



Analysis Graph-Based Data Model For Simple Search Application Using NER Data

Ade Hodijah and Trisna Gelar

Computer and Informatics Engineering
Politeknik Negeri Bandung
Bandung, Indonesia

ade.hodijah@polban.ac.id, trisna.gelar@polban.ac.id

Abstract. Analysis of the seven steps of the Graph Data Modeling approach to obtain application functionality by utilizing the links (relationships) between entities (nodes) of the Named Entity Recognition (NER) entities was the aim of this study. The seven steps have two main models: the Initial and the Refactoring graph. This paper used the NER horticultural dataset to support data requirements in simple search applications with graph-based data storage technology (Neo4j). The domain of applying the NER model was an Indonesian language instructional text document on the sub-topic of horticulture in the context of the cultivation process of fruit trees in Indonesia. Extraction results (NER entities) from horticultural records were stored in Neo4j, and the Cypher query provided by Neo4j was used to support the data search application. Use case scenarios as application functionalities had been successfully tested based on sufficient questions (keywords) for scientific study material. The experimental results showed that the Refactoring graph was faster than the Initial graph. The total number of nodes scanned in the Refactoring graph was almost three times less than in the Initial graph by specifying labels, nodes, and relationships. This paper can be a reference in graph-based modelling.

Keywords: NER, horticultural, Graph Data Modeling

1. Introduction

Named Entity Recognition (NER) was used to identify entities from an Indonesian language horticultural instructional text document [1]. The horticultural instructional text documents used in this research were sourced from four books on the sub-theme of fruit cultivation (mango, orange, passion fruit, and dragon fruit), which have a similar discussion framework to the research [1]. The results of the NER research obtained a list of 14 entities, namely crop, chemical, quantity, distance, disease, pests, location, count, trait, varieties, period, tools, method, and condition [1]. This paper used three of the 14 entities as datasets: crop, pests, chemical, and period. There was a lot of research that used entities generated by NER as a node in a graph [2], [3], [4].

This paper used a more straightforward approach that focuses on implementing the seven steps of graph data modelling fundamental from Neo4j [5] in providing a graph data model with the best Cypher performance using the horticultural dataset generated by NER [1]. There was another paper (SuMo) discussed scenario-based graph data modelling [6], having two phases as the rules for the subsequent transformation from real-world scenarios (Entity-Relationships or ER) to produce a graph model. Other

research was developed for the NoSQL database formal schema introduction from the ER model [7], [8], [9].

The performance discussed in paper [6] was about a naïve approach (tuples are mapped to nodes, and Foreign Keys are to edges) and SuMo comparison. Otherwise, the goal of the data modelling should be to reduce the graph size touched by a query. In [5], node labels serve as an anchor point for a query. Specifying a label defines a subset of one or more nodes to start a query [5]. Using a label helps reduce the amount of retrieved data [5]. Neo4j has been optimized to handle the traversal process by implementing a graph algorithm [10], and other studies have used it [11], [12] as graph database technology. Table 1 shows the research research gap in this paper.

Table 1. Similar Scientific Works

Year	Title	Author	Domain
2022	Serverless Named Entity Recognition (NER) untuk Teks Instruksional Pertanian Kota	T. Gelar, A. Nanda, and A. Bakhrun	NER, Horticultural
2023	Graph Data Modeling Fundamentals	Neo4j Graph academy	<ol style="list-style-type: none"> 1. Question list defining 2. Initial graph modeling 3. Questions and graph checking 4. Instances model testing 5. The Cypher performance reviewing 6. Graph model refactoring 7. Retesting
2016	Towards a systematic approach to graph data modeling: Scenario-based design and experiences	M. Zhao, Y. Liu, and P. Zhou	<ol style="list-style-type: none"> A. Scenario-based domain modeling: <ol style="list-style-type: none"> 1. Discovery 2. Invention 3. Integration 4. Split off B. Model transformation: <ol style="list-style-type: none"> 1. Creating nodes by domain classes 2. Extending nodes by generalization 3. Generating edges by associations 4. Appending edges by operations

