


A Comprehensive Approach for Graph Data Warehouse Design: A Case Study for Learning Path Recommendation Based on Career Goals

Viet Nguyen^{1,2}, Vu Bui^{1,2} and Thu Nguyen^{1,2} ^{a,b}

¹Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam

²Viet Nam National University, Ho Chi Minh City, Vietnam

{20127664, 20127668}@student.hcmus.edu.vn, ntmtthu@fit.hcmus.edu.vn

Keywords: Data Warehouse, NoSQL Databases, Graph Data Warehouse, NoSQL-Based Data Warehouse Modeling, Methodology.

Abstract: Today, organizations leverage big data analytics for insights and decision-making, handling vast amounts of structured and unstructured data. Traditional data warehouses (TDW) are suboptimal for such analytics, creating a demand for NoSQL-based modern data warehouses (DWs) that offer improved storage, scalability, and unstructured data processing. Graph-based data models (GDMs), a common NoSQL data model, are considered the next frontier in big data modeling. They organize complex data points based on relationships, enabling analysts to see connections between entities and draw new conclusions. This paper provides a comprehensive methodology for graph-based data warehouse (GDW) design, encompassing conceptual, logical, and physical phases. In the conceptual stage, we propose a high-abstraction data model for NoSQL DW, suitable for GDM and other NoSQL models. During the logical phase, GDM is used as the logical DW model, with a solution for mapping the conceptual DW model to GDW. We illustrate the GDW design phases with a use case for learning path recommendations based on career goals. Finally, we carried out the physical implementation of the logical DW model on the Neo4j platform to demonstrate its efficiency in managing complex queries and relationships, and showcase the applicability of the proposed model.

1 INTRODUCTION


The proliferation of social networks, cloud computing, and IoT devices has precipitated the generation of vast amounts of data, heralding the era of 'big data.' This surge in data volume presents significant challenges for TDW systems, which increasingly struggle with scalability and the management of unstructured data Oussous et al. (2017). Although once efficient, TDWs are now inadequate in addressing the growing demands for speed and complexity in data processing Bhogal and Choksi (2015). In response to these limitations, NoSQL databases have emerged as a critical solution, offering enhanced flexibility, scalability, schemalessness, high availability, and cost-effectiveness in data storage and analysis Bhogal and Choksi (2015).

Among the various NoSQL data models, GDMs stand out for their capability to manage large-scale data and represent complex relationships, making

them particularly suited for applications such as recommendation systems and social networks Akid and Ayed (2016); Sellami et al. (2019). Consequently, a growing number of researchers and organizations are transitioning from Relational Online Analytical Processing (ROLAP) to NoSQL DW, despite the technical challenges that accompany this shift Banerjee et al. (2021). This transition necessitates not only technological adaptations but also a thorough design process that spans from conceptual to physical models Abdelhédi et al. (2017); Banerjee et al. (2021).

Nevertheless, a significant research gap persists in the absence of a unified conceptual model specifically designed for NoSQL DW Banerjee et al. (2021). Addressing this gap is crucial for ensuring the effective design and efficient implementation of NoSQL DW across various organizational contexts. This study contributes to the field by proposing:

- A novel symbolic system for conceptual modeling in NoSQL DW, building on prior research Banerjee et al. (2021).
- A comprehensive NoSQL DW design methodol-

^a  <https://orcid.org/0009-0009-6961-3976>

^b Corresponding author

ogy, with a particular focus on GDM.

- A case study that demonstrates the application of the proposed system within GDW tailored for learning path recommendations.

The paper is structured as follows: **Section 1** introduces the context and objectives of the study. **Section 2** reviews the relevant literature. **Section 3** presents the symbolic system and the proposed conceptual DW model. **Section 4** elaborates on the conversion process to GDM, supported by a case study. **Section 5** discusses deployment results and performance analysis. Finally, **Section 6** concludes the paper and explores future research directions.

2 RELATED WORKS

In NoSQL DW research, implementations have predominantly focused on the logical level, targeting specific NoSQL types. Yangui et al. (2016) proposed a method for transforming multidimensional conceptual models into column- and document-oriented NoSQL models. Chevalier et al. (2015) introduced three conversion strategies for document-oriented models, and their study rigorously evaluated the performance and memory utilization of these strategies.

