Data Science and Engineering (2022) 7:57–70
https://doi.org/10.1007/s41019-021-00174-0

RESEARCH PAPERS

A Unification of Heterogeneous Data Sources into a Graph Model in E-commerce

Sonal Tuteja[1] · Rajeev Kumar[1]

Received: 17 June 2020 / Revised: 15 September 2021 / Accepted: 28 November 2021 / Published online: 18 December 2021

Abstract
The incorporation of heterogeneous data models into large-scale e-commerce applications incurs various complexities and overheads, such as redundancy of data, maintenance of different data models, and communication among different models for query processing. Graphs have emerged as data modelling techniques for large-scale applications with heterogeneous, schemaless, and relationship-centric data. Models exist for mapping different types of data to a graph; however, the unification of data from heterogeneous source models into a graph model has not received much attention. To address this, we propose a new framework in this study. The proposed framework first transforms data from various source models into graph models individually and then unifies them into a single graph. To justify the applicability of the proposed framework in e-commerce applications, we analyse and compare query performance, scalability, and database size of the unified graph with heterogeneous source data models for a predefined set of queries. We also access some qualitative measures, such as flexibility, completeness, consistency, and maturity for the proposed unified graph. Based on the experimental results, the unified graph outperforms heterogeneous source models for query performance and scalability; however, it falls behind for database size.

Keywords Data mapping · E-commerce · Graph model · Ontology model · Relational model · Schema mapping

1 Introduction

The era of data modelling began with relational models representing data in the form of relations and interconnections among them using foreign keys [9, 32]. The relational model is the most widely accepted data model owing to various features, such as normalisation, integrity, and consistency [5, 9]. However, this model is not suitable for applications requiring flexible schema and a higher number of joins for query processing [2, 21, 33]. To overcome such disadvantages of the relational model, various other not-only-structured-query-language (NoSQL) models have been proposed (e.g. key–value models, document models, graph models, etc.). These models are suitable for various applications involving schema flexibility and faster query processing. However, these features are achieved at the cost of one or more properties, such as atomicity, consistency, isolation, and durability (ACID) [5, 17].

Subsequently, graphs have emerged as data modelling techniques for large-scale applications with heterogeneous, schemaless, and relationship-centric data [8, 33]. In a graph model, data are stored in the form of nodes and edges representing entities and relationships among the entities, respectively [25]. Nodes and relationships can have properties associated with them in the form of key–value pairs. The direct storage of edges with corresponding nodes helps in faster traversal of relationships in the graph model [21].

As several data models exist, it is very challenging to choose the appropriate data model for an application. For example, the relational model can be applied in e-commerce applications for transactional processing; however, these models do not cater to complex situations. Due to the inflexible schema and the higher number of joins, the relational model is inadequate for large e-commerce applications, requiring schema flexibility, personalised recommendations, etc. Therefore, other data models have also been considered for such applications. For example, researchers have incorporated the ontology model in e-commerce applications for representing products

✉ Rajeev Kumar
rajeevkumar.cse@gmail.com

Sonal Tuteja
sonalt9@gmail.com

1 School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India

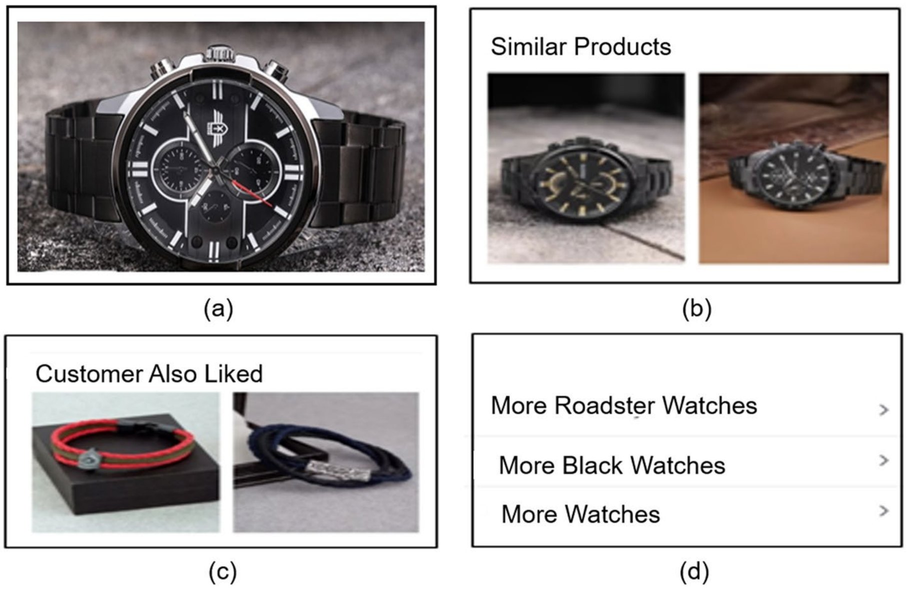

Fig. 1 A scenario representing various search results in an e-commerce application: **a** product viewed by a given customer, **b** products similar to the searched product, **c** products liked by the customers with similar preferences, and **d** other optional results

and their attributes [20, 39]. Graph models have been employed for the analysis of customer social networks in e-commerce applications [7]. However, the utilisation of heterogeneous data models in an application may leave out important relationships among entities of different data models. Moreover, queries entailing data from various data models may incur communication overheads [17].

To utilise the graph model for an application, its data needs to be mapped into the graph model. Different techniques have been discussed for mapping different types of data to graph models. The mapping techniques from a relational model to a graph model have been discussed in [24, 32, 38]. In addition, a set of guidelines has been presented for mapping an ontology model to a graph model [16]. However, data unification from heterogeneous source models into a graph model has not received much attention. In this work, we propose a framework for the unification of data from heterogeneous source models into a graph model. The motivation for adopting a graph model for the unification is due to its flexibility, and faster relationship traversals, which are important aspects in e-commerce applications [24]. We first transform each source model into a corresponding graph model and then unify all such graph models into a single graph. To assess this work's applicability, we analyse and compare a few objective measures, such as run-time query performance, scalability, and database size, of the proposed unified graph model with source data models. We also carry out the qualitative assessment of the different measures, namely flexibility, completeness, and consistency, for the proposed unified graph. The contributions of our research work are as follows:

- We design a framework for data mapping from heterogeneous source models into a unified graph model, and
- For the unification of data from different graph models, we propose an algorithm for schema unification from different schema graphs into a unified schema graph.

The rest of this article is organised as follows. We consider a motivating scenario to highlight the relevance of this work in Sect. 2. Next, we discuss the related work in Sect. 3. We present the proposed framework for the unification and illustrate the framework for e-commerce application data in Sect. 4. We detail the experimental design and include results in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 A Motivating Scenario

We have chosen this scenario from a popular e-commerce web application. When a customer clicks on a product, as shown in Fig. 1a, different types of results are generated by the application. In Fig. 1b, the products similar to the searched product are displayed. In Fig. 1c, the products liked by the customers with similar preferences are shown. In addition, the customer is provided with the option to explore the products with the same brand, colour, and category (Fig. 1d).