2. Methods

This paper applies an iterative graph data modeling approach from Neo4j Graph Academy [5]. The horticultural dataset was generated using the NER model [1], as seen in Fig. 1.

```

{"id":1300,"text":"Buah naga atau dragon fruit,
juga dikenal dengan apel kaktus, pitaya dan pitahaya.
Tanaman ini termasuk dalam keluarga kaktus (Cactaceae)
yang diduga berasal dari Amerika Selatan.,"label":
[[5,9,"CROP"],[15,28,"CROP"],[49,60,"CROP"],[62,68,"CROP"],
[73,81,"CROP"], [119,125,"CROP"],[127,136,"CROP"],
[163,178,"LOCATION"]],"Comments":[]}

```

Fig. 1. Horticultural dataset NER form snippet [1]

Fig. 1 shows a sentence that included a dragon fruit words had [15, 28, "CROP"] column. It has a label, namely CROP, and it was from char of 15 to 28 [13]. A research method is seen in Fig. 2.

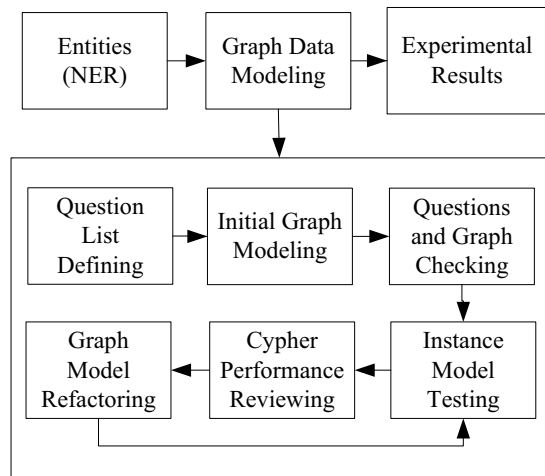


Fig. 2. Research methods

There are seven steps to create a graph data model:

- Recognize the domain and establish application-specific use cases (Question List Defining).
- Create the initial model of the graph data (Initial Graph Modeling):
- Compare the use cases to the initial data model (Questions and Graph Checking).
- Use Cypher to create the graph using the test data (Instances Model Testing).
- Evaluate the use cases by comparing their performance to the graph (Cypher performance Reviewing).
- Refactor the graph data model for performance-related reasons or because the main use cases have changed (Graph Model Refactoring).
- Apply the refactoring to the graph and use Cypher to retest.

The Neo4j components used nodes, labels, relationships, and properties to define the graph data model [5]. These components were tested using the Cypher query [14] to know the best performance for each model. The performance experimental used only three of 14 horticultural NER entities [1] as crop, pests, chemical, and period datasets.

The experimental used three questions related to the horticultural domain problem, which are:

- a. What horticultural plants are fruit types?
- b. What crop comes from a particular country?
- c. What pest attacked a particular crop?

The three questions defined based on the graph data modelling components for research experimental parameters as follows:

- a. The labels usage in a node (Labels)
- b. Replacing property with the nodes or implementing intermediate nodes for inheritance or the hyperedges relationships (Nodes)
- c. Utilizing specialized relationships (Relationships)

These experimental parameters run the Cypher for the initial graph modelling (initial) compared to the graph model refactoring (refactoring). The performance results were obtained from the execution times and the number of rows. The PROFILE keyword of a neo4j query plan is used to compare the Cypher query performance [5]. A query plan shows the number of DB hits for searching a particular node. However, the execution completed time established by the query profile may differ when using a small dataset because the cache is automatically populated [5]. Therefore, the experiment used three datasets called NER results [1]. Then, a dummy dataset with about 1,000 records was generated to support the experiment. The execution time results of the initial compared to the refactoring ten times to get the average, while for the total nodes conducted in one.

3. Results And Discussion

Fig.3 shows the initial graph modelling, and Table 2 shows the experimental results of it.

