IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 4, December 2022, pp. 1261~1269
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i4.pp1261-1269

Automated dynamic schema generation using knowledge graph

Priyank Kumar Singh[1], Sami Ur Rehman[1], Darshan J.[1], Shobha G.[2], Deepamala N.[2]
[1]Department of Computer Science and Engineering, R.V College of Engineering, Bengaluru, India
[2]Department of Computer Science and Engineering, Faculty of Computer Science and Engineering, R.V College of Engineering, Bengaluru, India

Article Info

Article history:

Received Sep 11, 2021
Revised Jun 26, 2022
Accepted Jul 20, 2022

Keywords:

Database
Graph neural network
Knowledge graph
Natural language processing
Schema

ABSTRACT

On the internet where the number of database developers is increasing with the availability of huge data to be stored and queried. Establishing relations between various schemas and helping the developers by filtering, prioritizing, and suggesting relevant schema is a requirement. Recommendation system plays an important role in searching through a large volume of dynamically generated schemas to provide database developers with personalized schemas and services. Although many methods are already available to solve problems using machine learning, they require more time and data to learn. These problems can be solved using knowledge graphs (KG). This paper investigates building knowledge graphs to recommend schemas.

Corresponding Author:

Priyank Kumar Singh
Department of Computer Science and Engineering
R.V College of Engineering
Banglore-560059, Karnataka, India
Email: priyankkumars.cs18@rvce.edu.in

1. INTRODUCTION

The database is a very integral part of any software application and relational database is the most popular type of database system available. Designing a database is a very tedious and critical part of any software engineering process. A large share of the total time is spent on designing the database. To design a database, professional database designers are required to avoid any problems in the future. So, in this ever-advancing world of technology, there is tremendous growth in the field of software engineering and hence there is great demand for efficient data storage and processing techniques. Hence under this big hood of software development, there is an implicit need of automating the entire process of designing a database which is a key support of any software project.

A huge amount of data is generated and stored in a structured and unstructured format. Database schemas are a structured method of storing the data. Even though the data is structured, there exist multiple schemas and tables which are related, but the relations cannot be established by simple databases. One of the methods to represent schemas as a combined knowledge source is to design a knowledge graph (KG). The term “KG” was known to be used in writing since no less than 1973 in the paper presented by Edward W. Schneider. Google created its own knowledge graph [1], which was announced in 2012, is a more sophisticated embodiment of the knowledge graph ever seen in the technology world [2]. Following then, knowledge graphs have become more important in terms of research and application. As a consequence of the research that took place, quite a few KG products were developed. Cyc [3], Freebase [4], Google knowledge vault [5], DBpedia [6], YAGO [7], NELL [8], PROSPERA [9], Microsoft’s Probase [10] are among the prominent KGs ever created. Many real-world applications like Semantic Web search engine [11], named entity recognition and disambiguation [12], [13], Extracting Information [14], [15], Recommendation systems [16], and Siri [17], Apple's personal assistant, Watson [18] from IBM also make use of generic knowledge graphs. Even in the field of medicines, KG called as Medical KG [19] is gaining lot of attention recently.

Graph neural network (GNN) is also one of the techniques that are used in graph applications, the input method in this also works the same as that of the neural network system, but the nodes are converted into graph embeddings which is an important step involved in any GNN model. The main reason behind using GNN would be to be able to predict relationships(edges) between completely unrelated nodes, technically known as Edge prediction [20], [21]. Zhao *et al.* [22], discusses KG's construction and architecture. It also gives an outline of KG's construction and looks at the approaches and difficulties that were encountered during the process. Building KG requires Semantic relatedness which plays a key role in linking a word with an existing word in our knowledge graph. It is a quantitative measure of how two words or concepts are related under a context, without any discrepancy of the syntactical form of the word. Conceptually speaking, semantic relatedness is based on functional relations like hyponymic, hypernymic, metronymic [23]. At the point when restricted to hyponymy/hypernymy relations, the action evaluates semantic similarities between two words or concepts.

This paper discusses a method to build a robust knowledge graph and query the same to build a database by suggesting tables and columns in an automated way. The proposed system is designed to be based on a knowledge graph, where it creates a smaller graph for each of the nuclear schemas and then merges this graph with the global KG intelligently based on the similarity between the nodes by comparing their word vectors. So, the proposed algorithm not only suggests database schemas but also serves as an efficient tool to store information while minimizing redundancy.