Recent studies highlight the superior performance of GDW compared to traditional DWs. For instance, Akid et al. (2022) demonstrated GDW's efficiency in handling OLAP queries, while Nguyen et al. (2022) explored the practical applications of GDM in complex scenarios, including learning path consultancy.

However, as Banerjee et al. (2021) points out, there remains a significant gap due to the absence of a unified conceptual model for NoSQL DW. Their proposed symbolic system and model, despite its novel approach, has constraints, particularly in depicting many-to-many relationships and defining dimension table attributes.

Building on these studies, our research proposes a novel symbolic system for constructing a conceptual DW model for NoSQL DW, applicable across various NoSQL databases, with a focus on GDM. We also offer transformation rules and a case study to illustrate the practical application, culminating in a physical implementation on the Neo4j platform.

3 A SYMBOL SYSTEM FOR CONCEPTUAL DW MODELING

This section introduces a set of symbols and rules for conceptual NoSQL DW modeling, using Crow's

Foot notation for relationships and tabular symbols for facts and dimensions, ensuring clarity in representing attributes of fact and dimension tables.

Before presenting the symbol system, fundamental concepts from the multidimensional model, as detailed by Sellami et al. (2019), are adapted. This model focuses on a business-centric view of data, emphasizing facts, dimensions, measures, and their inter-relationships.

3.1 Conceptual Multidimensional Model

A conceptual multidimensional model denoted MDM is defined by the triplet $(FTs, DTs, Star^{MDM})$, where:

- $FTs = \{FT_1, FT_2, \dots, FT_n\}$ is a set of fact tables.
- $DTs = \{DT_1, DT_2, \dots, DT_m\}$ is a set of dimension tables.
- $Star^{MDM} : F^{MDM} \rightarrow 2^{DTs}$ is a function that associates each fact table $FT_i \in FTs$ with the set of dimension tables $DT_i \in DTs$, and 2^{DTs} is a set of all combinations of the dimension tables.

A fact table, denoted $FT_i \in FTs$, is defined by the pair $(F_{name}, F_{measures})$, where:

- F_{name} represents the name of the fact table.
- $F_{measures} = \{m_1, m_2, \dots, m_k\}$ is the set of measures of the fact table.

A dimension table, denoted $DT_i \in DTs$, is defined by the pair $(D_{name}, D_{attributes})$, where:

- D_{name} represents the name of the dimension table.
- $D_{attributes} = \{a_1, a_2, \dots, a_k\}$ is a collection of dimension table attributes.

3.2 The Set of Symbols for Conceptual NoSQL DW Representation

In this section, we propose a refined set of symbols, building upon previous research of Banerjee et al. (2021), to effectively represent the conceptual model of a NoSQL DW. This enhanced set of symbols aim to address and overcome the limitations identified in earlier studies, providing a robust framework for modeling DW at the conceptual level.






3.2.1 The Refined Set of Symbols

The Crow's Foot notation is widely recognized for its simplicity and clarity in Entity-Relationship (ER) modeling, making it particularly effective for conceptual database representation Puja et al. (2019). This notation intuitively depicts relationships, including

many-to-many connections, using Crow’s Foot symbols, while tabular symbols represent facts and dimensions along with their attributes and measures. This dual-notation approach ensures clear expression of fact and dimension table attributes, as summarized in **Table 1**.

The conceptual model for a NoSQL DW will be constructed using these symbols, with definitions of model components and relationships detailed in the next section.

Table 1: Proposed symbol system based on Banerjee et al. (2021).

Shape	Symbol	Significance
Rectangle		Components within the Collection layer
Tabular		Table of facts or table of dimensions
Ellipse		The optional attribute for a dimension table
Arrow		Indicates a relationship in which a component of one layer encapsulates a component of another layer
Crow’s Foot Notation		Indicates the relationship between a dimension table and a fact table, or a dimension table and another dimension table

3.2.2 Conceptual NoSQL DW Model

Consistent with the symbol set presented in Table 1, this section presents a conceptual DW model specifically designed for NoSQL DW that can be applied to different kinds of NoSQL data models. Particular guidelines for conceptual information representation are included in this model. In-depth information about the elements and how they interact with one another in the model will also be supplied. The suggested conceptual DW model is shown in Figure 1, which provides a clear and organized representation of these elements and their relationships. The model is explained in depth in the following:

Components of the Model

This section presents comprehensively information regarding the components of the model, which is structured into three distinct layers, namely the Collection layer, Family layer, and Optional attribute layer.