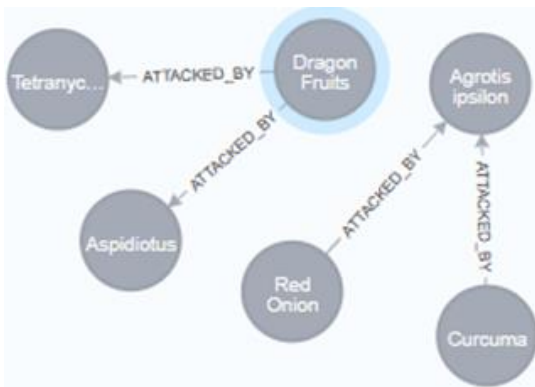


Fig. 3. The initial graph modeling

Fig. 4 shows the Query Plan results from Neo4j for use case number two, that is, what Crop comes from a particular country? The steps required are: (1). Scan all Crop nodes

in the graph; (2). Retrieve a Crop by its origin; (3). Return the name of the Crop. These steps are executed by all use cases, where all the Cypher queries use a Crop as an anchor point.

Table 2. The Initial Graph Experimental Results

Use Case	Cypher	Results	
		Total nodes	Execution times (ms)
What horticultural plants are fruit types?	PROFILE MATCH (c:Crop {type: 'Fruits'}) RETURN c.name AS Crop	502	2.0
What crop comes from a particular country?	PROFILE MATCH (c:Crop {origin: 'America'}) RETURN c.name AS Crop	502	2.0
What pest attacked a particular crop?	PROFILE MATCH (c:Crop {name: 'Dragon Fruits'})-[:ATTACKED_BY]-(p:Pest) RETURN p.name AS Pest	502	1.8

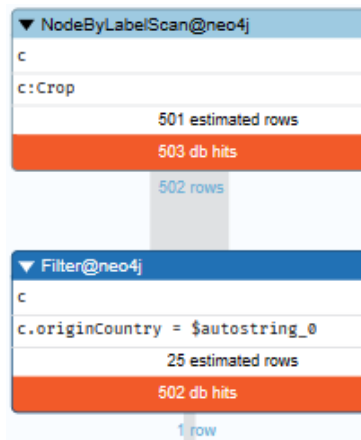


Fig. 4. Query plan result for use case number two which returned 502 nodes

The graph should be modified or refactored to get the best model. One of three causes for refactoring [5] is that Cypher's performance in the use cases is not optimal, especially when the graph scales. This paper focuses on refactoring the labels (Fig. 5), the nodes (Fig. 6), and the relationships (Fig. 9). The refactoring approach means that this paper was given a more specific name for the labels, the nodes, and the relationships. The representation was based on the application's use case specification or question list.

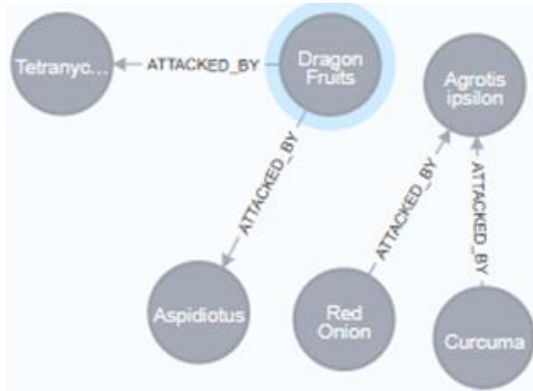


Fig. 5. The labels refactoring

Table 3. Experimental Results Of The Labels Refactoring

Use Case	Cypher	Results	
		Total nodes	Execution times (ms)
What horticultural plants are fruit types?	PROFILE MATCH (c:Fruits) RETURN c.name AS Crop	168	1.4

The Fruits became labelled in the refactoring graph as seen in Table 4 (A), where the Fruits were a value of the type property as seen in Table 4 (B). There are only three use cases, as mentioned in the Method section, which no longer need the Crop as the label. Therefore, the label of Crop was updated by Fruits. Otherwise, the Crop label should not be updated if a use case with a query used the Crop label as an anchor point of the scanning process. The labels will help to reduce the number of nodes retrieved and the group nodes used effectively [5].

