Kùzu: A Database Management System For “Beyond Relational” Workloads

Semih Salihoğlu
University of Waterloo
semih.salihoglu@uwaterloo.ca

I would like to share my opinions on the following question: *how should a modern graph DBMS (GDBMS) be architected?* This is the motivating research question we are addressing in the Kùzu project at University of Waterloo [4, 5].[1] I will argue that a modern GDBMS should optimize for a set of what I will call, for lack of a better term, “beyond relational” workloads. As a background, let me start with a brief overview of GDBMSs.

Overview: Modern GDBMSs [7, 8, 11, 15] adopt the property graph data model, where applications model their data as a set of node and relationship records and query these records using SQL-like high-level languages that have specialized graph syntax, such as the arrow syntax to describe the joins between node records. As other DBMSs with high-level query languages, GDBMSs are relational at their cores as these high-level constructs compile to relational operators, such as joins, filters, and projections. Yet, GDBMSs support workloads that require a set of features that are not traditionally optimized in RDBMSs. Here are some examples of such “beyond relational” capabilities:

- Complex many-to-many (m-n) joins: Datasets that are modeled as graphs often contain many-to-many relationships across nodes, e.g., friendships in social networks. Many applications search for patterns in these datasets, which translate to complex m-n joins.
- Recursive joins: Many queries of “graph workloads” can be recursive, e.g., to find indirect connections between accounts. While recursion is an afterthought in SQL, it is a first-class citizen feature in GDBMSs.
- Schema-flexible queries: Some applications require answering questions that require flexibility in the type of records to process, such as finding “any type of connections between accounts u and v”. GDBMSs have elegant means to ask these queries, while in SQL such queries are asked by unioning many queries.
- Heterogeneous datasets: Some datasets, such as knowledge graphs like Wikidata [1], model very complex domains, which: (i) cannot be tabulated; and (ii) are best modeled as URI-heavy RDF triples.

[1]I have previously written about the vision of Kùzu in a longer blog post without space constraints here [16].

Techniques For Beyond Relational Workloads: Our work in Kùzu has so far focused on integrating state-of-the-art techniques to evaluate complex many-to-many joins efficiently [5, 6, 9]. For example, Kùzu has a factorized [3, 13] query processor, which compresses intermediate results of many-to-many joins and implements novel worst-case optimal join algorithms [12, 17]. We are also implementing common recursive joins, such as shortest path and variable-length joins, and plan to implement a URI data type. However, a lot remains undone, such as integrating automata-based techniques for regular path queries and advanced string compression to manage URIs to achieve our vision of a feature-rich, competent GDBMS for beyond relational workloads. We are actively developing Kùzu to achieve this goal.

An Ideal Worth Remembering: With the hope of inspiring some PhD students, let me bring up another beyond relational capability that may at first seem unrelated to GDBMSs: *the ideal of systems with general deductive capabilities*. Consider a database of 3 items and their colors, 1 red, 1 blue, and 1 with a NULL color, and a constraint that every item must be red or blue. Suppose we ask: “what is the maximum of the count of red items and blue items?” With relational capabilities of joins and group by-aggregates, one would compute the answer as 1, yet with a simple deduction, we can see that the answer is 2, as the unknown value is either red or blue. The ideal of information systems that can perform advanced logical deductions, for example proofs by contradictions or if-then implications, has existed since the birth of CS [2], with deep connections to AI. Because logical deductions are often recursive and recursion is a first-class citizen in GDBMSs, GDBMSs can integrate deductive capabilities. In fact, some RDF systems [10], which are graph-based DBMSs, perform limited OWL-based deductions [14]. With the current momentum around symbolic AI, it is a good time for our community to revisit these ideals, and maybe some research groups can attempt to develop systems with such capabilities and without forgetting our relentless focus on performance and scalability!

REFERENCES

[1] Wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page, 2023.

[2] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2004.

[3] Factorization & Great Ideas from Database Theory. https://kuzudb.com/blog/factorization.html, 2023.

[4] Xiyang Feng, Guodong Jin, Ziyi Chen, Chang Liu, and Semih Salihoğlu. Kùzu Source Code. https://github.com/kuzudb/kuzu, November 2022.

[5] Xiyang Feng, Guodong Jin, Ziyi Chen, Chang Liu, and Semih Salihoğlu. Kùzu Graph Database Management System. In *The Conference on Innovative Data Systems Research*, 2023.

[6] Pranjal Gupta, Amine Mhedhbi, and Semih Salihoglu. Columnar Storage and List-based Processing for Graph Database Management Systems. *Proceedings of the VLDB Endowment*, 14(11):2491–2504, 2021.

[7] Chathura Kankanamge, Siddhartha Sahu, Amine Mhedbhi, Jeremy Chen, and Semih Salihoglu. Graphflow: An Active Graph Database. In *Proceedings of the ACM International Conference on Management of Data*, 2017.

[8] Memgraph. https://memgraph.com/, 2023.

[9] Amine Mhedhbi, Chathura Kankanamge, and Semih Salihoglu. Optimizing One-time and Continuous Subgraph Queries using Worst-Case Optimal Joins. *ACM Transactions on Database Systems*, 46(2):1–45, 2021.

[10] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RDFox: A Highly-Scalable RDF Store. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web*, 2015.

[11] Neo4j. http://neo4j.com, 2023.

[12] H. Ngo, C. Ré, and A. Rudra. Skew Strikes Back: New Developments in the Theory of Join Algorithms. *SIGMOD Record*, 42(4):5–16, 2014.

[13] Dan Olteanu and Jakub Závodnỳ. Size Bounds For Factorised Representations of Query Results. *ACM Transactions on Database Systems*, 40(1):1–44, 2015.

[14] OWL Semantic Web Standard. https://www.w3.org/OWL/, 2022.

[15] Tigergraph. http://tigergraph.com, 2023.

[16] What Every Competent GDBMS Should Do. https://kuzudb.com/blog/what-every-gdbms-should-do-and-vision.html, 2023.

[17] Why (Graph) DBMSs Need New Join Algorithms: The Story of Worst-case Optimal Join Algorithms. https://kuzudb.com/blog/wcoj.html, 2023.