As observed from this scenario, data from different sources is merged to provide appropriate recommendations to customers. Product attributes such as colour and brand have been considered to provide different results to customers. Customers' search history of products has also been incorporated into the results.

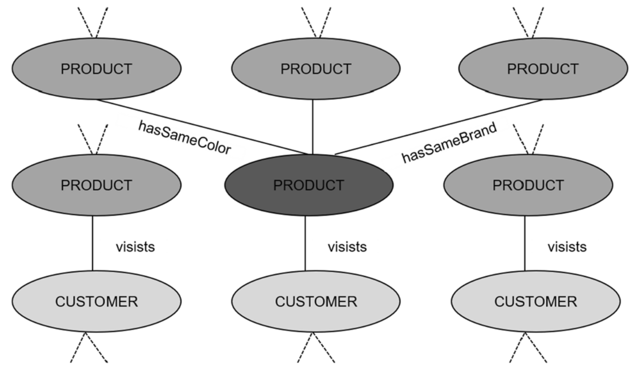

Fig. 2 Storage of relationships for results of the e-commerce search using a graph model

To model the scenario shown in Fig. 1, a graph model unifying the data from different data sources can be effectively utilised (Fig. 2). The graph model stores different types of relationships *a priori* based on the requirements of the application; such an approach reduces overheads for the generation of search results. For this scenario, relationships among products and customers have been created based on different features (Fig. 2); these relationships yield faster search results. For a given product, the products with the same brand, colour, etc., can be fetched in a one-level relationship traversal. In addition, the products visited by similar customers can be fetched in a two-level relationship traversal. The complexity of a traversal operation using the graph model is asymptotically lower than that of a join operation using the relational model [21]. Thus, the proposed framework for the unification of data from heterogeneous sources into a graph model is beneficial.

3 Related Work

3.1 Schemaless Data Models in E-commerce

Schemaless data models are being evolved and adopted by applications to represent their data due to their flexible nature [29]. The major categories of schemaless data models include: key–value model, document model, column-oriented model, and graph model [6]. Researchers have experimented with various schemaless data models for e-commerce applications to speed up query processing and improve search results. Liu et al. [20] incorporated the ontology model to store the preferences of customers which helped provide customised results to them. Zhang et al. [39] proposed a framework based on the ontology model for product information retrieval in e-commerce applications. The incorporation of ontology showed improvement of results over keyword-based searching.

Graph models have been applied successfully for data management and real-time recommendations by e-commerce companies, such as eBay and Walmart [11, 36]. Ríos et al. [28] represented co-purchased products in different transactions of e-commerce data in the form of a graph; they applied community detection techniques to find the clusters of similar products. Ranganath [27] utilised a graph model for representing query attributes which helped improve the ranking of relevant items for e-commerce queries. Ding et al. [10] proposed the snapshot-based dynamic graph to express users' behaviour considering short-term and long-term preferences for recommendations. The clustering of users and recommendations using the dynamic graph provided better results than other contemporary algorithms. Huang et al. [15] incorporated a graph model to assess the similarity among products. The generated graph model was utilised to analyse the impact of network parameters on the sale of products. Li et al. [19] included a graph model for representing different types of relationships among products in e-commerce applications. The graph-based modelling was found to be suitable for knowledge representation, search ranking, and recommendation.

3.2 Graph Model-Based Frameworks

The emergence of a graph as a data modelling technique has led to the design and development of different conceptual models for graphs. Virgilio et al. [34] proposed a technique for transforming an entity-relationship model into a graph database with a focus on minimising data access for queries. Angles et al. [4] proposed mapping techniques to transform an RDF database into a property graph database, including data and schema. The authors verified the semantic and information perseverance of the proposed mapping techniques. Pokorný [26] focused on the conceptual modelling of graph databases. The author also designed various types of integrity constraints to be incorporated into the graph databases. Ghrab et al. [14] designed logical models for graph databases, their integrity constraints, and algebra to retrieve results from such databases.

Several frameworks have also been proposed for mapping different types of data into graphs. Petermann et al. [24] proposed an architecture for mapping relational data

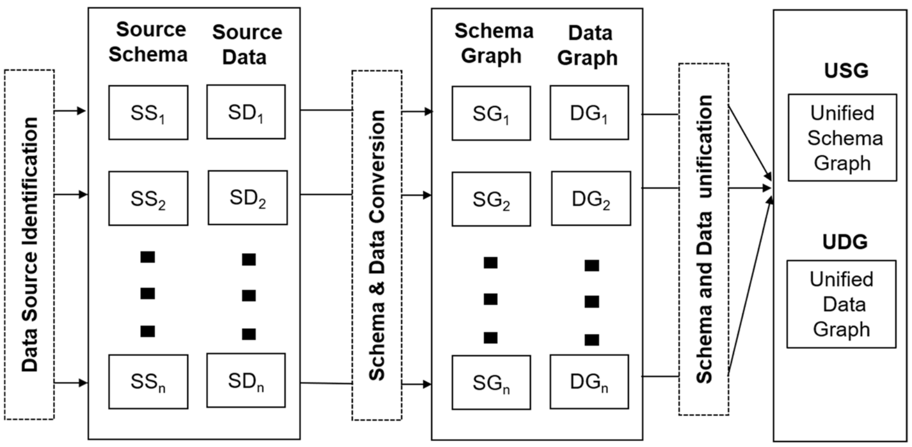

Fig. 3 The proposed framework for the unification of heterogeneous data sources into a graph model

from inter-related business information systems into a graph. Maccioni [22] proposed a flexible query answering system over graph modelled data. In this article, source data in different formats were converted into a graph and then different queries were performed on the graph model. Abulaish et al. [1] proposed a graph-based approach for modelling tweets in which relationships among tweets were created based on the correlation among them. The network was utilised for clustering tweets for event detection. Noel et al. [23] designed a graph-based cyber-security model, which captured complex relationships among network entities. The generated graph model was utilised to predict possible attack paths as well as critical vulnerabilities.

Researchers have analysed and compared the performance of the graph model with other data models. Vicknair et al. [32] performed several experiments to compare query performance, scalability, and space requirements of the graph model with the relational model. Based on the experiments, the graph model outperformed the relational model for structural as well as data queries. Yoon et al. [38] integrated heterogeneous biological data into a graph and compared its performance with the relational model. The research concluded the out-performance of the graph model over the relational model for various queries. Wang et al. [37] proposed a technique for improving query performance of the graph model based on the indexing of the query graph. For executing a query, the algorithm checked whether the new query was a sub/super query of the existing query graph and utilised it to run the new query. Maccioni [22] proposed a flexible query evaluation framework over heterogeneous data sources. The framework created a compressed graph model for processing queries, which significantly improved the query performance for different datasets. Kumar [18] designed various graph models to represent election tweets and analysed different types of queries on these data models. The author concluded the reduction in execution times by using the indices on frequently accessed attributes.