Table 4. Example of A Node Property Became A Label

<p>Node Properties </p> <p>Crop</p> <p><id> 3043</p> <p>name Dragon Fruits</p> <p>origin AmericaOrigin</p> <p>type Fruits</p> <p>(A)</p>	<p>Node Properties </p> <p>Fruits</p> <p><id> 3043</p> <p>name Dragon Fruits</p> <p>origin America</p> <p>(B)</p>
--	--

However, the label usage is not overused but is used when it will help with most of the use cases. A best practice is limiting the number of node labels to 4 [5]. Table 5 shows the experimental result of how the number of labels affected a node's performance for 4, 10, and 50 labels. The result shows no significant effect when running the matching (retrieve) cipher query, even when the label was 50. The impact came when running the merging (insert) the Cypher query. The longest time to create a node was when the number of labels was 50.

Table 5. Experimental Results Of The Number of Labels In A Node

Total Labels	Cypher Query	Total Nodes	Execution Times (ms)
4	PROFILE MATCH (c:Fruits4 {name: 'Dragon Fruits1000'}) RETURN c.name AS Fruits	1000	2.6
10	PROFILE MATCH (c:Fruits10 {name: 'Dragon Fruits1000'}) RETURN c.name AS Fruits	1000	2.4
50	PROFILE MATCH (c:Fruits50 {name: 'Dragon Fruits1000'}) RETURN c.name AS Fruits	1000	2.7

The steps required when a property becomes labelled as shown in Fig.4, but when the property becomes a node, are described as follows: (1). Scan all Fruits nodes in the graph; (2). Follow the CAME_FROM relationship to Country; (3). Evaluate the Country by its name; (4). Return the name of the Fruits from the CAME_FROM relationship between the two nodes.

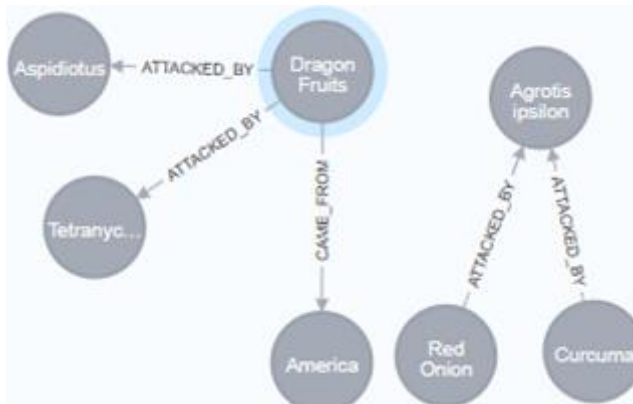


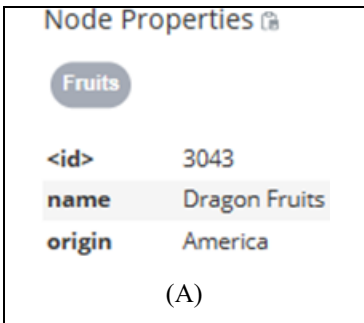
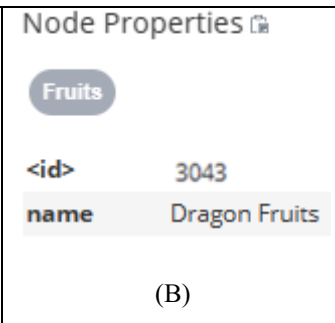
Fig. 6. The nodes refactoring

Table 6. Experimental Results Of The Nodes Refactoring

Use Case	Cypher	Results	
		Total nodes	Execution times (ms)
What crop comes from a particular country?	PROFILE MATCH (c:Fruits)-[:CAME_FROM]-(:Country {name: 'America'}) RETURN c.name AS Crop	168	1.4

Table 3 shows property becomes a label, Table 6 shows property becomes a node with a relationship, and Table 11 shows property becomes a node or a relationship. When the property was changed to the nodes, as seen in Table 7 (B), it aimed to reduce the property duplication (Table 7 A) and get the advantage property in many nodes by changing it to the nodes. De-duplicating data will add benefit if the Cypher query is done through a node [5].