- a. **Collection Layer (CL):** The topmost layer of the model is defined by its primary component, col. CL comprises one or more col:

$$CL = \{col_1, \dots, col_q\}$$

- b. **Family Layer (FL):** The middle layer of the conceptual DW model is represented by this. The field of FL encompasses two distinct categories, namely Fact Family (FF) and Dimension Family (DF). Fact tables and dimension tables are the primary components of FF and DF, respectively.

- **Fact Family (FF):** The components of FF consist of fact tables (FT), denoted as:

$$FF = \{FT_1, \dots, FT_n\}$$

- **Dimension Family (DF):** The components of DF consist of dimension tables (DT), denoted as:

$$DF = \{DT_1, \dots, DT_n\}$$

- c. **Optional Attribute Layer (OAL):** This layer is the last layer of the model and includes optional attributes that can be present or absent between instances of dimension tables $DT \in DF$. The key component of this class is OAT, which is defined by several pairs (D_{name}, OA) , where:

- D_{name} represents the name of DT.
- OA is the optional attribute name of the DT

$$OAT = \{(D_{name_1}, OA_1), \dots, (D_{name_k}, OA_k)\}$$

Relationship

This section will deal with the relationships between the components identified in the above section. Relationships within the model can be classified into two fundamental types: internal and external relationships. Internal relationships indicate relationships inside the same component, whereas external links illustrate interactions between distinct components.

External Relationship: According to Figure 1, these following relationships are considered external relationships:

- a) **Component FF – Component DF (FactDim-Rel):** the FactDimRel relationship is a mapping $f : FF \rightarrow \mathcal{P}(DF) \setminus \{\emptyset\}$ where $\mathcal{P}(DF)$ is the power set of DF . In this relationship, each $FT \in FF$ is

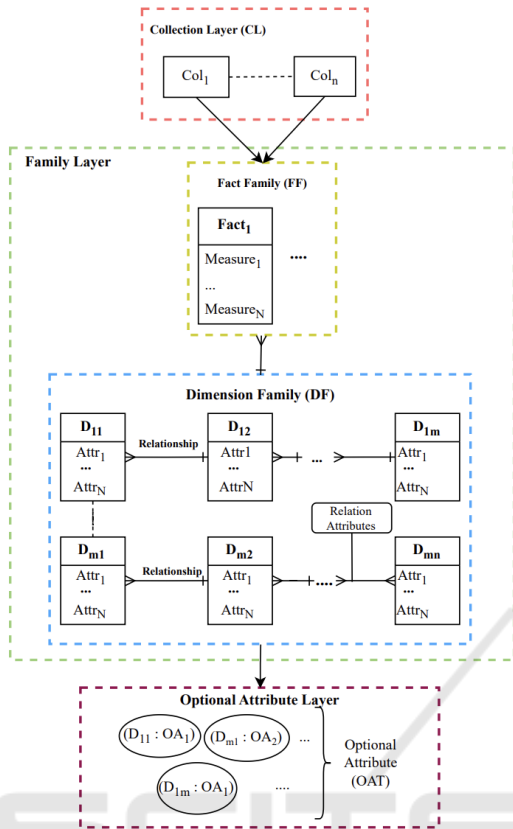


Figure 1: Proposed conceptual DW model for NoSQL DW.

