Developing a B2C e-Commerce Graph Data Model from a Relational Schema

Ilija Hristoski[1], Tome Dimovski[2], and Violeta Manevska[2]

Abstract – **The emerging NoSQL databases have capabilities that traditional relational databases do not possess. In this paper, we are focusing on NoSQL graph databases, and propose a graph data model that corresponds to a generic traditional relational database model found within the majority of today's B2C e-Commerce applications. The act of mapping relational databases to graph databases provides a solid basis for carrying out profound analyses of the implicit relationships among entity types' instances and gaining significant insights that cannot be incurred otherwise.**

Keywords – **E-Commerce, Relational schema, Graph data model, Mapping, Neo4j.**

I. Introduction

Nearly 50 years after Edgar F. Codd postulated the mathematical principles of the relational data model back in 1970, relational databases are still playing the role of a main workhorse in the contemporary digital landscape. There are many reasons for such an observation, e.g. relational databases are keeping data redundancy at a minimum; they are transaction friendly and highly consistent during update operations. Among major disadvantages of relational databases, which include the necessity to rebuild relationships with JOINs and other inexact techniques, their sensitivity to changes is, perhaps, of the utmost importance. The fact that relational databases are not designed to handle changes is crucial, having minded the nature of today's data, which is more dynamic, more intense, more voluminous, more unpredictable, more variable, and more diverse than ever before. Moreover, relational databases are neither designed for heterogeneous data, nor for scale and for mixed workloads, which make them a mismatch for modern app development.

Nowadays it becomes quite clear that relational databases cannot cope successfully with the challenges posed by today's data, and justifies what Vivek Kundra, the former CIO of the U.S. Federal Government, said in 2009: "This notion of thinking about data in a structured, relational database is dead."

Graph databases, on the other hand, belong to the family of NoSQL databases; they address one of the great macroscopic business trends of today: "leveraging complex and dynamic relationships in highly connected data to generate insight and competitive advantage" [1]. Unlike the relational databases, which store data to efficiently store facts, graph databases store both facts and the relationships among the facts, making certain types of analyses more efficient and intuitive. Emerging as a major driver of innovation during recent years, graph databases have already exhibited many advantages over relational databases, like storing large volumes of data that might have little to no structure, sharing data across multiple servers in the cloud, speeding the development, boosting the horizontal scalability, demonstrating superior performances, and supporting iterative algorithms and other data mining and machine learning algorithms.

The paper is organized as follows. Section 2 provides an insight into some of the most relevant research in this area. A generic B2C e-Commerce relational schema, which is used as a basis for this study, is presented in Section 3. In Section 4, the mapping of the B2C e-Commerce relational schema into a corresponding graph data model is described. Section 5 concludes.

II. Related Research

There is intense ongoing research in the area of mapping / converting / transforming / migrating relational databases into graph databases during recent years.

De Virgilio et al. propose a methodology for converting a relational to a graph database by exploiting their schema and the constraints of the source database. They also provide experimental results that demonstrate the feasibility of their approach and the efficiency of query execution against the target database [2]. Bordoloi & Kaita propose a method for transforming a relational database to a graph database model by transforming dependency graphs for the entities in the system into star graphs, which are then transformed into a hypergraph model [3]. Wardani & Küng propose a methodology for mapping and converting relational models to graph models without any semantic loss [4]. De Virgilio et al. present R2G, a tool for automatic migration of relational databases to a Graph Database Management System (GDBMS) [5]. Recognizing the fact that identifying correlations and relationships between entities within and across different data sets (or databases) is of a great importance in many domains, Lee et al. develop a reconfigurable and reusable graph construction tool, named *Table2Graph*, based on a Map-Reduce framework over Hadoop, with an aim of constructing a graph-based model from relational source databases [6]. Recently, Filho et al. proposed some heuristics, aiming at mapping systematization from relational model data to graph representation, in order to provide support for an adequate choice of the graph model, according to the type of analysis to be performed [7].

[1]Ilija Hristoski is with the Faculty of Economics at the "St. Kliment Ohridski" University in Bitola, 133 Gjorche Petrov St, 7500 Prilep, North Macedonia, E-mail: ilija.hristoski@uklo.edu.mk.