Table 7. Example Of A Node Property Became A Relationship

 <p>(A)</p>	 <p>(B)</p>
---	---

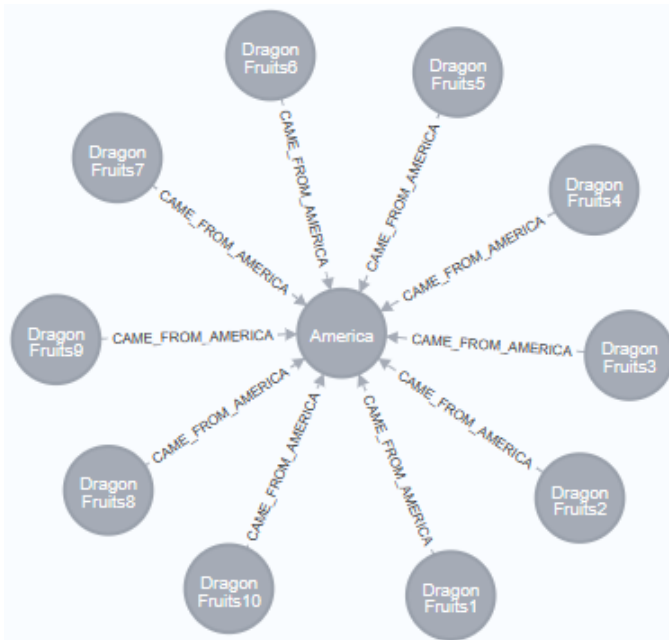


Fig. 7. The nodes refactoring

Fig.7 shows the ten sample graph nodes of 1000 nodes for the property, node, and relationship experimental. Table 8 shows that the property model took the longest time because it needs property value checking and scanning 1000 nodes. Other models (the node and relationship) only need one node scanning.

Table 8. Experimental Results Of The Property (A), Node (B), and Relationship (C)

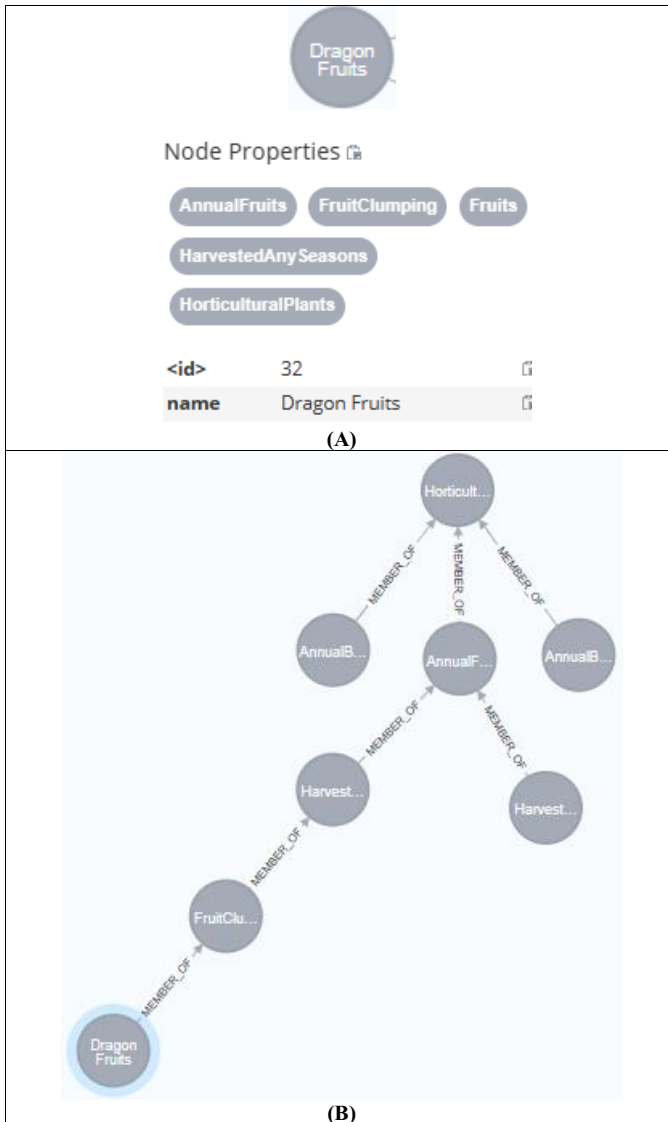
Type	Cypher Query	Total Nodes	Execution Times (ms)
A	PROFILE MATCH (c:Fruits {origin: 'America'}) RETURN c.name AS Fruits	1000	3.5
B	PROFILE MATCH (c:Country)-[]-(f:Fruits) RETURN f.name AS Fruits	1	3.2
C	PROFILE MATCH (f:Fruits)-[:CAME_FROM_AMERICA]-(c:Country) RETURN f.name AS Fruits	1	3.3

