The reproducible experimental framework, open dataset specifications, and quantified recommendations provide a foundation for informed decision-making in MongoDB spatial index selection. By grounding recommendations in empirical evidence rather than theoretical speculation, this work addresses the gap identified in prior literature and offers actionable guidance for practitioners.

Future research should extend this evaluation to distributed environments, real-time data streams, and hybrid indexing approaches. The integration of machine learning techniques for performance prediction and automated index selection represents a particularly promising direction that could further reduce the complexity of spatial database optimization.

REFERENCES

[1] D. Guo and E. Onstein, "State-of-the-art geospatial information processing in NoSQL databases," *ISPRS Int. J. Geo-Information*, vol. 9, no. 5, 2020, doi: 10.3390/ijgi9050331.

[2] G. Boeing, "Modeling and analyzing urban networks and amenities with OSMnx," *Geographical Analysis,* vol. 57, no. 4, pp. 567–577, 2025, doi: 10.1111/gean.70009.

[3] "Geospatial Indexes - Database Manual - MongoDB Docs." 2025. [Online]. Available: https://www.mongodb.com/docs/manual/core/indexes/index-types/geospatial/2d/internals/

[4] R. S. Susarla and V. B. Kolla, "SQL Vs. NoSQL Comparative Analysis of Database Management Systems for Scalability and Performance," *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*, Boston, MA, USA, 2026, pp. 01-07, doi: 10.1109/ISDFS69419.2026.11459004.

[5] D. Bartoszewski, A. Piorkowski, and M. Lupa, "The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases," *Commun. Comput. Inf. Sci.*, vol. 1018, pp. 291–302, 2019, doi: 10.1007/978-3-030-19093-4_22.

[6] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, "MongoDB Vs PostgreSQL : A comparative study on performance aspects," pp. 243–268, 2021.

[7] C. Activity *et al.*, "GeoYCSB : A benchmark framework for the performance and scalability evaluation of geospatial NoSQL Databases Recommended Citation GeoYCSB : A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases," 2023.

[8] X. Yao *et al.*, "LandQ v2 : A MapReduce-based system for processing arable land quality big data," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 7, pp. 271, 2018, doi: 10.3390/ijgi7070271.

[9] C. Pandiani, "A comprehensive guide to better MongoDB performance." 2025. [Online]. Available: https://www.percona.com/blog/want-mongodb-performance-you-will-need-to-add-and-remove-indexes/

[10] G. Boeing, "OSMnx : New methods for acquiring, constructing, analyzing, and visualizing complex street networks," pp. 126–139, 2017, doi: 10.1016/j.compenvurbsys.2017.05.004.

[11] "OpenStreetMap Foundation: Strategic Plan." 2026, https://osmfoundation.org/wiki/Strategic_Plan

[12] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," in IEEE Pervasive Computing, vol. 7, no. 4, pp. 12-18, Oct.-Dec. 2008, doi: 10.1109/MPRV.2008.80.

[13] Z. Li *et al.*, "MapQA : open-domain geospatial question answering on map data," 2025, arXiv:2503.07871, vol. 1, no. 1. Association for Computing Machinery, doi: 10.48550/arXiv.2503.07871.

[14] W. Rowe, "MongoDB geolocation query examples," [Online]. Available: https://www.bmc.com/blogs/mongodb-geolocation-query-examples/.

[15] K. Chodorow and M. Dirolf, MongoDB: The Definitive Guide, O'Reilly Media, Inc., Sebastopol, 2010.

[16] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," *SoCC '10: Proceedings of the 1st ACM Symposium on Cloud Computing*, pp. 143-154, 2010, doi: 10.1145/1807128.180715.

[17] Y. Zhong, J. Han, T. Zhang, Z. Li, J. Fang, and G. Chen, "Towards parallel spatial query processing for big spatial data," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, 2010, pp. 2085-2094, doi: 10.1109/IPDPSW.2012.245.

[18] A. Aji *et al.*, "Hadoop-GIS : A high performance spatial data warehousing system over MapReduce," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, 2013.

[19] H. H. H. Tropf, "Multidimensional Range Search in Dynamically Balanced Trees." 1981, [online]. Available: https://hermanntropf.de/media/multidimensionalrangequery.pdf

[20] A. Al-Mamun, H. Wu, Q. He, J. Wang, and W. G. Aref, "A survey of learned indexes for the Multi-dimensional Space," *ACM Comput. Surv.* 58, 4, Article 96 (March 2026), 37 pages, doi: 10.1145/3768575.

[21] C. Technology and A. B. Thapa, "Optimizing MongoDB performance with indexing," 2022. [Online]. Available: https://www.mongodb.com/company/blog/performance-best-practices-indexing

[22] M. Staniek, R. Schumann, M. Züfle, and S. Riezler, "Text-to-OverpassQL : A natural language interface for complex geodata Querying of OpenStreetMap," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 562-575, 2024, doi: 10.1162/tacl_a_00654.

[23] M. M. F. Fahima, A. H. S. Sreen, S. L. F. Ruksana, D. T. E. Weihena, and M. H. M. Majid, "Machine learning for database management and query optimization," *Elementaria : Journal of Educational Research*, vol. 2, no. 1, pp. 96–108, 2024, doi: doi.org/10.61166/elm.v2i1.66.

[24] Z. Yang, "Machine learning for query optimization," 2022, Technical Report No. UCB/EECS-2022-194. [Online], Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-194.html.

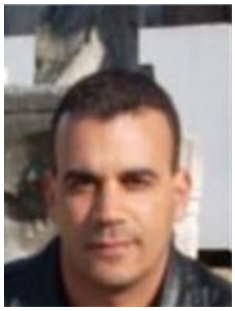
[25] D. Dominguez-Sal, N. Martinez-Bazan, V. Muntes-Mulero, P. Baleta, and J. L. Larriba-pey, "A discussion on the design of graph database benchmarks," *Lecture Notes in Computer Scienc*e, 2010, doi: 10.1007/978-3-642-18206-8_3.

[26] F. Cazzaro, J. Kleindienst, S. Marquez, and A. Quattoni, "ZOGRASCOPE : A new benchmark for property graphs," *In Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4239–4246, Suzhou, China, 2025, doi: 10.18653/v1/2025.findings-emnlp.227.

AUTHORS BIOGRAPHIES



Mahmoud Hammadi he is currently a Ph.D. student in Computer Science at the University of Mostaganem – Abdelhamid Ibn Badis in Algeria. He specializes in big data management and analysis. He can be contacted by email at mahmoud.hammadi.etu@univ-mosta.dz.



Dr. Charef Abdallah Bensalloua he is a computer science lecturer at the University of Mostaganem in Algeria. He specializes in business intelligence, big data, and geographical information systems (GIS). He can be contacted by email at: charef.abdellahbensalloua@univ-mosta.dz.