2. METHOD

This section describes the approach that was applied to produce a knowledge graph of different entities. The workflow of the pipeline is shown in Figure 1. In short, the framework includes the following steps:

- Schema to knowledge graph: convert each of the schemas to the knowledge graph
- Executing the Algorithm: which exploits the link prediction using multiple natural language processing and machine learning tools.
- Storing the data: in which the final knowledge graph is stored in the TinkerPop database.
- Interaction with the System: in which multiple questions are asked to the user to get complete insights.
- Query refining and output: in which keywords are extracted using natural language processing tools and the final schema is given to the user.

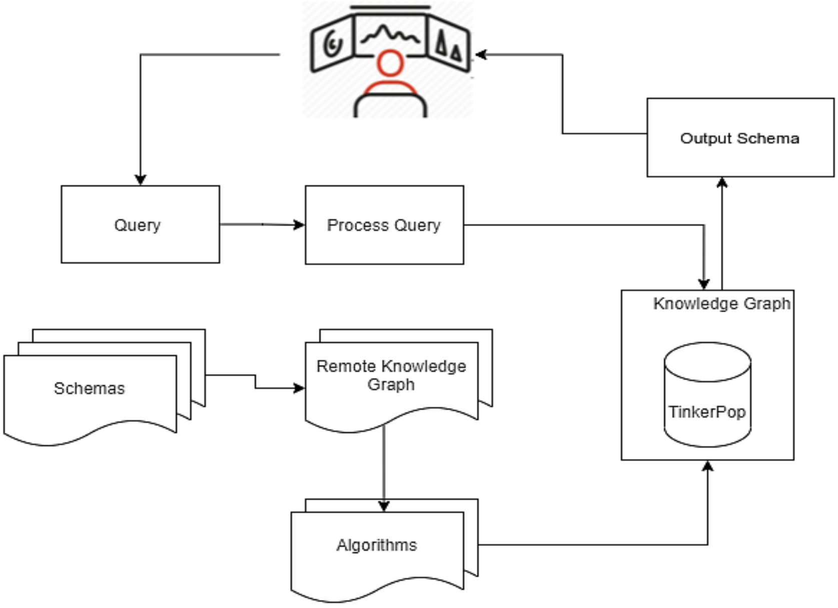


Figure 1. Shows the flowchart of the AI-based models and experimental methods applied

2.1. Schema to knowledge graph

Given the multiple schemas, the first step is to convert the given schema into a knowledge graph. Firstly, each table is converted to a node [24], table attributes are added as the properties of node and relation as an edge for the graph and a complete knowledge graph is built. Table attributes are converted to properties of nodes so that it is easier to keep the properties of the original schema intact.

2.2. Executing the algorithm

This is the core step of the complete architecture. For the algorithm as shown in Table 1 to work at least, two knowledge graphs are required. For each node *n1* in graph *G1*, iterate over each node *n2* of graph *G2.* Using the synset similarity [25] function (from NLTK library) the percentage of similarity is calculated between the node *n1* and *n2,* if the similarity is 100% (this mean both the nodes are the same) both the nodes are merged to form a single node otherwise if the similarity percentage is more than the threshold value, a new node *n* is created i.e., the hypernym of both nodes. Once the new node is created it is then connected to the node *n1* and *n2* by adding new edges from node *n1* to *n* and from *n2* to *n*.

Table 1. Algorithm to merge two schemas based on similarity

```
Algorithm 1: Join Graph
Data: G1: graph1, G2: graph2
Result: single merged graph
for each node n1ϵ nodes(G1) do
    for each node n2ϵ nodes(G2) do
        if similarity(n1,n2) == 1 then
            combine both nodes to one;
        else
            if similarity (n1,n2) ≥ thresholdvalue then
                create-new-node(n);
                add-edge(n1,n);
                add-edge(n2,n);
            end
        end
    end
end
```

2.3. Storing the data

Once the final knowledge graph is generated, it is then stored in a graph database i.e., Tinker Pop. After storing the knowledge graph in the database, the server is started and now the knowledge graph is ready to query. As of now, Gremlin query language [26] has been used to query the graph.

2.4. Interaction with the system

Users interact with the system through a chatbot and a sequence of questions is asked to get a complete insight into what a user wants. Based on the insights the system will provide the possible nodes (tables) to choose from. Once the nodes (table) are selected, the final schema is given to the user with all the relevant attributes and relations.