mapped to a non-empty collection of dimension tables $DT_s \subseteq DF$. This can be represented by the function f as follows:

$$f(FT) = DT_s \text{ with } DT_s = \{DT_1, \dots, DT_m\}$$

- b) **Component DF – Component OAT (DimOAttrRel):** The DimOAttrRel relationship is a mapping $f : DF \rightarrow \mathcal{P}(OAT)$ where $\mathcal{P}(OAT)$ is the power set of OAT . In the DimOAttrRel relationship, each $DT(D_{name}, D_{attributes}) \in DF$ is mapped to a set of optional attributes $oat \subseteq OAT$. This can be represented by the function f as follows:

$$f(DT) = oat, \text{ where } oat = \{(D_{name}, OA_1), (D_{name}, OA_2), \dots, (D_{name}, OA_m)\}$$

- c) **Component Col – Component FF (CLFFRel):** The CLFFRel relationship is a mapping $f : Col \rightarrow \mathcal{P}(FF) \setminus \{\emptyset\}$ where $\mathcal{P}(FF)$ is the power set of FF . In the CLFFRel relationship, each $Col \in CL$ is mapped to a non-empty set of fact tables $FT_s \subseteq FF$. This can be represented by the function f as follows:

$$f(Col) = FT_s \text{ with } FT_s = \{FT_1, \dots, FT_m\}$$

Internal Relationships: According to Figure 3.1, internal relationships occur within the DF component. There are two types: hierarchical relationships (many-to-one) and semantic relationships (many-to-many).

- a) **Hierarchical Relationships (HierarRel):** Representing many-to-one relationships between two dimensions DT_1 and DT_2 of DF , this is a mapping: $f : DT_1 \rightarrow DT_2$ such that each element in DT_1 (child) is mapped to a unique element in DT_2 (parent).

$$f : DT_1 \rightarrow DT_2$$

$$\forall y \in DT_1, \exists! x \in DT_2 \text{ such that } f(y) = x.$$

- b) **Semantic Relationships (SemantRel):** Representing many-to-many relationships between two dimensions DT_1 and DT_2 of DF . To represent this SemantRel, an intermediate structure M is required. In this structure:

- M is the intermediate set containing pairs of mappings between elements in DT_1 and DT_2 along with the set of attributes RA derived from their relationship.
- $M \subseteq DT_1 \times DT_2 \times \mathcal{P}(RA)$ where $\mathcal{P}(RA)$ is the power set of the set of attributes derived from RA .
- Each element of M can be represented as a tuple (d_1, d_2, ra) where $d_1 \in DT_1, d_2 \in DT_2$ and $ra \subseteq RA$.

4 CONVERTING CONCEPTUAL NOSQL DW MODEL TO GRAPH-BASED LOGICAL MODEL RULES

In this section, we propose rules to convert the conceptual DW model to a logical DW model, which uses a graph-based data model. Our rules are stated based on Sellami et al. (2019), a study about rules for mapping a multidimensional model to a graph-based logical DW model. Before delving into the conversion rules, we will mention the foundation of a graph-based data model, laying the groundwork for the conversion.

4.1 Graph-Based Data Model Foundation

Information about the concept of the GDM is inherited from reference Sellami et al. (2019), with

some adjustments for higher clarity and consistency. Specifically:

A graph-oriented database G is defined by (V^G, E^G, P^G) , where:

- $V^G = \{V_1, V_2, \dots, V_n\}$ is the set of nodes.
- $E^G = \{E_1, E_2, \dots, E_m\}$ is the set of edges denoting relationships between nodes.
- P^G is the set of nodes' properties in the graph.

An attribute is defined by a key-value pair.

Node: A node V consists of properties and labels, identified by (id^V, P^V, L^V) , where:

- id^V is the identifier of the node.
- P^V is the set of properties of the node.
- L^V is the set of labels of the node, each label l shows meaning to the node, helping in identifying the roles for each node.

Relationship: A relationship R is defined as a link connecting two nodes, which represent two entities that interact or relate to one another. Each relationship might contain more information through additional properties. A relationship can be identified by $(id^R, sourceNodeID, targetNodeID, T^R, P^R)$, where:

- id^R is the identifier of the relationship.
- $sourceNodeID$ is the identifier of the source node.
- $targetNodeID$ is the identifier of the target node.
- T^R denotes the relationship type, helping determine which relationship is linking the two nodes.
- P^R is a set of properties representing additional information about it.