3.3 Research Gaps

As per the state of the art of technology, multiple heterogeneous models have been utilised to represent different types of data in e-commerce applications [7, 19, 39]. However, queries entailing data from different data models may incur communication overheads [17]. Moreover, this may leave out important relationships existing among data from various sources. Therefore, the unification of these heterogeneous data models into a single model is a challenging problem to be addressed. Petermann et al. [24] proposed a framework for the unification of heterogeneous data sources into a graph; however, the unified graph's performance was not analysed. Existing research articles in the literature focus on the mapping of data from relational/RDF models to graph models [16, 33]. In addition, the comparison of query performance for graph models has been carried out with respect to relational or ontology models [3, 32]. Therefore, we identify the following research gaps:

- The unification of data from heterogeneous source models into a graph model has not been much considered, and
- The performance analysis of the graph model with heterogeneous source models has not been carried out.

In this work, we propose a framework to map data from heterogeneous source models into a unified graph model. The motivation for adopting a graph model for the unification is due to its flexibility, and faster relationship traversals [24]. We also compare the performance of graph models with heterogeneous data models.

4 The Proposed Framework

We propose a framework for the unification of data from heterogeneous source models into a graph model, which consists of three steps: (i) data source identification, (ii) schema and data conversion, and (iii) schema and data unification. These are depicted in Fig. 3.

4.1 Terminology

Before discussing the proposed framework, we first describe terminologies used in this work, as follows:

- **Source schema and source data:** The source schema represents the schema of the data source considered for the unification. For different data sources, source schemas can be relational models, ontology, graph models, etc. Source data represent data instances corresponding to the source schema [33].
- **Schema graph and data graph:** The schema graph represents the schema of the data source in the form of a graph model including nodes, relationships, and their properties [35]. The data graph represents data instances corresponding to the schema graph [33].
- **Unified schema graph and unified data graph:** The unified schema graph is the unification of different schema graphs where nodes of the same type are unified [24]. The unified data graph represents the unified data in the form of a graph corresponding to the unified schema graph.

4.2 Data Source Identification

In an e-commerce application, different data sources are utilised to provide relevant search results to customers. These data sources are represented in different data models based on requirements of the application. For example, an ontology model can be used to represent products' attributes [39]. Similarly, a graph model can be used to represent customers' social network [7]. Therefore, the first step of the unification is to identify all data sources as well as their schemas, which are further processed by the framework (Fig. 3). There may exist a data source having schemaless data model in an application. However, schemaless data models have implicit schema associated with them that can be extracted using different methods, as discussed in [6, 29].

4.3 Schema and Data Conversion

The source schemas identified in the previous step exist in different data models; therefore, each of them is converted into a corresponding schema graph. Researchers have proposed several techniques to convert various types of source schemas into a schema graph. The conversion from a relational schema into a schema graph is discussed in [33, 35]. In this work, we adopt an algorithm from [33] to convert a relational schema into a schema graph. We also propose an algorithm for schema conversion from an ontology model into a graph model based on guidelines discussed in [16].

Algorithm 1 Conversion from an Ontology Model into a Schema Graph

```
    Input: Ontology {$ONT$}
    Output: Schema Graph {$SG$}
 1: $SG \leftarrow \phi$
 2: for all $cls_i$ in $ONT.classes$ do
 3:     $SG$.addNode($cls_i$)
 4: end for
 5: for all $dp_i$ in $ONT.DataProperties$ do
 6:     $cls \leftarrow dp_i.domain$
 7:     $cls$.addProperty($dp_i$)
 8: end for
 9: for all $op_i$ in $ONT.ObjectProperties$ do
10:     $cls_d \leftarrow dp_i.domain$
11:     $cls_r \leftarrow dp_i.range$
12:     addEdge($cls_d, cls_r$)
13: end for
```

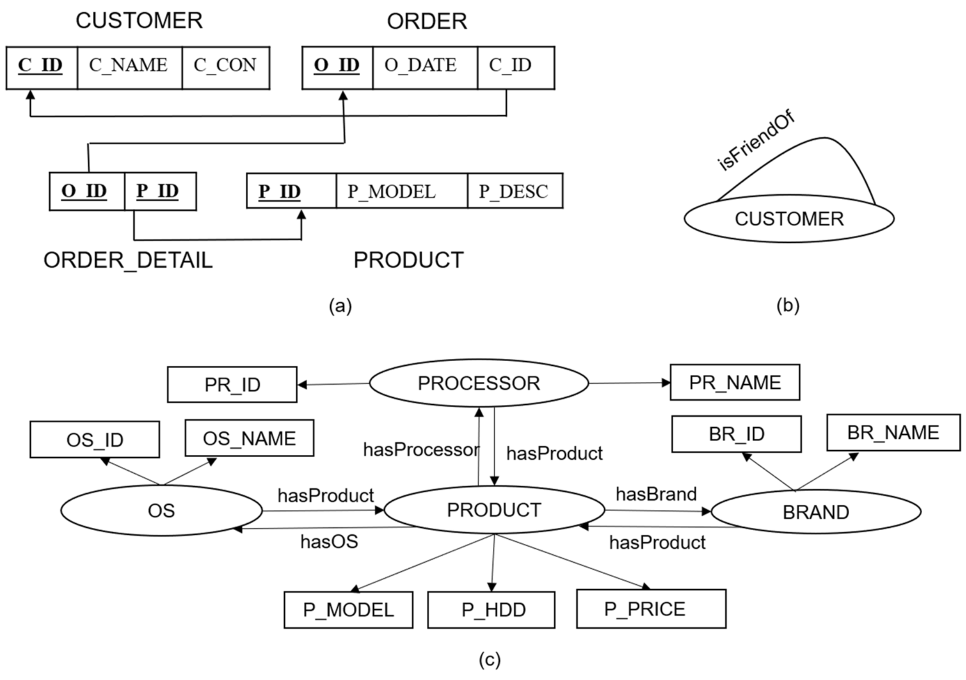

Fig. 4 Different data sources and their schema: **a** transactional data (relational model), **b** social network (graph model) and **c** product description (ontology model)

The steps for the conversion of an ontology model into a schema graph are presented in Algorithm 1. It takes an ontology *ONT* as input and returns a schema graph *SG* as output. The algorithm first initialises *SG* to ϕ as shown in line 1. Then, it creates nodes in the *SG* corresponding to each class cls_i in *ONT* (lines 2–4). For each data property dp_i in *ONT*, its class *cls* is determined and the property is attached to the node corresponding to *cls* in *SG* (lines 5–8). For each object property op_i in *ONT*, its source class cls_d and target class cls_r are determined, and an edge is created from source class node cls_d to target class node cls_r (lines 9–13).

For each source schema SS_i, schema graph SG_i is generated using Algorithm 1. Based on SG_i, source data SD_i is also converted into the corresponding data graph DG_i (Fig. 3).

4.4 Schema and Data Unification

In this step, different schema graphs generated in the previous step are combined into a unified schema graph. This unification aims to combine nodes and edges of different schema graphs into a single schema graph. Therefore, nodes representing same entities in different schema graphs are unified into one node by merging their attributes and relationships.