2.5. Query refining and output

Once the queries are received from the user, it is then processed to remove the unwanted words like articles, propositions, quantifiers. After cleaning the query, the final keyword is extracted, now this keyword will be used to extract the final sub-knowledge graph. After extracting the sub-knowledge graph, it is then converted into the schema so that it can be easily used by the user.

Using an example, the complete proposed system is illustrated in the following steps. Two schemas are considered i.e., School and student schema,

Step1: Converting schema to knowledge graph

The first step converts the schema into a knowledge graph as shown in Figures 2-3, where node name is the table name and the attributes are stored in the properties of a node. Edges of the graphs contain the properties which also denotes the relationships between the tables. Now these knowledge graphs will be fed to the algorithm to generate the final knowledge graph.

Step2: Executing the algorithm on knowledge graphs

In this step, previously generated knowledge graphs are fed to the algorithm that predicts the new edges and nodes based on the similarity between two nodes (table) and if two nodes are similar then their properties (attributes) are merged to get a single node. The final knowledge is generated as shown in Figure 4. The final knowledge graph contains much more context than the individual knowledge graphs.

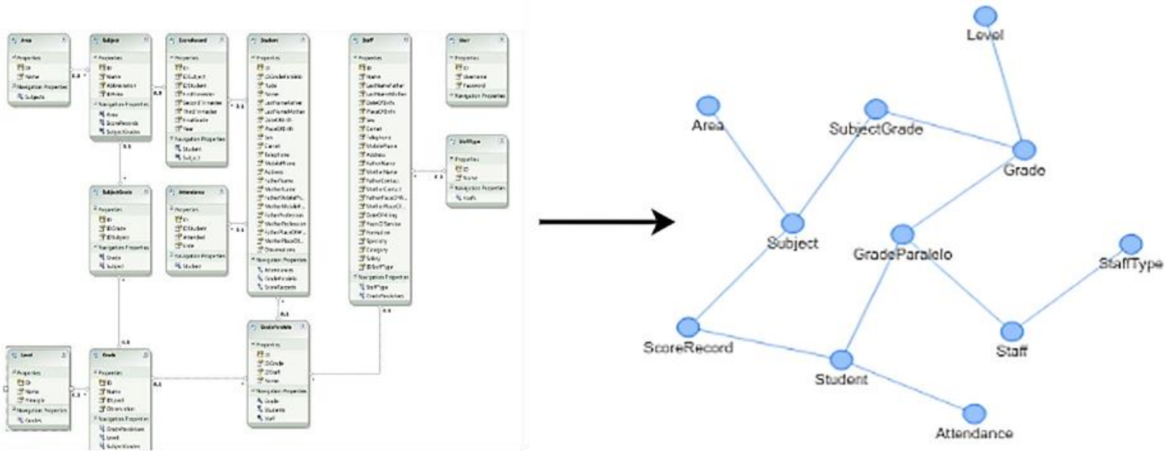


Figure 2. School schema to a knowledge graph

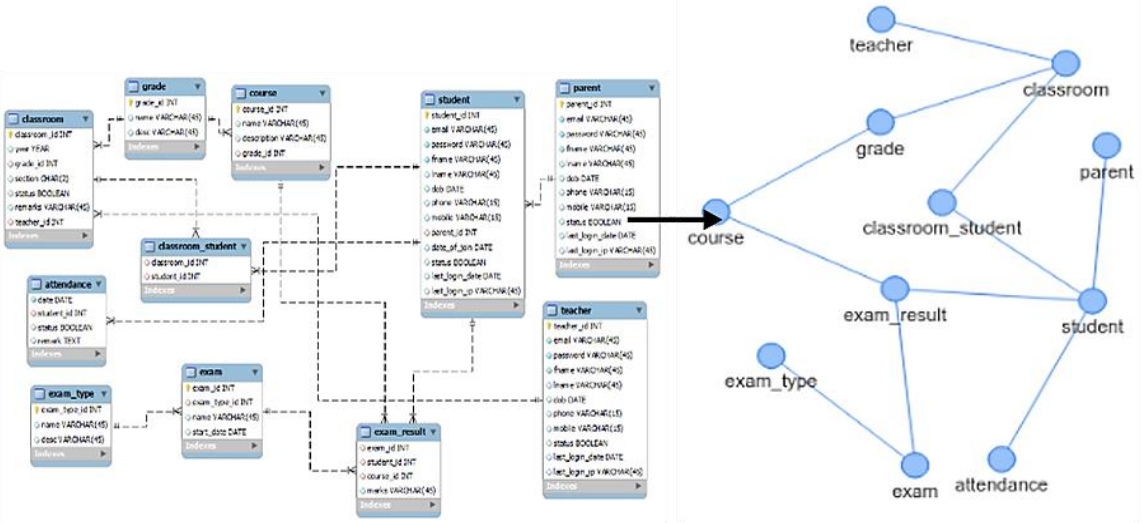


Figure 3. Student schema to a knowledge graph

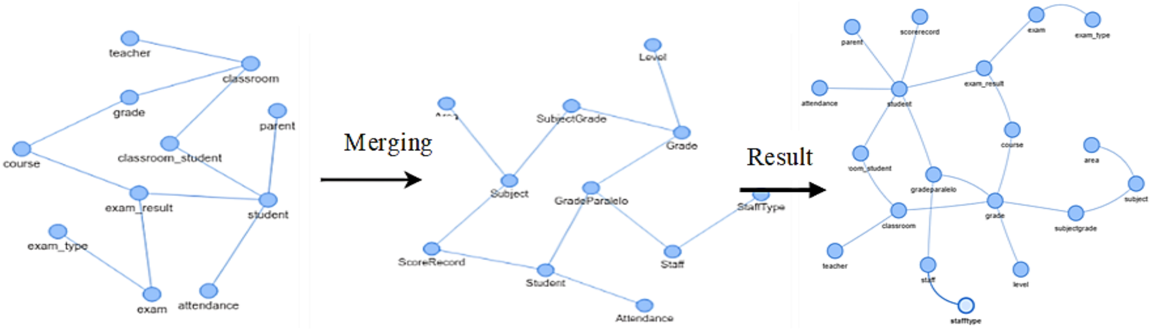
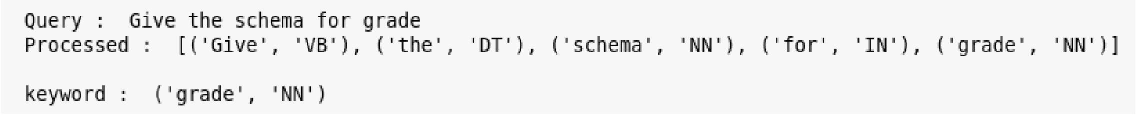


Figure 4. Merging student and school schema and result is a combined knowledge graph

Step3: Query and preprocessing

In this step, the user-query is processed using the natural language processing modules that split the sentences into verbs, nouns and determiners. The irrelevant words (like: articles, preposition) are removed, to get only the essential words. Out of these useful words the keyword is extracted out as shown in Figure 5.



```
Query :  Give the schema for grade
Processed :  [('Give', 'VB'), ('the', 'DT'), ('schema', 'NN'), ('for', 'IN'), ('grade', 'NN')]

keyword :  ('grade', 'NN')
```

Figure 5. Grade query processing

Step4: Output schema

This is the final step of the process, where the keywords extracted in step3 are used to extract the subgraph over the final knowledge graph that is generated in step2. Depending on the level of details a user wants, the knowledge graph is generated as shown in Figure 6. Based on the output graph, the schema is generated with all the attributes.

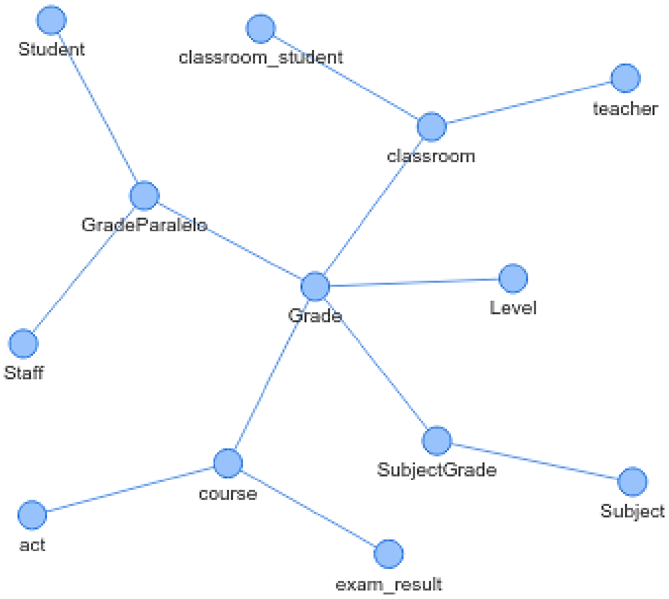


Figure 6. Output grade schema with details up to level 2 when queried over the merged graph