[2]Tome Dimovski and Violeta Manevska are with the Faculty of Information and Communication Technologies at the "St. Kliment Ohridski" University in Bitola, Partizanska St, 7000 Bitola, North Macedonia.

III. B2C E-COMMERCE RELATIONAL SCHEMA

In order to achieve the goal of this study, as a starting point, we use the generic B2C e-Commerce relational data model published by Williams in 2009 [8]. The database model consists of 14 tables, drawn from the following relational schema:

CUSTOMER
(**customer_id**, …)
CUSTOMER_PAYMENT_METHOD
(**customer_payment_id**, …, *customer_id**, *payment_method_code**)
REF_PAYMENT_METHOD
(**payment_method_code**, …)
PRODUCT
(**product_id**, …, *product_type_code**)
REF_PRODUCT_TYPE
(**product_type_code**, …, *parent_product_type_code**)
ORDER
(**order_id**, …, *customer_id**, *order_status_code**)
REF_ORDER_STATUS_CODE
(**order_status_code**, …)
ORDER_ITEM
(**order_item_id**, …, *product_id**, *order_id**, *order_item_status_code**)
REF_ORDER_ITEM_STATUS_CODE
(**order_item_status_code**, …)
INVOICE
(**invoice_number**, …, *order_id**, *invoice_status_code**)
REF_INVOICE_STATUS_CODE
(**invoice_status_code**, …)
PAYMENT
(**payment_id**, …, *invoice_number**)
SHIPMENT
(**shipment_id**, …, *order_id**, *invoice_number**)
SHIPMENT_ITEM
(**shipment_id**, **order_item_id**)

In the above relational schema, non-key attributes are being omitted, i.e. only table names, along with corresponding primary and foreign keys are given, due to simplicity reasons. Primary keys are bolded and underlined, whilst foreign keys are italicized.

IV. MAPPING THE RELATIONAL SCHEMA INTO A GRAPH DATA MODEL

Before transforming the relational database schema into a graph database model, it is convenient to point out the process of transformation, which can be carried out through a number of steps [9-11]:

- Each entity table is represented by a label on nodes;
- Each row in an entity table becomes a particular node;
- Columns on those tables become node properties;
- Technical primary keys should be removed, whilst keeping business ones;
- Unique constraints should be added for business primary keys;
- Indexes should be added for frequent lookup attributes;
- Each foreign key should be replaced with a relationship to the other table, and removed afterward from the original table;
- Data with default values should be removed, there is no need to store those;
- Data in tables that is denormalized and duplicated might have to be pulled out into separate nodes to get a cleaner model;
- Indexed column names might indicate an array property;
- Simple JOIN tables become relationships;
- Attributed JOIN tables become relationships with properties.

Five tables in the original relational schema: CUSTOMER, REF_ PAYMENT_METHOD, CUSTOMER, REF_ORDER_ STATUS_CODE, REF_ORDER_ITEM_STATUS_CODE, and REF_INVOICE_STATUS_CODE, which, besides other non-key attributes, does not include any foreign keys, but solely a primary key, can be directly converted into nodes. The names of these tables become node labels, whilst columns on those tables become node properties (Fig. 1)

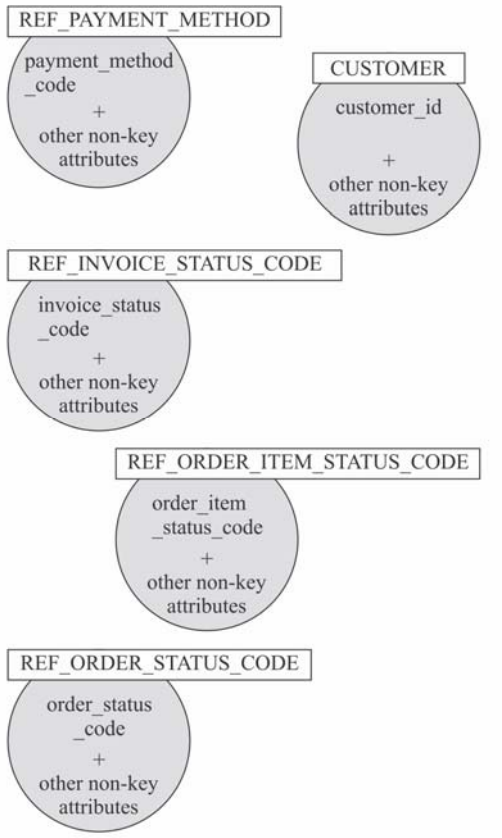

Fig. 1. Nodes coming out from the tables that do not include any foreign keys

Semantic relations between tables in the relational schema can be both identified and drawn from foreign keys, starting from the table with a foreign key(s) and following the relationship(s) to the table(s) where those foreign keys are primary keys (Table 1).