Neo4j, as a native graph database, is implemented to traverse relationships quickly rather than properties in the nodes [5]. When the origin became a property, as seen in

Table 7 (A), the query would need to retrieve all Fruits and then check the value of the origin property, which has a particular country name. All of these fruit nodes would need to be evaluated. After the origin property was changed to CAME_FROM relationships, the Cypher query will scan the relationship and then get only the Fruits nodes with relationships to the America nodes. Relationships are fast to traverse and only take up a little space in the graph [5].

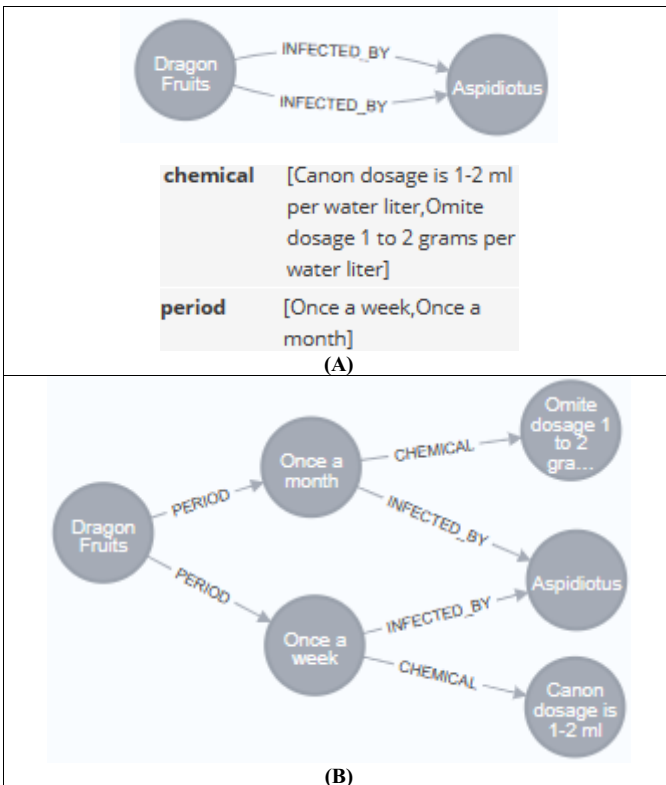
There was another issue related to the node labels usage, which are IS-A and M-N relationships. Avoid using labels excessively, but do so if they will be helpful for most use cases. Table 9 shows the horticultural hierarchy [15] as seen in Fig.8. Dragon fruits are in Horticultural Plants – Annual Fruits – Harvested Any Seasons – Fruit Clumping classification. The recursive Ciper query can be used by refactoring the labels hierarchy, as seen in Table 9 (A), into the nodes hierarchy, as seen in Table 9 (B).