3. RESULTS AND DISCUSSION

This section briefly introduces some illustrative queries and results representing typical schema. The dataset that was used for experimenting has more than 10 schemas from different domains (like: school, employee, sales, registration, and library). The graph in Figure 7 shows the final graph obtained by merging all the nuclear graphs according to the algorithm. As we keep on adding the more schemas this graph keeps on growing.

Input Query:

For the query “Give the schema for students” the query was first preprocessed and the keyword was extracted i.e. “Student” shown in Figure 8. After the keyword is extracted, it is then used to extract out the subgraph from the main knowledge graph as shown in Figure 7. From the output schema shown in Figure 9 it can be seen that the student node has more than 10 directly connected nodes, this shows how much-enriched details are provided by the schema.

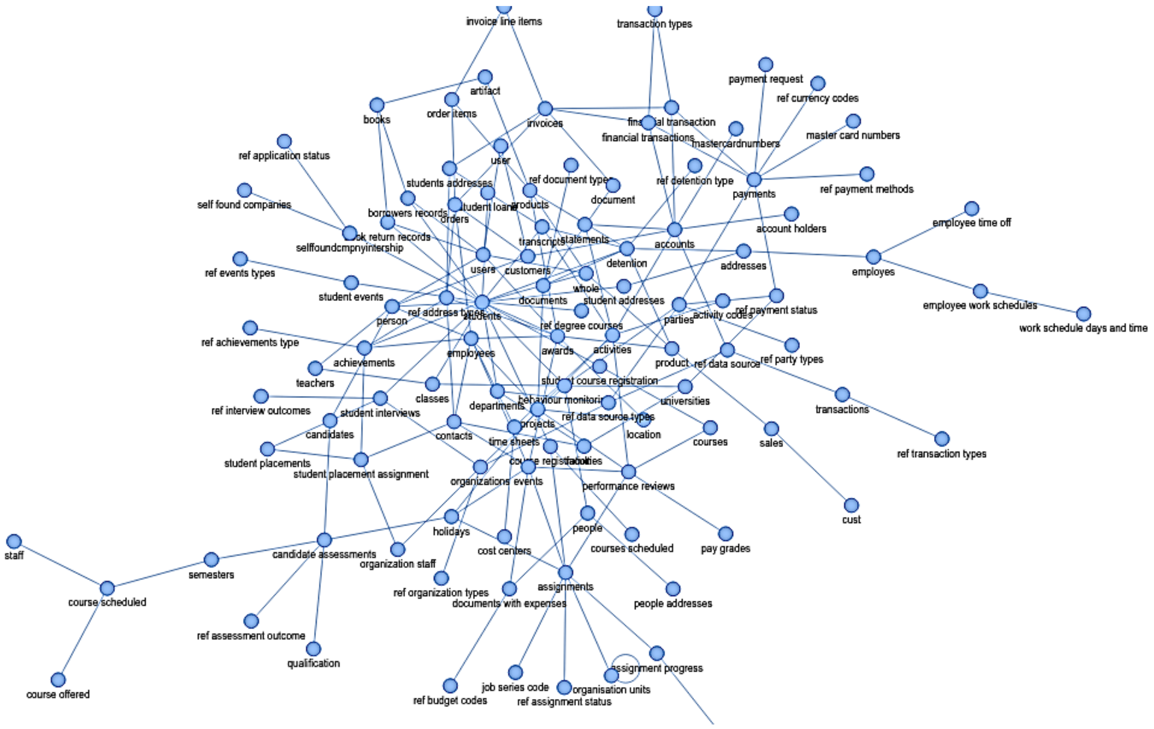
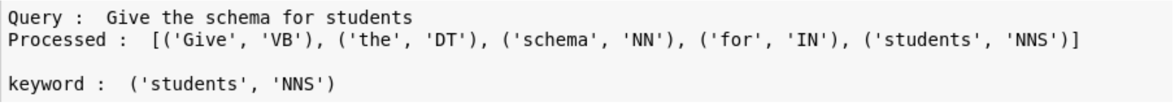