Algorithm 2 Unification of Schema Graphs

```
Input: Set of Schema Graphs $\{SG_1, SG_2, ..., SG_n\}$
Output: Unified Schema Graph $\{USG\}$
1: $USG \leftarrow \phi$
2: for $SG_i \in \{SG_1, SG_2, ..., SG_n\}$ do
3:     for all nodes $n_j$ in $SG_i$ do
4:         if $n_j \notin SG_i.nodes$ then
5:             $USG.addNode(n_j)$
6:         end if
7:         for all properties $p_k$ in $n_j$ do
8:             if $p_k \notin n_j.properties$ then
9:                 $n_j$.addProperty($p_k$)
10:            end if
11:        end for
12:    end for
13:    for all edges $e_j$ in $SG_i$ do
14:        $USG.addEdge(e_j)$
15:        for all properties $p_k$ in $e_j$ do
16:            $e_j$.addProperty($p_k$)
17:        end for
18:    end for
19: end for
```

Algorithm 2 describes the steps required for the unification of schema graphs. It takes a set of schema graphs $\{SG_1, SG_2, ..., SG_n\}$ as input and returns a unified schema graph *USG* as output. The algorithm first initialises *USG* to ϕ as shown in line 1. For each schema graph SG_i, the algorithm processes nodes (lines 3–12) followed by edges (lines 13–18) of SG_i. For each node n_j of SG_i, it adds n_j to *USG.nodes* if it is not present in *USG.nodes* (lines 4–6). Then, it adds properties p_k of n_j to n_j*.properties*. After processing nodes of SG_i, the algorithm adds edges e_j of SG_i to *USG.edges* (lines 13 and 14). Then, it adds properties p_k of e_j

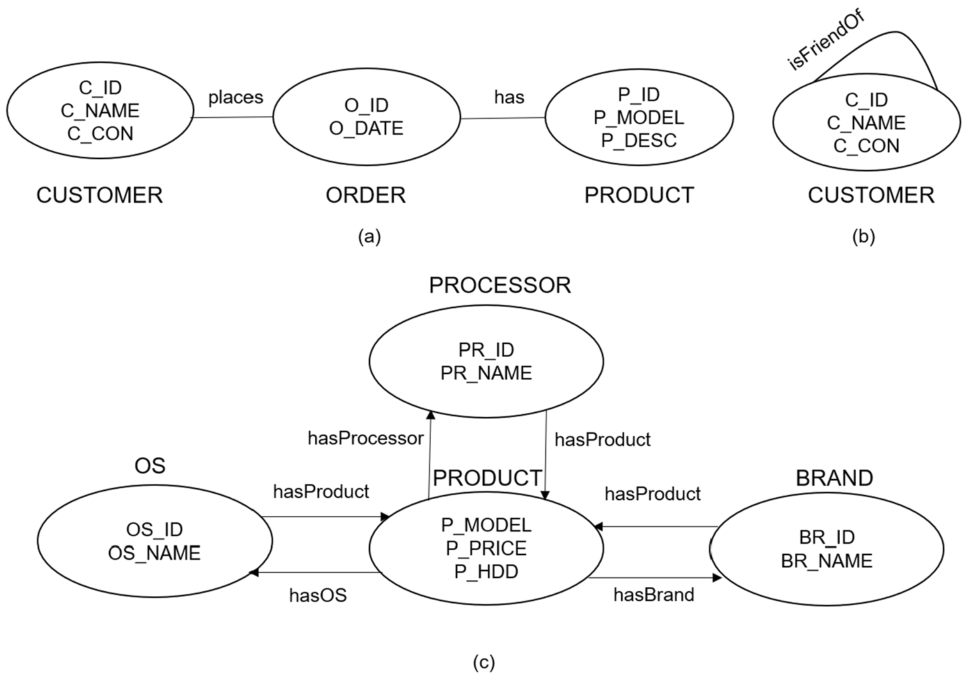

Fig. 5 Schema graphs corresponding to **a** relational model in Fig. 4a, **b** graph model in Fig. 4b and c ontology model in Fig. 4c

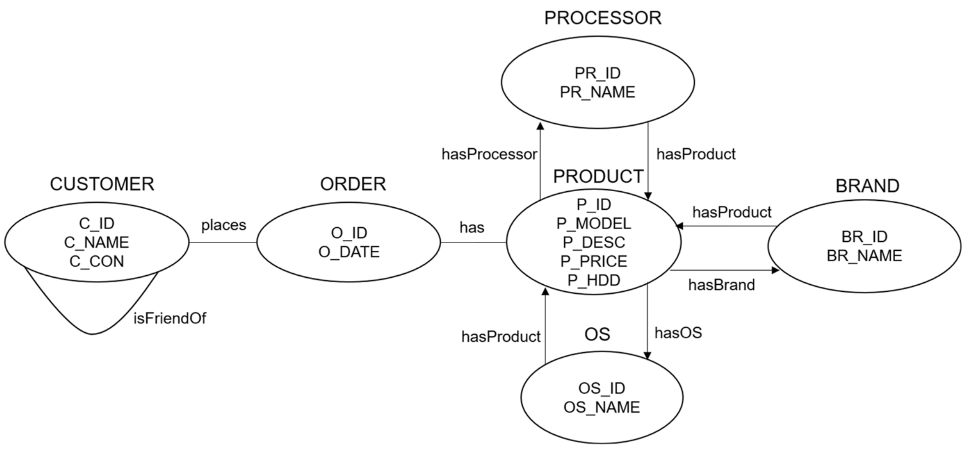

Fig. 6 Unified schema graph for schema graphs in Fig. 5

to $e_j.properties$ (lines 15–17). The same process is repeated for all schema graphs in the set.

Using Algorithm 2, the set of schema graphs $\{SG_1, SG_2, ..., SG_n\}$ is converted into a unified schema graph USG. Based on USG, the set of data graphs $\{DG_1, DG_2, ..., DG_n\}$ is also converted into a unified data graph UDG as shown in Fig. 3.

Using the architecture mentioned above, the data from heterogeneous sources with different data models are converted into a unified data graph which is further used for query processing by different subsystems of an e-commerce application. The advantage of using the unified graph is the reduction in the query execution time. In addition, it reduces overheads of passing intermediate results from one model to another for query processing.

4.5 An Illustration

To illustrate the proposed framework (Fig. 3), we consider data from an e-commerce application with different data models. It consists of three different types of data: (i) transactional data, (ii) product ontology, and (iii) customer social networks. We have taken transactional data from an online repository[1]. The transactional data is represented in the form of a relational model consisting of four relations:

[1] https://www.kaggle.com/carrie1/ecommerce-data/data .

Table 1 Data subsets in increasing order of their size

Dataset	No. of Transactions (K)	Data Size (KB)
DS1	4	56
DS2	40	320
DS3	400	6570
DS4	2200	30900

CUSTOMER, ORDER, ORDER_DETAIL, and PRODUCT (Fig. 4a).

Product ontology has been crawled from the web and stored in RDF format containing four classes: PRODUCT, OS, BRAND, and PROCESSOR. Data properties and object properties associated with these classes are shown in Fig. 4c. Correspondingly, we have generated customer social networks using randomisation represented in the form of a graph model (Fig. 4b).