4.2 Rules for Mapping Components

We will offer the conversion rules in the respective components, taking into account the attributes of the components given in the suggested conceptual DW model and the symbols of the graph-based data model described above. There are two main groups into which these rules fall: relationships and entities. Here, we will offer greater detail about these guidelines.

4.2.1 Rules for Mapping Entities

Rule 1: Mapping col and FT. Each fact table $FT(F_{name}, F_{measures})$ of the specified conceptual DW (via $CLFRFel$ relationship) will become a node $V(id^V, P^V, L^V)$ where:

- Label l_1 represents the table type: $l_1 = \{fact\}/L^V = \{l_1\}$

- Label l_2 represents the table name: $l_2 = F_{name}/L^V = L^V \cup l_2$

- Label l_3 represents the related collection name: $l_3 = col/L^V = L^V \cup l_3$ (optional)

- Each measure $m_i \in F_{measures}$ will be converted into a property $p \leftarrow m_i/P^V = P^V \cup p$

Rule 2: Mapping DT. Each dimension table $DT(D_{name}, D_{attributes})$ of the DF will become a node $V(id^V, P^V, L^V)$ where:

- Label l_1 represents the table type, which can be *dimension* or *parameter* depending on the following cases:

– If DT is related to a fact table via the FactDimRel relationship or is related to another dimension via the SemanRel relationship, then $l_1 = \{dimension\}/L^V = \{l_1\}$

– If DT is related to a dimension table via the HierarRel relationship and is a parent, then $l_1 = \{parameter\}/L^V = \{l_1\}$

- Label l_2 represents the table name: $l_2 = D_{name}/L^V = L^V \cup l_2$

- Each identifier a_w (if any) will be converted into a property $p \leftarrow a_w/P^V = P^V \cup p$

- Each attribute $a_i \in D_{attributes}$ will be converted into a property $p \leftarrow a_i/P^V = P^V \cup p$

Rule 3: Mapping OA. Each optional attribute $a_k \in oa^{Att}$ ($oa^{Att} \subseteq OA^{Att}$) of DT via DimOAttrRel relationship will be converted into a properties $p \leftarrow a_i/P^V = P^V \cup p$ in dimension node.

4.2.2 Rules for Mapping Relationships

Rule 4: Mapping FactDimRel Relationship. Each FactDimRel relationship will be transformed into a relationship $R(id^R, sourceNode, targetNode, T^R, P^R)$:

- sourceNode represents the fact;
- targetNode represents the dimension.
- t_1 denotes the relationship type, $t_1 = \{linkfact - dimension\}/TR = \{t_1\}$

Rule 5: Mapping HierarRel Relationship. Each HierarRel relationship will be transformed into a relationship $R(id^R, sourceNode, targetNode, T^R, P^R)$:

- sourceNode represents the child.
- targetNode represents the parent.
- t_1 denotes the relationship type, $t_1 = \{Precede\}/TR = \{t_1\}$

Rule 6: Mapping SemanRel Relationship. Each SemanRel relationship will be transformed into a relationship $R(id^R, sourceNode, targetNode, T^R, P^R)$:

- sourceNode represents the source node.
- targetNode represents the target node.
- For each attribute $a_i \in ra^m \subseteq RA^M$ it will be transformed into a property $p \leftarrow a_i/P^R = P^R \cup p$
- t_1 denotes the relationship type, $t_1 = \{dim - to - dim\}/TR = \{t_1\}$

5 CASE STUDY

This section presents the development of a personalized learning path recommendation system (LPRS) based on career goals, showcasing our graph-based NoSQL DW. We provide a practical example of applying the conceptual, logical, and physical design phases for NoSQL DW to validate our conceptual DW model and highlight graph databases' strengths in handling complex relationships and queries in recommendation systems.

Graph NoSQL databases outperform in complex relationship modeling Fernandes and Bernardino (2018); Sellami et al. (2019), making them ideal for LPRS that analyze relationships between courses, learning objectives (LO), and job positions. While previous studies have focused on graph databases Beutling and Spahic-Bogdanovic (2024); Nguyen et al. (2022), we emphasize GDW's scalable data integration and advanced analytics for personalized recommendations. Based on our previous Knowledge Graph (KG) architecture Nguyen et al. (2022) links educational resources with career objectives across three layers: Career Concept, Course Concept, and Competency Concept. We then create a GDW using this KG to enhance data-driven recommendations for LPRS.