Figure 7. Knowledge graph after merging 11 schemas



```
Query :  Give the schema for students
Processed :  [('Give', 'VB'), ('the', 'DT'), ('schema', 'NN'), ('for', 'IN'), ('students', 'NNS')]

keyword :  ('students', 'NNS')
```

Figure 8. Query for student schema

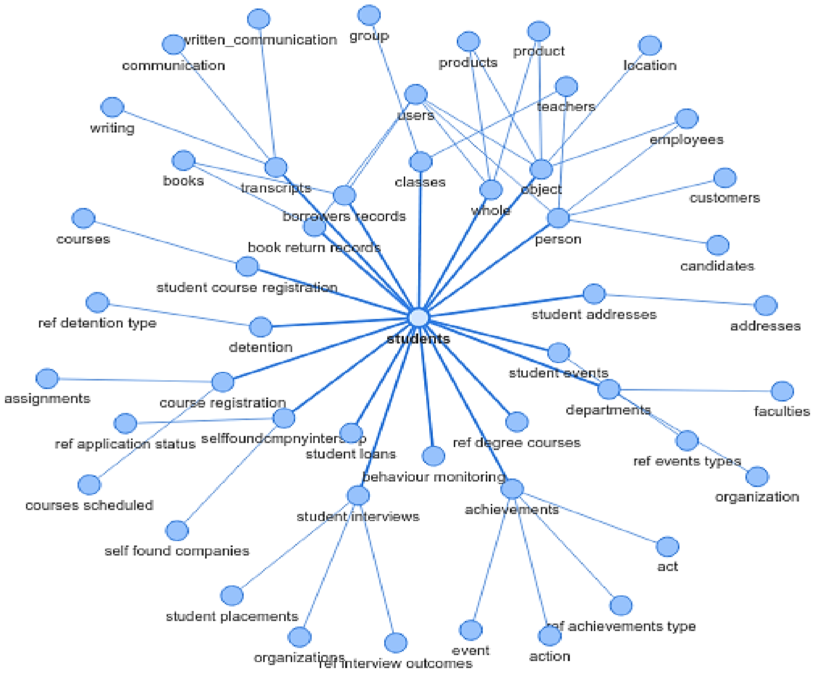
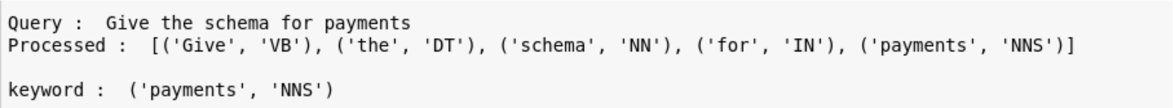


Figure 9. Student schema when queried over the merged graph

For the query “Give the schema for payments” as shown in Figure 10, the query is processed to remove the unwanted words, and the keyword extracted is “payments”. Using this keyword subgraph is extracted from the merged graph as shown in Figure 7. The subgraph generated has details up to depth 2 as shown in Figure 11.



```
Query :  Give the schema for payments
Processed :  [('Give', 'VB'), ('the', 'DT'), ('schema', 'NN'), ('for', 'IN'), ('payments', 'NNS')]

keyword :  ('payments', 'NNS')
```