As described in the framework (Fig. 3), each source schema is converted into a corresponding schema graph. The schema graph corresponding to the relational model in Fig. 4a is generated using the technique given by [33] (Fig. 5a). As the schema of the social network of customers shown in Fig. 4b is already present in graph format, schema conversion is not required. For the product ontology shown in Fig. 4c, the corresponding graph model (Fig. 5c) is generated using Algorithm 1. For each data source, the data graph is also generated based on the schema graph.

Different schema graphs shown in Fig. 5 are converted into a unified schema graph using Algorithm 2, as shown in Fig. 6. As CUSTOMER and PRODUCT nodes are present in more than one schema graph (Fig. 5), they are merged into the unified schema graph (Fig. 6). Based on the unified schema graph, different data graphs are also converted into the unified data graph.

5 Experimental Design and Results

5.1 Experimental Design

To demonstrate the effectiveness of our proposed work, we consider various objectives and subjective measures for the comparison of the unified graph with other data models. We design queries to analyse and compare the performance of source data models with the unified graph. The datasets contain data from heterogeneous sources, which are converted into the unified graph using the proposed framework discussed in Sect. 4 (Table 1).

Datasets:

We consider data from heterogeneous sources: (i) transactional data, (ii) product ontology, and (iii) customer social networks. We have taken transactional data from an online

Table 2 E-commerce queries for the comparison of unified graph with heterogeneous data models

Query	Before Unification		Unified Graph	
	Data Model (R,O,G)	Operations	Operations	Description
Q1	R	Aggregation	1-level traversal, aggregation	For each product, find the number of times it has been purchased
Q2		3-level join	2-level traversal	For a given customer *c*, find the products ordered by *c*
Q3		1-level traversal	1-level traversal	For a given processor *pr*, find the products having processor *pr*
Q4	O	2-level traversal	2-level traversal	For a given product *p*, find the products with the same brand as of *p*
Q5		1-level traversal	Searching of nodes based on property value	For a given model *m*, find all the products having model *m*
Q6		3-level join (R), 2-level traversal (O)	4-level traversal	For a given customer *c*, find the products with the same brand as of the products purchased by *c* in the past
Q7	R,O,G	1-level traversal (G), 3-level join (R)	3-level traversal	For a given customer *c*, find the products purchased by the customers in the social network of *c*
Q8		1-level traversal (G), 3-level join (R), 2-level traversal (O)	5-level traversal	For a given customer *c*, find the products with the same brand as of the products purchased by customers in the social network of *c*

R: Relational; *O*: Ontology; *G*: Graph repository[2], which is in the form of a relational model. Product ontology is crawled from the web and stored in RDF format. Correspondingly, we have generated a social network of customers using randomisation. The schemas of heterogeneous data sources before the unification are shown in Fig. 4. The data from these heterogeneous sources are converted into the unified graph (Fig. 6) using our proposed architecture. Further, we create appropriate indices in different data models for faster data access. The original dataset described above contains 400K e-commerce transactions and corresponding customers, products, and their attributes. We create subsets and supersets of the original datasets to analyse and compare the unified graph's scalability and database size with heterogeneous data models. The subsets DS1 and DS2 contain 4K orders and 40K transactions, respectively. The complete dataset with 400K transactions is named DS3. The superset DS4 with 22,000K transactions is generated by randomisation of DS3. All datasets are stored in heterogeneous data models and the unified graph; then, we compare different performance measures.

Queries:

To compare the run-time query performance and scalability of different data models, we design different queries, as shown in Table 2. The criteria for designing queries are based on the coverage of various functionalities of e-commerce subsystems as well as the coverage of different source data models [13, 39]. We design queries Q1 and Q2 to compare the relational model with the proposed unified graph. To compare the ontology model with the proposed unified graph, we design queries Q3, Q4, and Q5. Similarly, queries Q6, Q7, and Q8 are designed to compare the proposed unified graph with heterogeneous source data models.

Set-up:

The experimental set-up consists of a system with an Intel Core i7 processor with 12 GB of RAM, running on Windows 10, version 1709. During experimentation, only system programs and databases needed for query execution are run on the system. For the execution of queries on different data models, we chose MySQL[3], Virtuoso[4] and Neo4j[5]. for the implementation of relational, ontology, and graph models, respectively. The criteria for selecting these tools are their open source availability and support from the academic and scientific community. We have loaded different types of data in their corresponding databases for the execution of queries.

[2] https://www.kaggle.com/carrie1/ecommerce-data/data.

[3] https://www.mysql.com/.

[4] https://virtuoso.openlinksw.com/.

[5] https://neo4j.com/.

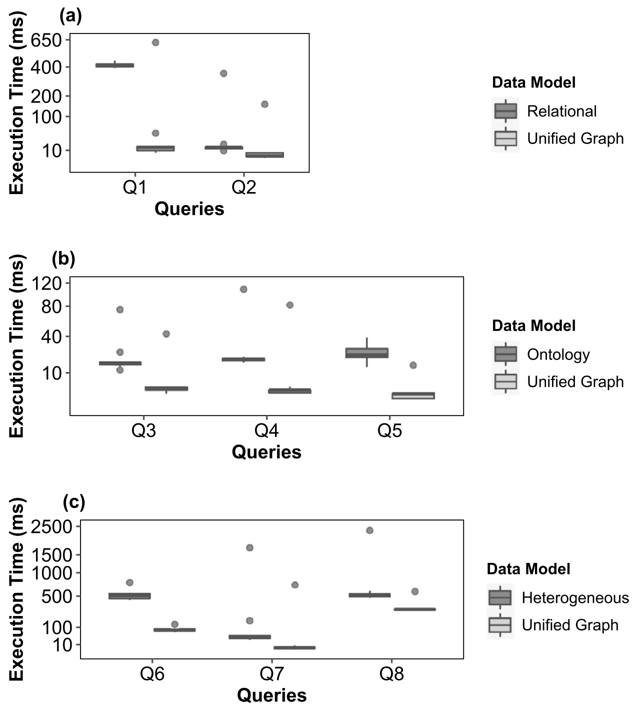

Fig. 7 Analysis and comparison of execution times of the unified graph with **a** relational, **b** ontology and **c** heterogeneous data models

5.2 Results

To assess the viability of the proposed framework, we have tested and compared the run-time query performance, scalability, and database size of the unified graph with those of heterogeneous data models. We have also accessed qualitative parameters for the unified graph, such as accuracy, completeness, and flexibility.

5.2.1 Run Time Query Performance

Query execution is considered an important criterion for the comparison of different data models [32]. Therefore, we have tested and compared the run-time performance of the unified graph with that of source data models for a predefined set of queries (Subsect. 5.1). Using dataset DS4, we have executed each query ten times over source data models as well as the unified graph and measured their execution time in milliseconds. In addition, parameters for queries are generated randomly, and the same parameters are used to compare different data models.