The following subsections detail the next three phases of NoSQL DW design, involving conceptual, logical, and physical design.

5.1 Conceptual DW Model Design

In the conceptual DW model design process, we use a top-down approach to identify essential data components and their relationships, ensuring alignment with problem requirements. For this case, we focus on addressing learning path recommendation issues based on users' existing skills and career goals. We identify the main entity groups: Courses, Careers, Users, and Skills. Skills encompass various entities such as Programming Languages, Frameworks, Platforms, Knowledge Areas, and Tools, forming the "Family Layer." Within this layer, we define two types of tables: "Fact Family" and "Dimension Family."

The "Fact Family" includes fact tables like job postings and course enrollments, along with related measures. The "Dimension Family" supports each fact table with detailed data. For example, the job posting fact table has dimensions like posting location, time, career, website, and organization, while the course enrollment fact table includes course name, instructor, website, and training organization.

We also identify optional attributes for dimensions, such as skill requirements, proficiency levels for courses, certifications for instructors, and phone numbers for users, to optimize data querying and support advising. The complete conceptual DW model is presented in Figure 2.

5.2 Mapping Logical DW Model Design

After designing the conceptual NoSQL DW model, transitioning to a logical GDW model is crucial for simplifying deployment and enhancing efficiency on specific database types. This section outlines how to map the conceptual NoSQL DW model to the logical GDW model using the mapping rules from section 4. We'll apply these rules to transform the LPRS problem's conceptual NoSQL DW model into a complete logical GDW model, focusing on mapping entities and relationships to corresponding components, thereby streamlining deployment on graph databases. The process involves two main steps:

Entity Mapping: Each entity identified in the conceptual DW model, such as courses and websites, is mapped to a separate node in the graph database. The names and types of these entities are transformed into corresponding labels, ensuring clear node identification. Attributes are also mapped to the properties of corresponding nodes.

Relationship Conversion: The relationships between entities, depicted by crow's foot symbols in the conceptual DW model, are converted into edges connecting the corresponding nodes in the graph. This step elucidates relationships, enabling a comprehensive and interconnected representation of nodes in the graph database. Figure 3 illustrates the logical graph model after conversion.

5.3 Building LPRS Solution Based on GDW

Based on the proposed logical GDW model, we deployed it using the Neo4j platform, incorporating data from 4,000 course records and 2,000 job postings sourced from public job websites. In this GDW, data entities are represented as nodes, and relationships are effectively modeled.