Table 9. Experimental Results Of IS-A Relationships: (A) Initial; (B) Refactoring



In Neo4j, there is no way to create a relationship that connects a relationship to a third node. Neo4j relationships can only connect two nodes [5]. The INFECTED_BY hyperedge has an array of chemical properties and has a period association with the chemicals, as seen in Table 10 (A). Since nodes are connection points [5], creating a node in the middle of the hyperedge would benefit. When there was a use case to retrieve what should be given to the Dragon Fruits infected by Aspidiotus, it would first scan the Aspidiotus nodes. After that, the Cypher query will retrieve two-period nodes: Once a week and Once a month with each chemicals node. Furthermore, based on the chemical relationship property being changed to a node, it would be able to answer questions about how the same chemical is used in multiple fruits.

Table 10. Experimental Results Of M-N Relationships



In Table 6, the explanation stated that the data model would benefit from having the CAME_FROM relationships replaced with the origin property. Moreover, having specialized relationships between the nodes would be more beneficial. Table 11 shows how the ATTACKED_BY relationships, as seen in Table 5, followed by the pest name, which was as follows:

- ATTACKED_BY_KUTU_BATOK
- ATTACKED_BY_HAMA_TUNGAU
- ATTACKED_BY_ULAT_TANAH
-

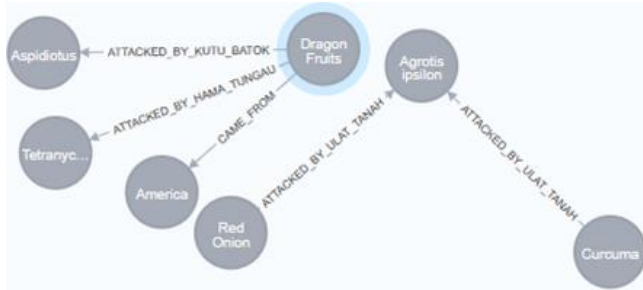


Fig. 8. The nodes refactoring

These specified relationships were made based on use case number three. The Cypher query would scan only the `ATTACKED_BY_KUTU_BATOK` relationships and then get only the Fruits nodes, which have relationships to the Aspidiotus nodes. These refactoring (labels, nodes, relationships) made the initial graphs more meaningful in further analysis, performant to query, and sharable between nodes.

Table 11. Experimental Results Of The Relationships Refactoring

Use Case	Cypher	Results	
		Total nodes	Execution times (ms)
What pest attacked a particular crop?	PROFILE MATCH (:Fruits)-[:ATTACKED_BY_KUTU_BATOK]-(p:Pest) RETURN DISTINCT p.name AS Pest	168	1.4

Table 12 shows the experimental results for all use case scenarios. The initial graph modelling results were longer than the graph model refactoring with fewer nodes scanned.

Table 12. Refactoring Grap Experimental Results For 1000 Datasets

Use Case	Initial		Refactoring	
	Total nodes	Execution times (ms)	Total nodes	Execution times (ms)
What horticultural plants are fruit types? (Labels)	502	2.0	168	1.4
What crop comes from a particular country? (Properties)	502	2.0	168	1.4
What pest attacked a particular crop? (Relationships)	502	1.8	168	1.4

4. Conclusions

This paper has explored graph-based data modeling without utilizing graph algorithm consideration available in Neo4j [16]. Three critical points in modeling the best graph: specify the labels, nodes, and relationships. This research experimented with graph data modeling using a horticultural dataset. The total number of nodes scanning in the Initial graph was 502 using Crop as labels, while the total nodes scanning was 168 in the refactoring graph. The total nodes affect the execution times. The study result showed that the label use case of the refactoring was faster (about 1.4 ms) than the initial in all use cases. The more specific the use of labels, nodes, and relationships, the fewer the number of nodes scanned and the faster the query execution time.