We have used box-plots to compare the unified graph's execution time with source data models. Although error-bar plots are the most popular method for showing data distributions, they are not appropriate for our results. The amount of information provided by error-bar plots is limited to two values, namely mean and standard deviation, which is also

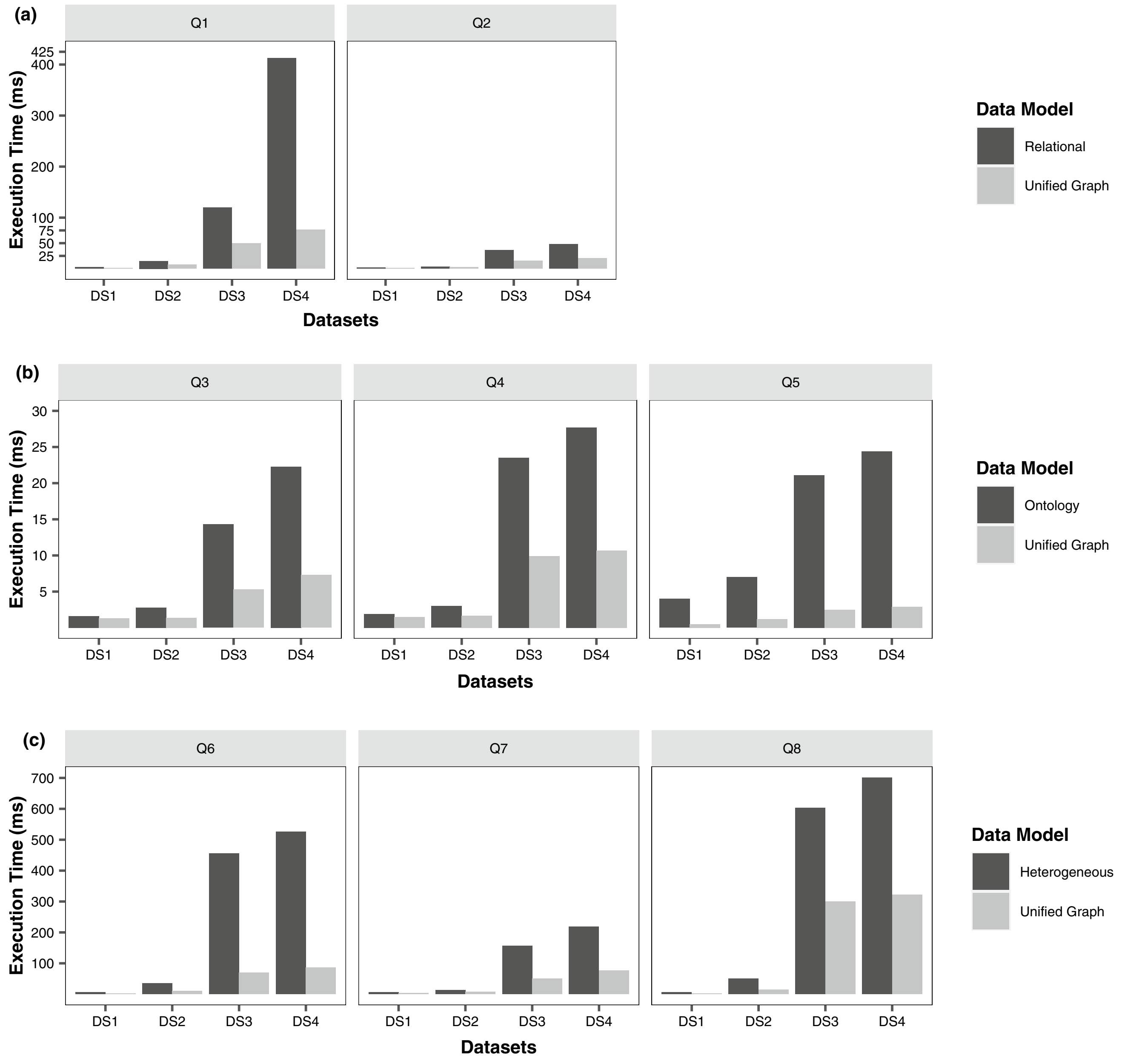

Fig. 8 Impact of data size on the average execution time of the unified graph and its comparison with **a** relational, **b** ontology and **c** heterogeneous data models

affected by the presence of outliers [12, 31]. The cold start execution time for queries can be much higher than the subsequent execution times; therefore, error-bar plots may also give misleading results. On the other hand, box-plots can display different measures, including median, lower and upper quartiles, and whiskers. They also explicitly represent outliers. Thus, a box-plot helps to understand results with better accuracy and precision.

Comparison with relational model: To compare the unified graph's performance with the relational model, we have considered queries Q1 and Q2 (Table 2). It can be observed from Fig. 7 that the cold execution time of Q1 using the unified graph is greater than that of the relation model, represented as outliers in box-plots. It is due to a single relation that is accessed in the relational model, whereas a one-level relationship traversal is performed for every product node in the unified graph. However, all other values such as the median, first, and third quartiles are lower for the unified graph than the relational model. For query Q2, the graph model outperforms the relational model in cold execution as well as in other subsequent executions (Fig. 7). Evidently, the time-complexity of relationship traversals in the graph

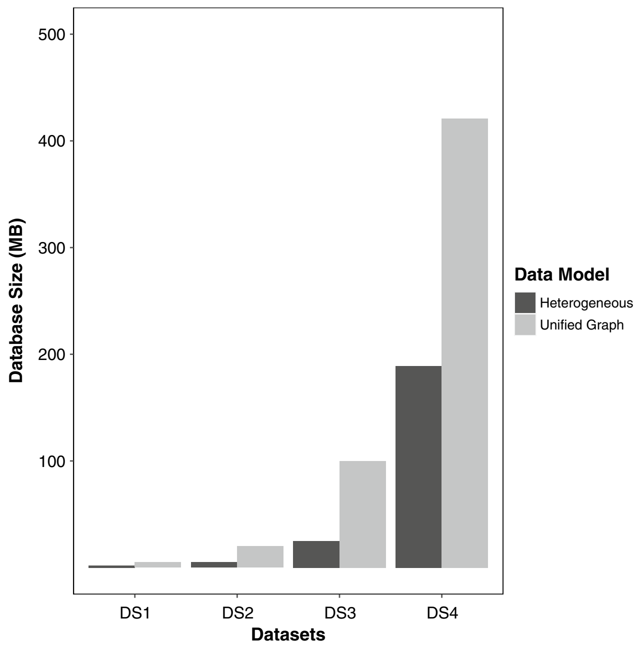

Fig. 9 Comparison of the database size of unified graph with heterogeneous data models

model is asymptotically lower than that of join operations in the relational model [21].

Comparison with ontology model:

We have executed queries Q3, Q4, and Q5 for fetching data from the ontology model before the unification and the unified graph model after the unification (Table 2). As observed for queries Q3 and Q4, the unified graph performs better in cold execution, represented as outliers in box-plots, as well as in other subsequent executions (Fig. 7). It shows the unified graph's efficiency over the ontology model for such traversal queries [3]. For query Q5, the unified graph outperforms the ontology model as indices created on node properties in the graph model help in faster searching.

