Almendros-Jiménez, J. M.: Debugging Empty Answers in Graph Queries. In: Sáenz-Pérez, F. (ed.) Actas de las XXV Jornadas de Programación y Lenguajes (PROLE 2026). Sistedes (2026). https://hdl.handle.net/11705/PROLE/2026/6

Debugging Empty Answers in Graph Queries (Extended Abstract)

Jesús M. Almendros-Jiménez

University of Almería.
Carretera de Sacramento s/n, 04120-Almería, Spain.
jalmen@ual.es

1 Introduction

In recent years, there has been a significant increase of interest in Graph Databases (GDBs) [9,1]. Unlike traditional relational systems, GDBs utilize a property graph model where data is structured as a network of nodes and directed relationships. In this model, both nodes and relationships serve as first-class citizens and can store properties (key-value pairs) that function as database attributes. GDBs are particularly powerful in domains where the value lies in the connectivity of the data—such as social networks [6], fraud detection [12], spatial data [4] and bioinformatics [7] —allowing for complex relationship traversal that would be computationally expensive in traditional database systems.

Graph database standards and implementations have matured significantly. For instance, Neo4j[1] [14] has emerged as the most prominent native graph database, popularized by its creation of Cypher[2][2], an intuitive, declarative query language that has now been standardized as GQL (Graph Query Language)[3] by the ISO [5]. Beyond native solutions, Apache TinkerPop[4][11] also provides the Gremlin traversal query language[5][10], allowing to interact with various graph systems through a common interface. Semantics of graph databases and associated query languages have also been studied in several studies [2,8].

Debugging remains one of the most resource-intensive phases of software development. While imperative programming environments benefit from a mature ecosystem of diagnostic tools—such as breakpoints, step-by-step execution, and state watchers—declarative database query debugging has been comparatively neglected. As noted by [3], debugging queries is inherently non-trivial, as it requires a double understanding of the logical query structure and the underlying data instance. In the context of graph databases, this challenge is amplified; the interconnected nature of the data means a single structural failure can render an entire complex pattern match void without providing any diagnostic feedback.

[1] https://neo4j.com/
[2] https://neo4j.com/docs/cypher-manual/current/introduction/
[3] https://www.gqlstandards.org/
[4] https://tinkerpop.apache.org/
[5] https://tinkerpop.apache.org/gremlin.html

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.

The fundamental objective of debugging is to isolate an error and pinpoint its location. In the context of database queries, syntactic errors typically stem from a misuse of the database schema vocabulary; referencing incorrect entity names or labels leads to immediate failure. While most modern tools effectively flag these structural issues, semantic errors are far more elusive. These occur when the complex interrelationships within the data are misunderstood, resulting in queries that are syntactically valid but logically flawed. A common symptom of a semantic error is the empty answer set[13], which creates an ambiguity: does the data simply not exist, or is the query formulation incorrect? This null response is a primary source of developer frustration.

Consider the following Cypher query which seeks titles of movies (`m.title`) from a movie database released after 2005 (`m.released > 2005`) where Tom Hanks acted `(p:Person {name: "Tom Hanks"})-[:ACTED_IN]-> (m: Movie)` and Christopher Nolan directed `(d:Person {name: "C. Nolan"}) -[:DIRECTED] -> (m:Movie)`:

```
MATCH (p:Person {name: "Tom Hanks"})-[:ACTED_IN]->
        (m:Movie)<-[:DIRECTED]-(d:Person {name: "C. Nolan"})
WHERE m.released > 2005
RETURN m.title
```

If this query returns an empty set, the root cause remains opaque to the user. The failure could be structural (e.g., Tom Hanks has never collaborated with C. Nolan), constraint-based (e.g., they collaborated, but only on films released prior to 2005), or data-missing/lexical (e.g., a misspelling of "C. Nolan" or the absence of these specific entities in the dataset).

This paper presents the design and implementation of a specialized debugger for graph databases. The initial phase of this development focuses on diagnosing empty answers. By identifying the specific point where a pattern fails to map to the database instance, the tool provides causes for missing results and distinguishes between data-level inconsistencies and logic errors.

Acknowledgments. This study was funded by Grant PID2021-124124OB-I00 funded by the Spanish Government MCIN/AEI/10.13039/501100011033 and the "ERDF A way of making Europe".

References

1. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR) **50**(5), 1–40 (2017)
2. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., Taylor, A.: Cypher: An evolving query language for property graphs. In: Proceedings of the 2018 International Conference on Management of Data. pp. 1433–1445 (2018)
3. Gathani, S., Lim, P., Battle, L.: Debugging database queries: A survey of tools, techniques, and users. In: proceedings of the 2020 CHI conference on human factors in computing systems. pp. 1–16 (2020)

Almendros-Jiménez, J. M.: Debugging Empty Answers in Graph Queries. In: Sáenz-Pérez, F. (ed.) Actas de las XXV Jornadas de Programación y Lenguajes (PROLE 2026). Sistedes (2026). https://hdl.handle.net/11705/PROLE/2026/6

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.

4. Gómez, L.I., Kuijpers, B., Vaisman, A.A.: Analytical queries on semantic trajectories using graph databases. Transactions in GIS **23**(5), 1078–1101 (2019)
5. ISO/IEC: Iso/iec 39075:2024 information technology — database languages — gql. Tech. rep., International Organization for Standardization (2024), `https://www.iso.org/standard/76120.html`
6. Kanakaris, N., Michail, D., Varlamis, I.: A comparative survey of graph databases and software for social network analytics: The link prediction perspective. In: Graph Databases, pp. 36–55. CRC Press (2023)
7. Mazein, I., Rougny, A., Mazein, A., et al.: Graph databases in systems biology: a systematic review. Briefings in Bioinformatics **25**(6) (2024)
8. Pino, E., Orejas, F., Mylonakis, N., Pasarella, E.: A logical approach to graph databases. Journal of Logical and Algebraic Methods in Programming **141**, 100997 (2024)
9. Robinson, I., Webber, J., Eifrem, E.: Graph Databases: New Opportunities for Connected Data. O'Reilly Media, Inc. (2015)
10. Rodriguez, M.A.: The gremlin graph traversal machine and language (invited talk). In: Proceedings of the 15th symposium on database programming languages. pp. 1–10 (2015)
11. Rodriguez, M.A.: The tinkerpop graph traversal stack. In: Proceedings of the 2015th International Conference on Management of Data. pp. 1617–1624 (2015)
12. Srivastava, S., Singh, A.K.: Fraud detection in the distributed graph database. Cluster Computing **26**(1), 515–537 (2023)
13. Vasilyeva, E., Thiele, M., Bornhövd, C., Lehner, W.: Answering "Why Empty?" and "Why So Many?" queries in graph databases. Journal of Computer and System Sciences **82**(1, Part A), 3–22 (2016)
14. Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., Partner, J.: Neo4j in action. Manning Publications Co. (2014)

Almendros-Jiménez, J. M.: Debugging Empty Answers in Graph Queries. In: Sáenz-Pérez, F. (ed.) Actas de las XXV Jornadas de Programación y Lenguajes (PROLE 2026). Sistedes (2026). https://hdl.handle.net/11705/PROLE/2026/6

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.