Figure 10. Query for payments

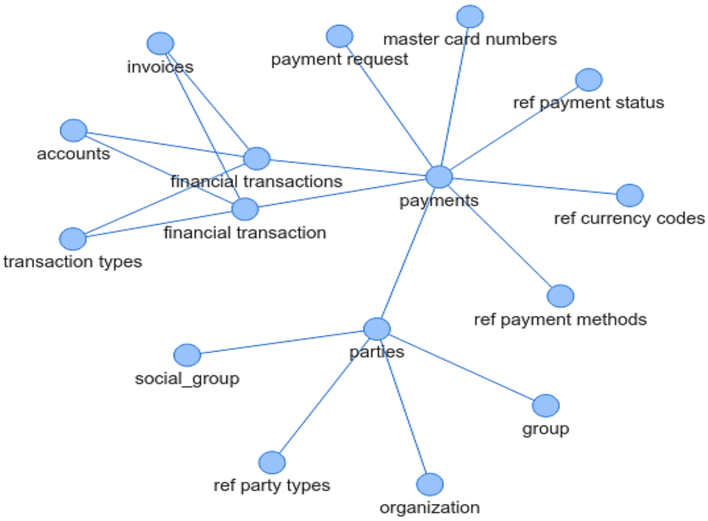


Figure 11. Payments schema when queried over the merged graph

4. CONCLUSION

Technological advancements in software industries have led to an increase in the usage of data storage systems, especially, relational database systems are being extensively used. Designing a database schema takes a considerable share of total development time in any project. Hence, there is a need for automating the entire process of designing a database schema. In this paper, various approaches were discussed to solve the problem and it was evident that the knowledge graph-based approach suits best for the purpose. Here, usage of KG was investigated to dynamically generate schemas that satisfy all the constraints discussed in the paper and also reduces the redundancy of the data in the knowledge graph. The existing KG built from various schemas is queried to suggest schemas to the user. The proposed model was tested with multiple schemas to generate the global knowledge graph and it was queried and verified manually. This method helps the user in developing relationships between given schemas to draw the KG and query the same to suggest a schema.

REFERENCES

[1] E. W. Schneider, “Course modularization applied: the interface system and its implications for sequence control and data analysis,” *Association for the Development of Instructional Systems (ADIS)*, pp. 10–73, 1973, doi: 10.1037/e436252004-001.

[2] L. Ehrlinger and W. Wöß, “Towards a definition of knowledge graphs,” in *Conference: Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16)*, 2016, vol. 1695, pp. 1–4.

[3] B. L. Douglas, “CYC: A Large-Scale Investment in Knowledge Infrastructure,” *Communications of the ACM*, vol. 38, no. 11, pp. 33–38, 1995, doi: 10.1145/219717.219745.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1247–1249, 2008, doi: 10.1145/1376616.1376746.

[5] X. Dong *et al.*, “Knowledge vault: A web-scale approach to probabilistic knowledge fusion,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 601–610, 2014, doi: 10.1145/2623330.2623623.

[6] J. Lehmann *et al.*, “DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia,” *Semantic Web*, vol. 6,

no. 2, pp. 167–195, 2015, doi: 10.3233/SW-140134.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” *16th International World Wide Web Conference, WWW2007*, pp. 697–706, 2007, doi: 10.1145/1242572.1242667.

[8] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, “Toward an architecture for never-ending language learning,” *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, pp. 1306–1313, 2010.

[9] N. Nakashole, M. Theobald, and G. Weikum, “Scalable knowledge harvesting with high precision and high recall,” *Proceedings of the 4th ACM International Conference on Web Search and Data Mining, WSDM 2011*, pp. 227–236, 2011, doi: 10.1145/1935826.1935869.

[10] W. Wu, H. Li, H. Wang, and K. Q. Zhu, “Probase: a probabilistic taxonomy for text understanding,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 481–492, 2012, doi: 10.1145/2213836.2213891.

[11] G. Sudeepthi, G. Anuradha, and M. S. P. Babu, “A survey on semantic web search engine,” *International Journal of Computer Science*, vol. 9, no. 2, pp. 241–245, 2012. Accessed: 20-Aug-2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.482&rep=rep1&type=pdf.

[12] D. Damljanovic and K. Bontcheva, “Named entity disambiguation using linked data,” *Proceedings of the 9th Extended Semantic Web Conference (ESWC 2012), Poster session*, pp. 231–240, 2012. Accessed: 20-Aug-2021. [Online]. Available: http://2012.eswc-conferences.org/sites/default/files/eswc2012_submission_334.pdf.

[13] X. Han and J. Zhao, “Named entity disambiguation by leveraging wikipedia semantic knowledge,” *International Conference on Information and Knowledge Management, Proceedings*, pp. 215–224, 2009, doi: 10.1145/1645953.1645983.

[14] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, “Knowledge-based weak supervision for information extraction of overlapping relations,” *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 541–550, 2011.

[15] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, “Improving efficiency and accuracy in multilingual entity extraction,” *ACM International Conference Proceeding Series*, pp. 121–124, 2013, doi: 10.1145/2506182.2506198.

[16] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, “Knowledge graph convolutional networks for recommender systems,” *The Web Conference 2019-Proceedings of the World Wide Web Conference, WWW 2019*, pp. 3307–3313, 2019, doi: 10.1145/3308558.3313417.

[17] Apple, “Apple Siri,” 2016. [Online]. Available: https://www.apple.com/ios/siri/. (Accessed Aug. 24, 2021).