TABLE I
FINDING THE SEMANTIC RELATIONS BETWEEN TABLES

Semantic relation	Logical link (foreign key)
CUSTOMER_PAYMENT_METHOD – *BelongsTo* → REF_PAYMENT_METHOD	payment _method_code
CUSTOMER_PAYMENT_METHOD – *IsUsedBy* → CUSTOMER	customer_id
ORDER – *IsPlacedBy* → CUSTOMER	customer_id
ORDER – *HasOrderStatus* → REF_ORDER_STATUS_CODE	order_status _code
INVOICE – *IsIssuedFor* → ORDER	order_id
INVOICE – *HasInvoiceStatus* → REF_INVOICE_STATUS_CODE	invoice_status _code
PAYMENT – *IsMadeFor* → INVOICE	invoice_number
SHIPMENT – *CorrespondsTo* → ORDER	order_id
SHIPMENT – *IsCoveredBy* → INVOICE	invoice_number
PRODUCT – *IsOf* → REF_PRODUCT_TYPE	product_type _code
ORDER_ITEM – *RefersTo* → PRODUCT	product_id
ORDER_ITEM – *IsPartOf* → ORDER	order_id
ORDER_ITEM – *HasOrderItemStatus* → REF_ORDER_ITEM_STATUS_CODE	order_item _status_code
REF_PRODUCT_TYPE – *IsSubtypeOf* → REF_PRODUCT_TYPE	product_type _code

The logical links like *BelongsTo*, *IsUsedBy*, *IsPlacedBy*, etc., are implemented through the mechanism of foreign keys, but they are not obvious, because neither the existence nor the meaning of such semantic relations is not explicitly coded in the relational database schema. After identifying the semantic relations originating from the foreign keys, the latter ones are being removed from the tables. Those tables also become nodes, and the semantic relations originating from these become relations pointing towards corresponding nodes in a graph data model (Fig. 2).

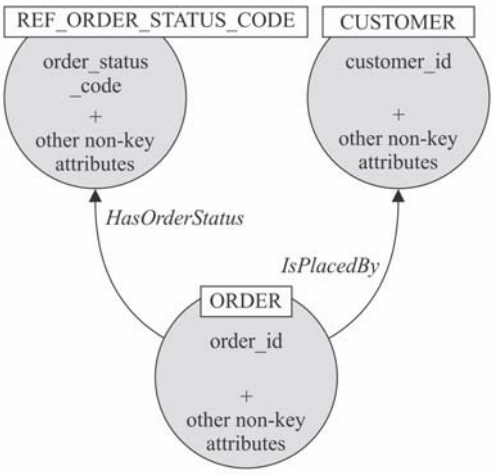

Fig. 2. Transforming foreign keys of the table ORDER into relations to nodes CUSTOMER and REF_ORDER_STATUS_CODE

The table REF_PRODUCT_TYPE is a specific one since it contains a foreign key that references its own primary key. This is a recursive relationship of cardinality 1:M. In this particular case, the foreign key transforms into a semantic relation *IsSubtypeOf* between two nodes with the same label, REF_PRODUCT_TYPE, which represent two distinct product types (Fig. 3).