References

- [1] T. Gelar, A. Nanda, and A. Bakhrun, "Serverless Named Entity Recognition untuk Teks Instruksional Pertanian Kota," *J. Tek. Inform. dan Sist. Inf.*, vol. 8, no. 3, pp. 597–606, 2022, doi: 10.28932/jutisi.v8i3.5447.
- [2] I. Harrando and R. Troncy, "Named Entity Recognition as Graph Classification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12739 LNCS, pp. 103–108, 2021, doi: 10.1007/978-3-030-80418-3_19.
- [3] Y. Fang, D. Zhang, and G. Wu, "Toward establishing a knowledge graph for drought disaster based on ontology design and named entity recognition," *J. Hydroinformatics*, vol. 00, no. 0, pp. 1–14, 2023, doi: 10.2166/hydro.2023.046.
- [4] T. Al-Moslemi, M. Gallofre Ocana, A. L. Opdahl, and C. Veres, "Named Entity Extraction for Knowledge Graphs: A Literature Overview," *IEEE Access*, vol. 8, pp. 32862–32881, 2020, doi: 10.1109/ACCESS.2020.2973928.
- [5] Neo4j Graphacademy, "Graph Data Modeling Fundamentals." <https://graphacademy.neo4j.com/courses/modeling-fundamentals/>.
- [6] M. Zhao, Y. Liu, and P. Zhou, "Towards a systematic approach to graph data modeling: Scenario-based design and experiences," *Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE*, vol. 2016-Janua, no. July 2016,

- pp. 634–637, 2016, doi: 10.18293/SEKE2016-119.
- [7] M. Šestak and M. Turkanović, “Extended Property-level k-vertex Cardinality Constraints Model for Graph Databases,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 4, pp. 126–138, 2023, doi: 10.1016/j.jksuci.2023.03.013.
- [8] G. C. G. Van Erven, R. N. Carvalho, W. M. C. Da Silva, S. Lifschitz, H. Vera-Olivera, and M. Holanda, “Designing graph databases with GRAPHED,” *J. Database Manag.*, vol. 30, no. 1, pp. 41–60, 2019, doi: 10.4018/JDM.2019010103.
- [9] A. Schultheiß, A. Boll, and T. Kehrer, “Comparison of graph-based model transformation rules,” *J. Object Technol.*, vol. 19, no. 2, pp. 1–21, 2020, doi: 10.5381/jot.2020.19.2.a3.
- [10] S. A. Srivastava, “Graph: A Possible Solution for Environmental Pollution!” <https://neo4j.com/developer-blog/graph-a-possible-solution-for-environmental-pollution/>.
- [11] I. Hristoski1 and M. Malenkovska Todorova, “Graph Database Modeling of Urban Area Road Networks,” no. December, 2021, doi: 10.20544/tts2021.1.1.21.p08.
- [12] S. P. L. Filho, M. C. Cavalcanti, and C. M. Justel, “Graph Modeling for Topological Data Analysis,” *Lect. Notes Bus. Inf. Process.*, vol. 363, no. June 2020, pp. 193–214, 2019, doi: 10.1007/978-3-030-26169-6_10.
- [13] Stackoverflow, “Formatting training dataset for SpaCy NER.” <https://stackoverflow.com/questions/47443976/formatting-training-dataset-for-spacy-ner>.
- [14] Neo4j Graphacademy, “Cypher Fundamentals.” <https://graphacademy.neo4j.com/courses/cypher-fundamentals/>.
- [15] Direktorat Jenderal Hortikultura Kementerian Pertanian and Badan Pusat Statistik, “Pedoman Statistik Pertanian Hortikultura (SPH),” pp. 1–99, 2020.
- [16] M. Krishnamoorthy and R. Karthikeyan, “Pattern mining algorithms for data streams using itemset,” *Meas. Sensors*, vol. 24, no. August, p. 100421, 2022, doi: 10.1016/j.measen.2022.100421.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