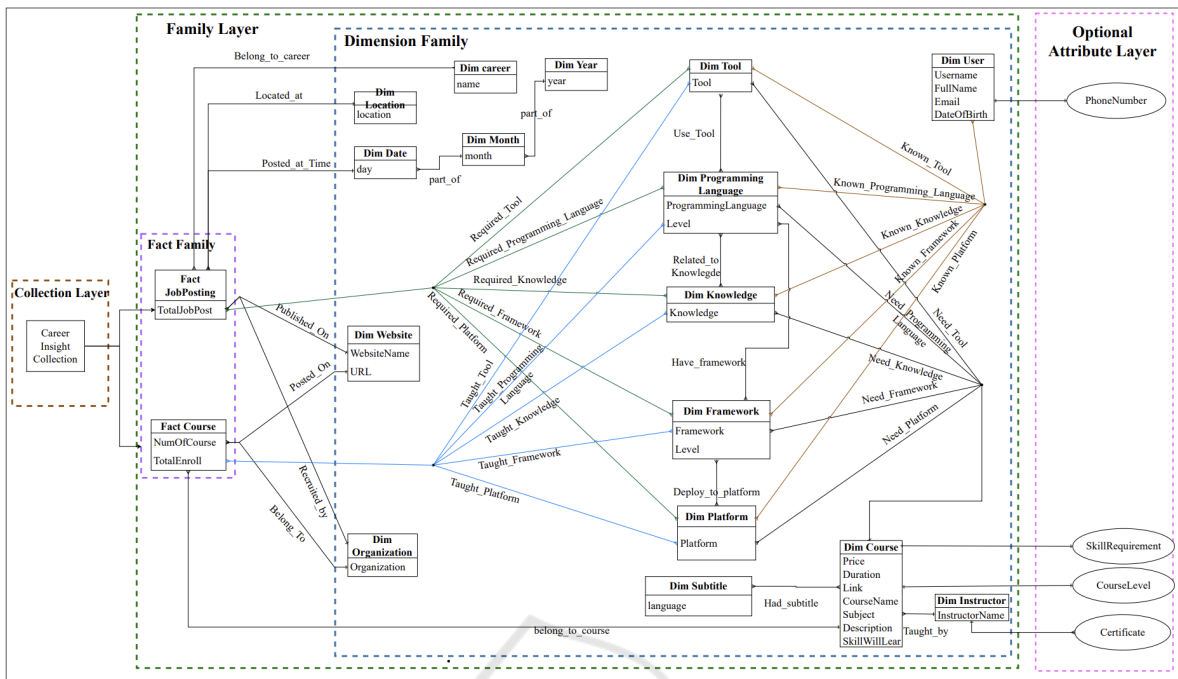


Figure 2: Proposed conceptual DW model for the LPRS problem.

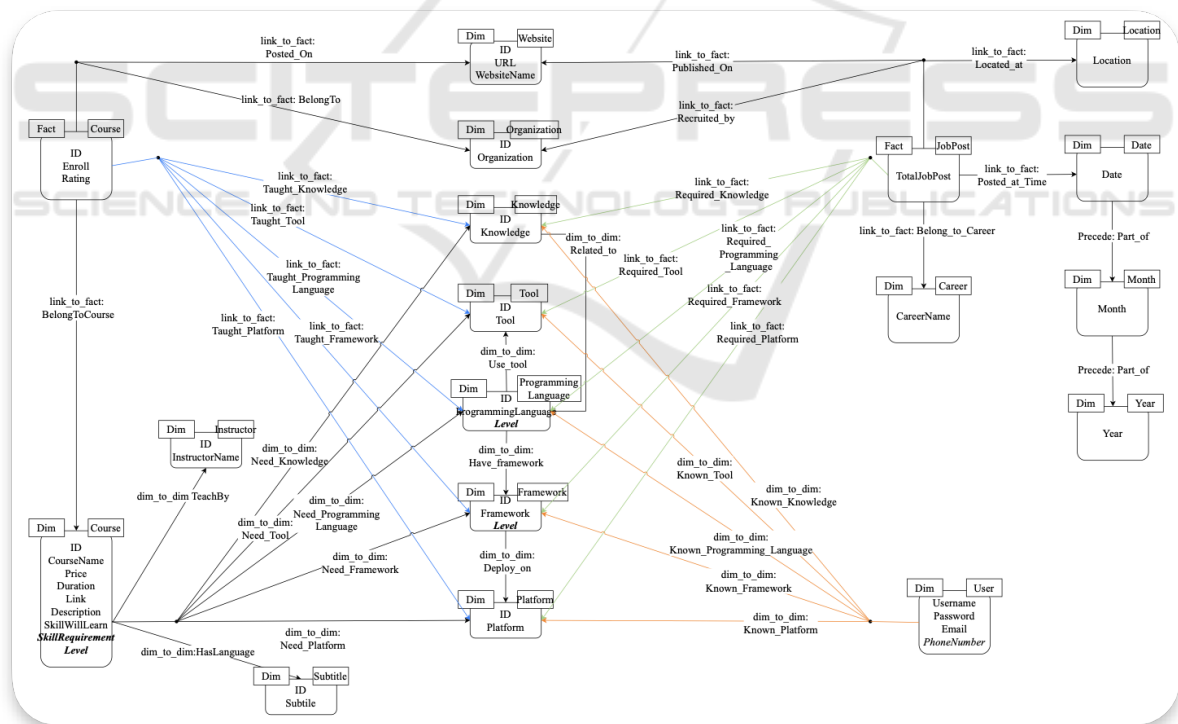


Figure 3: Logic model for the LPRS problem.