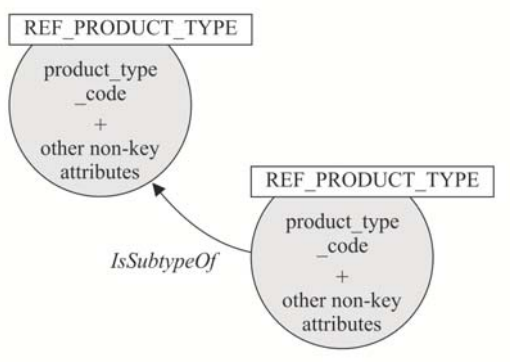

Fig. 3. Transforming the foreign key *parent_product_type_code** of the table REF_PRODUCT_TYPE into a semantic relation *IsSubtypeOf*

Finally, the table SHIPMENT_ITEM has been derived from an M:N relationship between the entity types SHIPMENT and ORDER_ITEM in the original E-R diagram. Since it represents a simple JOIN table without any additional attributes, it should not be represented as a node, but rather as a relationship, starting either from the node SHIPMENT to the node ORDER_ITEM, or vice-versa. Fig. 4 displays this relationship, named *Includes* (Fig. 4).

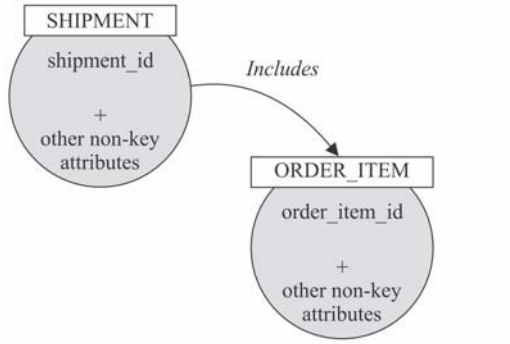

Fig. 4. Transforming the table SHIPMENT_ITEM into a semantic relation *Includes* between the nodes labeled SHIPMENT and ORDER_ITEM

The complete B2C e-Commerce graph data model has been implemented in Neo4j using the Cypher query language (CQL) and is given in Fig. 5. When it comes to Neo4j, it is worth pointing out that the Neo4j ETL (Extract-Transform-Load) tool allows one an automated translation of both relational data structures and actual data stored in an existing relational database into a graph data model. It includes a 3-step process that allows the user to specify a relational database via a JDBC setup, then to edit the data model

mapping that the tool creates for the graph, and finally, to import all of the data itself into a Neo4j database.