Comparison with heterogeneous models:

We have designed queries Q6, Q7, and Q8 entailing data from heterogeneous data models before the unification and compared their execution time with the unified graph (Table 2). It can be observed from Fig. 7 that the graph model outperforms heterogeneous data models in cold executions and other subsequent executions for all the queries. The unified graph's better performance is due to low-cost operations in the graph model compared with other data models. In addition, heterogeneous data models create overheads for passing data from one model to another; this further increases execution time. Therefore, the unified graph performs much better for queries entailing data from multiple heterogeneous sources.

5.2.2 Scalability

To analyse the impact of data size on execution times of different data models, we have executed queries on all datasets using the unified graph as well as source data models, and calculated their average execution times. For comparing the scalability of the unified graph with the relational model, we have executed queries Q1 and Q2 on both data models. It can be observed from Fig. 8, the growth of the average execution time of the unified graph is slower than that of the relational model. It justifies the scalability of the unified graph as compared to the relational model.

We have executed queries Q3, Q4, and Q5 on both data models to compare the unified graph's scalability with the ontology model. As can be seen from Fig. 8, the growth of the average execution time of the ontology model is faster than that of the unified graph. The unified graph's scalability is justified as the increase in the execution time of the unified graph is slower than that of the ontology model. Similar behaviour can be seen for queries Q6, Q7, and Q8, as represented in Fig. 8.

5.2.3 Database Size

For each of datasets (Sect. 5.1), we have measured disk space utilised by the unified graph as well as heterogeneous source data models. It can be observed from Fig. 9 that the growth of the database size of the unified graph is faster than that of heterogeneous data models. It is due to the index-free adjacency in the graph model, which physically stores relationships with associated nodes [32]. However, this is not a matter of much concern owing to the economics of the storage technology.

5.2.4 Qualitative Measures

Qualitative measures are of great significance for the comparison of different data models. We have considered various qualitative measures, namely accuracy, completeness, flexibility, and maturity. The similarity in results produced by heterogeneous models and the unified graph justifies the consistency and completeness of the unified graph. The graph model has a simple mutable schema, making it more flexible than other heterogeneous data models. However, the graph model may not be considered superior concerning maturity as this currently has lower market penetration and a smaller user base.

- *Flexibility:* The graph model provides better flexibility over other data models. For instance, the schema of the relational model is difficult to modify for changing requirements. In the ontology model, properties of relationships cannot be specified. On the other hand, the graph model provides better schema flexibility over other available data models. The graph model can also be used to specify types and properties of nodes and their relationships.

- *Completeness:* The queries chosen for the experimentation were run on source data models and the unified graph using the same parameters. The equivalence of the result sets generated by heterogeneous data models and the unified graph ensures the unified graph's completeness. However, more extensive formal techniques need to be employed to verify completeness. This is an area of future research.
- *Consistency:* The usage of heterogeneous data models in an application may lead to inconsistency as data for some entities are present in different models. However, the unified graph aggregates the same entities in different models and creates one node corresponding to them. Therefore, it is more consistent than the heterogeneous source data models. However, formal techniques need to be employed to verify consistency. This is also an area of future research.
- *Maturity and support:* Being an emerging data model, the graph model has lesser maturity and community support at present. There currently exist only few commercial databases implementing graph models, but most of them do not implement most features of the graph model.

6 Discussion and Conclusion

This research work proposes and validates an architecture that can be used to unify data from heterogeneous data sources into a graph model. To validate the proposed framework's effectiveness, we have analysed the runtime query performance, scalability, and database size of the unified graph and compared them with those of heterogeneous source data models. We have shown that the graph model outperformed the relational as well as ontology models in all performance measures, except for aggregation queries. The cold execution time using the relational model for aggregation queries was lower than that of the graph model. However, the graph model outperformed the relational model for descriptive statistics of the execution time, namely median, lower quartile, and upper quartile values. The unified graph performed much better for all queries entailing data from multiple heterogeneous sources before the unification. Besides, the unified graph was found to be more scalable compared with corresponding source data models. However, the database size of the unified graph was larger than that of source data models owing to the physical storage of relationships with associated nodes. We have also compared qualitative measures, such as flexibility, completeness, consistency, maturity, and support for various data models, and found them comparable. As an emerging technique, the unified graph model has less maturity and support than other data models at present.

As the graph model revolutionises, our work can be a prerequisite for e-commerce applications to map various data sources into a unified graph. It can also help in unveiling relationships hidden in heterogeneous data sources. The data transformation into the unified graph will reduce the query processing time for different subsystems of e-commerce applications. However, there are some limitations to our research work. First, we have considered a few data models, namely relational and ontology models, for the unification and comparison with the unified graph. In the future, we will also incorporate other emerging schemaless data models for the unification and analysis. Second, we have not considered entity linking for the unification of the same entities with different names in heterogeneous data models [30]. In the future, we will incorporate entity linking for the unification of these types of entities. In addition, we will apply the proposed unification architecture in other domains such as e-mail targeting and social networking.

Acknowledgements The authors thank the reviewers for their valuable comments by which the understanding and readability of this manuscript are greatly improved. The authors are also thankful to the Editor-in- Chief, the Editor, and the Editorial Office Assistant(s) for managing this manuscript.

Author Contributions The first author contributed to the design and implementation of the research. The second author provided critical feedback and helped shape the research, analysis, and manuscript.

Declarations

Availability of data and materials The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Conflict of interest We declare that we do not have any competing interest in connection with the work submitted.

Funding We declare that no funding has been received for conducting the research.