[18] “IBM Watson,” 2017. [Online]. Available: https://www.ibm.com/watson/. (Accessed Aug. 24, 2021).

[19] L. Li *et al.*, “Real-world data medical knowledge graph: construction and applications,” *Artificial Intelligence in Medicine*, vol. 103, 2020, doi: 10.1016/j.artmed.2020.101817.

[20] J. L. K. Xu, W. Hu and S. Jegelka, “How powerful are graph neural networks,” *Proc. ICLR*, pp. 1–17, 2019.

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021, doi: 10.1109/TNNLS.2020.2978386.

[22] Z. Zhao, S.-K. Han, and I.-M. So, “Architecture of knowledge graph construction techniques,” *Mathematics, International Journal of Pure and Applied*, vol. 118, no. 19, pp. 1869–1883, 2018.

[23] M. Strube and S. P. Ponzetto, “WikiRelate! Computing semantic relatedness using Wikipedia,” *AAAI*, vol. 6, pp. 1419–1424, 2006.

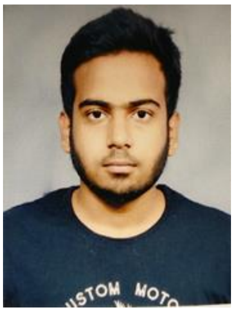
[24] O. Alotaibi and E. Pardede, “Transformation of Schema from Relational Database (RDB) to NoSQL Databases,” *Data*, vol. 4, no. 4, p. 148, Nov. 2019, doi: 10.3390/data4040148.

[25] P. Merlo, J. Henderson, G. Schneider, and E. Wehrli, “Learning document similarity using natural language processing,” *Linguistik Online*, vol. 17, no. 5, Dec. 2003, doi: 10.13092/lo.17.788.

[26] M. A. Rodriguez, “The gremlin graph traversal machine and language (Invited Talk),” *DBPL 2015 - Proceedings of the 15th Symposium on Database Programming Languages*, pp. 1–10, 2015, doi: 10.1145/2815072.2815073.

BIOGRAPHIES OF AUTHORS

Priyank Kumar Singh is currently pursuing his Bachelor's degree in Computer Science and Engineering at R.V College of Engineering. He completed his schooling and Pre-University degree at St Mary's Academy, Saharanpur, India. His areas of interest include Natural Language Processing, Big Data analysis, Web development, and Python Development. He can be contacted at email: priyankkumars.cs18@rvce.edu.in



Sami Ur Rehman is currently pursuing his Bachelor's degree in Computer Science and Engineering at R.V College of Engineering. He completed his schooling at Iqra High School and his Pre-University degree at M.E.S Chaitanya P.U. college. His areas of interest include Artificial Intelligence, Natural Language Processing, Cybersecurity, and Web development. He can be contacted at email: samiurrehman.cs18@rvce.edu.in

Darshan J. is currently pursuing his Bachelor's degree in Computer science at R.V College of Engineering. He has completed his schooling at Evershine English School and his PU degree at R.V PU college, Bengaluru in the year 2019. His areas of interest include IoT, Web development, and Machine learning. He can be contacted at email: darshanj.cs19@rvce.edu.in

Shobha G. Professor, Computer Science, and Engineering Department, R.V College of Engineering, Bengaluru, India has teaching experience of 26 years, her specialization includes Data mining, Machine Learning, and Image processing. She has published more than 150 papers in reputed journals/conferences. She has also executed sponsored projects worth INR 200 lakhs funded by various agencies nationally and internationally. She is a recipient of various awards such as the Career Award for young teachers 2007-08 constituted by All India Council of Technical Education, Best Researcher award from Cognizant 2017, GHC Faculty Scholar for Women in Computing in 2018, IBM Shared University Research Award in 2019, HPCC Systems community recognition award 2020. She is also an advisory committee member for IET India Scholarship Award 2021. She can be contacted at email: shobhag@rvce.edu.in

Deepamala N. Associate Professor, Computer Science, and Engineering Department, R.V College of Engineering, Bengaluru, India has Industry experience of 5 years and teaching experience of 11 years. She has worked in MNC like D-Link, Nortel Networks, and Radware before pursuing a teaching career. Her specialization is in Natural Language Processing and Networking. She has one patent in her name and has 20 publications in various reputed journals. She has executed consultancy and R&D projects as PI and Co-PI. She has published a chapter in Elsevier’s book. She can be contacted at email: deepamalan@rvce.edu.in