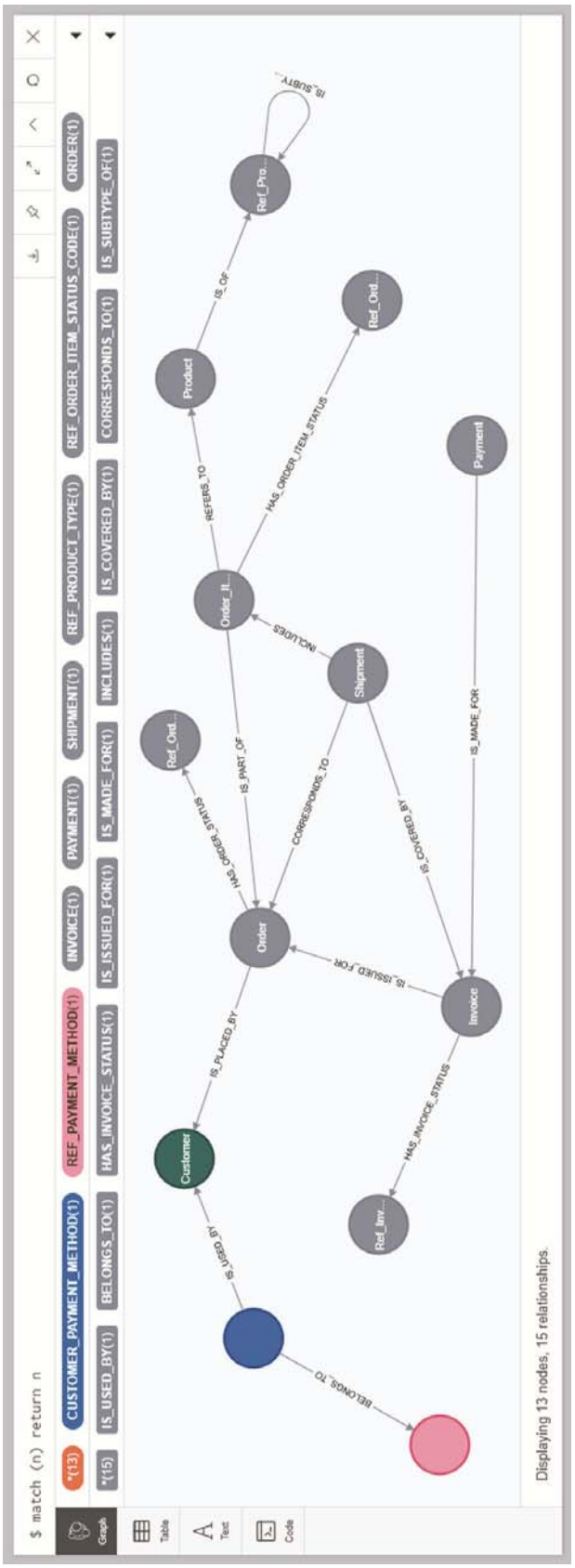

Fig. 5. The resulting B2C e-Commerce graph data model

V. Conclusion

The resulting graph data model differs from the original relational data model in several aspects: (1) There are no NULLs, i.e. non-existing value entries/properties are not present; (2) The graph model has no JOIN tables; (3) The graph model has no artificial primary keys or foreign keys; (4) Relationships are more detailed; (5) There is no need for intermediary tables, because they are all translated directly into relationships. The resulting graph data model captures the relationships among entity types in a highly intuitive and simplified manner because it is more expressive than any other corresponding relational data model. It provides a substantial basis for conducting advanced data analytics in a time-efficient manner, like e-Commerce recommendations, personalizations, security, real-time fraud detection, etc.

References

[1] I. Robinson, J. Webber and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, Second Edition, O'Reilly Sebastopol, CA, USA, 2015.

[2] R. De Virgilio, A. Maccioni and R. Torlone, "Converting Relational to Graph Databases", First International Workshop on Graph Data Management, Experience and Systems (GRADES 2013), Conference Proceedings, New York, NY, USA, pp. 1–6, 2013.

[3] S. Bordoloi, B. Kalita, "Designing Graph Database Models from Existing Relational Databases", International Journal of Computer Applications, vol. 74, no. 1, pp. 25–31, 2013.

[4] D. W. Wardani, J. Küng, "Semantic Mapping Relational to Graph Model", 2014 International Conference on Computer, Control, Informatics and Its Applications (IC3INA 2014), Conference Proceedings, Bandung, Indonesia, pp. 160–165, 2014.

[5] R. De Virgilio, A. Maccioni and R. Torlone, "R2G: a Tool for Migrating Relations to Graphs", 17th International Conference on Extending Database Technology (EDBT 2014), Conference Proceedings, Athens, Greece, pp. 640–643, 2014.

[6] S. Lee, B. H. Park, S-H. Lim and M. Shankar, "Table2Graph: A Scalable Graph Construction from Relational Tables Using Map-Reduce", 2015 IEEE First International Conference on Big Data Computing Service and Applications, Conference Proceedings, pp. 294–301, Redwood City, CA, USA, 2015.

[7] S. P. L. Filho, M. C. Cavalcanti and C. M. Justel, "Graph Modeling from Relational Databases", 2017 XLIII Latin American Computer Conference (CLEI 2017), Conference Proceedings, Cordoba, Argentina, pp. 843–842, 2017.

[8] B. Williams, "A Data Model for e-Commerce", 2009. URL: http://www.databaseanswers.org/data_models/e_commerce/index.htm (Accessed March 30, 2019)

[9] M. Hunger, R. Boyd and W. Lyon, *The Definitive Guide to Graph Databases for the RDBMS Developer*, e-Book, Neo Technology, 2016. URL: https://neo4j.com/whitepapers/rdbms-developers-graph-databases-ebook/ (Accessed April 02, 2019)

[10] M. Hunger, *From Relational to Graph: A Developer's Guide*, e-Book, DZone, Inc., 2016. URL: https://dzone.com/storage/assets/2054302-dzone-refcardz-231-neo4j.pdf (Accessed April 2, 2019)

[11] S. Yang, *How to Map Relational Data to a Graph DB in Four Steps*, e-Book, TIBCO Software Inc., Palo Alto, CA, USA, 2018. URL: https://www.tibco.com/sites/tibco/files/resources/sb-graph-database-final.pdf (Accessed April 4, 2019)