References

1. Abulaish M, Sharma S, Fazil M (2019) A multi-attributed graph-based approach for text data modeling and event detection in Twitter. In: Proceedings 11th International Conference Communication Systems Networks, IEEE, pp 703–708, https://doi.org/10.1109/COMSNETS.2019.8711451
2. Agrawal R, Somani A, Xu Y (2001) Storage and querying of e-commerce data. In: Proceedings 27th VLDB Conference, Morgan Kaufmann, pp 149–158
3. Alocci D, Mariethoz J, Horlacher O, Bolleman JT, Campbell MP, Lisacek F (2015) Property graph vs. RDF triple store: a comparison on GLYCAN substructure search. PloS one 10(12):1–17. https://doi.org/10.1371/journal.pone.0144578
4. Angles R, Thakkar H, Tomaszuk D (2020) Mapping RDF databases to property graph databases. IEEE Access 8:86091–86110. https://doi.org/10.1109/ACCESS.2020.2993117
5. Atzeni P, Jensen CS, Orsi G, Ram S, Tanca L, Torlone R (2013) The relational model is dead, SQL is dead and I dont feel so good myself. SIGMOD Record 42(2):64–68. https://doi.org/10.1145/2503792.2503808
6. Atzeni P, Bugiotti F, Cabibbo L, Torlone R (2020) Data modeling in the NoSQL world. Comput Stand Interfaces 67:103149
7. Cai YL, Wang WD, Gong XY, Li YH, Chen CF, Jian M (2008) Mobile e-commerce model based on social network analysis. J Ch Univ Posts Telecommun 15:79–97. https://doi.org/10.1016/S1005-8885(08)60160-0
8. Cheng Y, Ding P, Wang T, Lu W, Du X (2019) Which category is better: benchmarking relational and graph database management systems. Data Sci Eng 4(4):309–322. https://doi.org/10.1007/s41019-019-00110-3
9. Codd EF (1990) The Relational Model for Database Management: Ver. 2. Addison-Wesley Longman, USA
10. Ding L, Han B, Wang S, Li X, Song B (2019) User-centered recommendation using US-ELM based on dynamic graph model in E-commerce. Int J Mach Learn Cybern 10(4):693–703. https://doi.org/10.1007/s13042-017-0751-z
11. EBay (2014) EBay now tackles e-commerce delivery service routing with Neo4j. Tech. rep., Neo Technology, https://dist.neo4j.com/wp-content/uploads/Neo4j_CS_eBay.pdf
12. Editorial, (2014) Kick the bar chart habit. Nat Methods 11: 113113. https://doi.org/10.1038/nmeth.2837
13. García MDMR, García-Nieto J, Aldana-Montes JF (2016) An ontology-based data integration approach for web analytics in e-commerce. Expert Syst Appl 63:20–34. https://doi.org/10.1016/j.eswa.2016.06.034
14. Ghrab A, Romero O, Skhiri S, Vaisman AA, Zimányi E (2016) GRAD: On graph database modeling. CoRR abs/1602.00503, http://arxiv.org/abs/1602.00503
15. Huang HJ, Yang J, Zheng B (2019) Demand effects of product similarity network in e-commerce platform. Electro Commerc Res. https://doi.org/10.1007/s10660-019-09352-9
16. Jesús B (2017) RDF triple stores vs. labeled property graphs (Accessed on: Aug 25, 2021). https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/
17. Kaur K, Rani R (2015) Managing data in healthcare information systems: many models, one solution. Comput 48(3):52–59. https://doi.org/10.1109/MC.2015.77
18. Kumar P (2016) Graph data modeling for political communication on twitter. Masters thesis, Dept. Computer Science, Iowa State University
19. Li FL, Chen H, Xu G, Qiu T, Ji F, Zhang J, Chen H (2020) AliMeKG: Domain Knowledge Graph Construction and Application in E-Commerce. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Association for Computing Machinery, New York, NY, USA, CIKM '20, p 2581-2588, https://doi.org/10.1145/3340531.3412685
20. Liu W, Jin F, Zhang X (2008) Ontology-Based User Modeling for E-Commerce System. In: Proceedings 3rd International Conference Pervasive Computing & Applications, IEEE, pp 260–263, https://doi.org/10.1109/ICPCA.2008.4783589
21. Ma S, Li J, Hu C, Lin X, Huai J (2016) Big graph search: challenges and techniques. Front Comput Sci 10(3):387–398. https://doi.org/10.1007/s11704-015-4515-1
22. Maccioni A (2015) Flexible query answering over graph-modeled data. In: Proceedings 2015 ACM SIGMOD on PhD Symposium: Melbourne, ACM Press, pp 27–32, 10.1145/2744680.2744686
23. Noel S, Harley E, Tam K, Limiero M, Share M (2016) CyGraph: graph-based analytics and visualization for cybersecurity. In: Cognitive Computing: Theory & Applications, Handbook of Statistics, vol 35, Elsevier, pp 117–167, https://doi.org/10.1016/bs.host.2016.07.001
24. Petermann A, Junghanns M, Müller R, Rahm E (2014) Graph-based data integration and business intelligence with BIIIG. VLDB Endow 7(13):1577–1580. https://doi.org/10.14778/2733004.2733034
25. Pokorný J (2015) Graph databases: Their power and limitations. In: Computer Information Systems & Industrial Management, Springer, pp 58–69, https://doi.org/10.1007/978-3-319-24369-6_5
26. Pokorný J (2016) Conceptual and database modelling of graph databases. In: Proceedings 20th International Database Engineering & Applications Symposium, ACM Press, pp 370–377,https://doi.org/10.1145/2938503.2938547
27. Ranganath S (2018) Leveraging catalog knowledge graphs for query attribute identification in e-commerce sites. CoRR abs/1807.04923, arXiv: 1807.04923
28. Ríos SA, Videla-Cavieres IF (2014) Generating groups of products using graph mining techniques. Procedia Comput Sci 35:730–738. https://doi.org/10.1016/j.procs.2014.08.155
29. Sevilla Ruiz D, Morales SF, García Molina J (2015) Inferring Versioned Schemas from NoSQL Databases and Its Applications. In: Johannesson P, Lee ML, Liddle SW, Opdahl AL, Pastor López Ó (eds) Conceptual Modeling, Springer, pp 467–480, https://doi.org/10.1007/978-3-319-25264-3_35
30. Shen W, Han J, Wang J, Yuan X, Yang Z (2018) Shine+: A general framework for domain-specific entity linking with heterogeneous information networks. IEEE Trans Knowl Data Eng 30(2):353–366. https://doi.org/10.1109/TKDE.2017.2730862
31. Tripathi AK, Sharma K, Bala M (2018) A novel clustering method using enhanced grey wolf optimizer and MapReduce. Big Data Res 14:93–100. https://doi.org/10.1016/j.bdr.2018.05.002
32. Vicknair C, Macias M, Zhao Z, Nan X, Chen Y, Wilkins D (2010) A comparison of a graph database and a relational database: A data provenance perspective. In: Proceedings 48th Annual Southeast Regional Conference, ACM Press, pp 1–6, https://doi.org/10.1145/1900008.1900067
33. Virgilio RD, Maccioni A, Torlone R (2013) Converting relational to graph databases. In: Proc. 1st Int. Workshop Graph Data Management Experiences & Systems, ACM Press, GRADES '13, pp 1–6, https://doi.org/10.1145/2484425.2484426

34. Virgilio RD, Maccioni A, Torlone R (2014a) Model-driven design of graph databases. In: Conceptual Modeling, Springer, pp 172–185, https://doi.org/10.1007/978-3-319-12206-9_14
35. Virgilio RD, Maccioni A, Torlone R (2014b) R2G: A tool for migrating relations to graphs. In: Proceedings of 7th International Conference Extending Database Technology, pp 640–643, https://doi.org/10.5441/002/edbt.2014.63
36. Walmart (2015) Walmart uses Neo4j to optimize customer experience with personal recommendations. Technical reports, Neo Technology, https://go.neo4j.com/rs/710-RRC-335/images/neo4j-casestudy-walmart.pdf
37. Wang J, Ntarmos N, Triantafillou P (2016) Indexing query graphs to speedup graph query processing. In: Proceedings 19th International Conference Extending Database Technology, https://doi.org/10.5441/002/edbt.2016.07
38. Yoon BH, Kim SK, Kim SY (2017) Use of graph database for the integration of heterogeneous biological data. Genomics Inf 15(1):19–27. https://doi.org/10.5808/GI.2017.15.1.19
39. Zhang L, Zhu M, Huang W (2009) A framework for an ontology-based e-commerce product information retrieval system. J Comput 4(6):436–443. https://doi.org/10.4304/jcp.4.6.436-443