Two graph mining algorithms were developed for the Learning Path Recommendation System (LPRS). The first algorithm identifies critical competencies for a career and filters them based on the user’s existing

skills to recommend appropriate competencies. The second algorithm uses these competencies to find relevant courses, building a personalized learning pathway for the user based on course prerequisites and

levels. Figure 4 provides an example of the input and output for these algorithms.

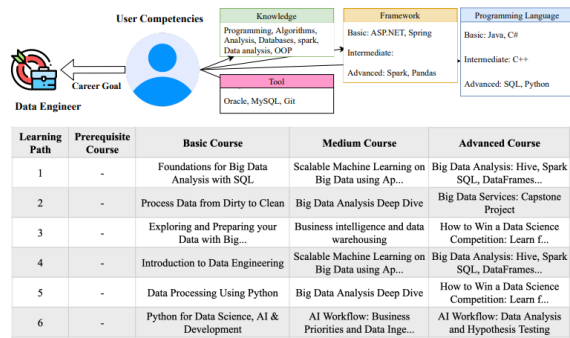


Figure 4: An example of the results of LPRS.

The implementation of LPRS on Neo4j underscores its efficiency and scalability, particularly in managing complex queries with intricate relationships. The GDW enables rapid and precise course recommendations tailored to user profiles, showcasing its capability to handle complex data models.

6 CONCLUSIONS

This study presents a comprehensive methodology for GDW design and implementation, introducing new symbols system and transformation rules that enhance the efficiency of model conversion. Through the Neo4j-based LPRS case study, we demonstrated GDW's ability to effectively manage complex data relationships, offering enhanced flexibility and performance in analyzing intricate data structures. GDW's strengths make it particularly valuable in Business Intelligence applications, such as recommendation systems and real-time analytics, where accurate and insightful decision-making is critical. Looking ahead, our future work will involve experimenting with large datasets (big data), exploring other NoSQL types like document, key-value, and column stores, and further refining GDW's performance and scalability. These efforts will help to validate GDW's benefits and broaden its applicability across diverse data management scenarios.

ACKNOWLEDGEMENTS

This research is partially supported by research funding from the Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam. This research is funded by the Faculty of Information Technology, University of Science, VNU-HCM, Viet-

nam, Grant number CNTT 2023-15.

REFERENCES

- Abdelhédi, F. et al. (2017). Logical unified modeling for nosql databases. In 2017.
- Akid, H. and Ayed, M. B. (2016). Towards nosql graph data warehouse for big social data analysis. In 2016.
- Akid, H. et al. (2022). Performance of nosql graph implementations of star vs. snowflake schemas. *IEEE Access*, 10:48603–48614.
- Banerjee, S. et al. (2021). A unified conceptual model for data warehouses. *Annals of Emerging Technologies in Computing (AETiC)*, 5(5):162–169.
- Beutling, N. and Spahic-Bogdanovic, M. (2024). Personalised course recommender: Linking learning objectives and career goals through competencies. In *Proceedings of the AAI Symposium Series*, volume 3, page 72–81.
- Bhagal, J. and Choksi, I. (2015). Handling big data using nosql. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshop*.
- Chevalier, M. et al. (2015). Implementation of multidimensional databases with document-oriented nosql. In *Big Data Analytics and Knowledge Discovery: 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015, Proceedings*, volume 17. Springer International Publishing.
- Fernandes, D. L. and Bernardino, J. (2018). Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. *Information Sciences*.
- Nguyen, T., Vu, N., and Ly, B. (2022). An approach to constructing a graph data repository for course recommendation based on it career goals in the context of big data. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE.
- Oussous, A. et al. (2017). Nosql databases for big data. In *2017 IEEE World Congress on Services*.
- Puja, I., Poscic, P., and Jaksic, D. (2019). Overview and comparison of several relational database modelling methodologies and notations. In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.
- Sellami, A., Nabli, A., and Gargouri, F. (2019). Transformation of data warehouse schema to nosql graph database. In *Advances in Intelligent Systems and Computing*, page 410–420.
- Yangu, R., Nabli, A., and Gargouri, F. (2016). Automatic transformation of data warehouse schema to nosql database: comparative study. *Procedia Computer Science*, 96:255–264.