A Method for Modeling a Schema for Graph Databases

Peretz Shoval

Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev (BGU), Beer-Sheva, Israel;

Faculty of Computer Science, Netanya Academic College, Israel

1 Introduction

The recent decade had brought about massive growth in the Web and Web data - pictures, documents, videos and more. What worked well for years with relational databases is not well suited for the unstructured massive amounts of data and applications that are part of the Web, such as social networks. As a result, new types of database have emerged, including NoSQL ("not only SQL") databases. Graph databases are one type of NoSQL database.

Nowadays, there are only a few design methods for graph databases. Current approaches lack formal guidelines (i.e. full process description) and important data components. To address existing limitations we proposed a method for modeling graph database schema (GDBS) (Roy-Hubara, & et al, 2017). The method is based on an Entity-Relationship Diagram (ERD) of the domain of application, that is mapped to a graph database schema in two steps: in the first step, the original ERD, which includes constructs, such a ternary-relationships, that cannot be mapped directly to a graph database schema, is mapped to an equivalent ERD with alternative constructs that can be mapped to a graph database; in the second step, the adjusted ERD is mapped to the target graph database schema. The resulting schema is expressed in form of a diagram and as DDL statements that can be added to a future definition language of a graph database system.

2 The Graph Database Model

This section provides a brief description of the graph database model used in this study. For clarity, we use an example from the domain of movies recommendation system that provides information about movies, their actors, directors, etc., and also stores users' ratings of movies. Fig. 1 presents the ERD of the example.

Digital Presentation and Preservation of Cultural and Scientific Heritage. Conference Proceedings. Vol. 8, Sofia, Bulgaria: Institute of Mathematics and Informatics – BAS, 2018. ISSN: 1314-4006, eISSN: 2535-0366

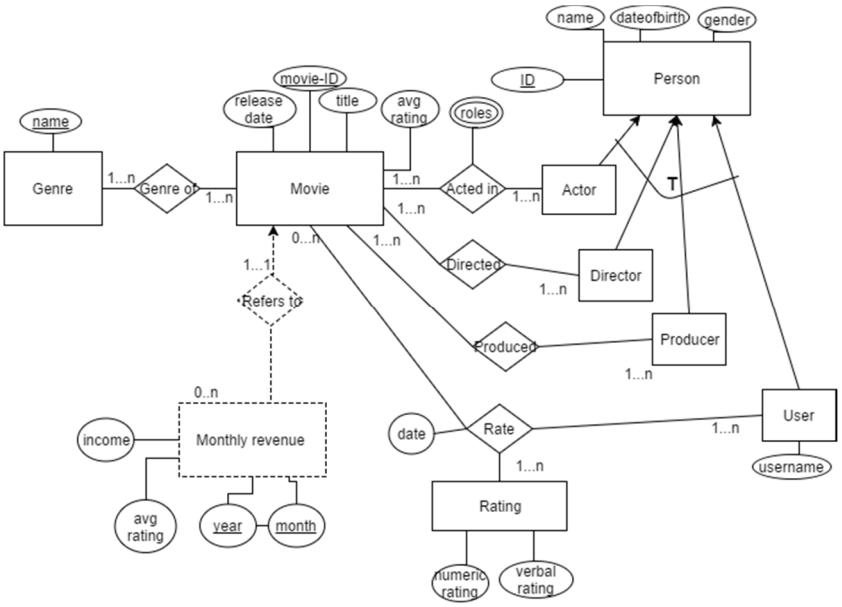

Fig. 1. ERD of a movies recommendation system

Components of a graph database model

The graph database model that we used consists of nodes, edges, and properties. The graphic notations of the components of this model are presented in Fig. 2.

Node: A node represents an entity in the real world. It has a label (name) and properties, including a key property which enables its unique identification. In Fig. 2 Movie is a node that is identified by the movie-id property.

Edge: An edge represents a binary relationship between two nodes. As in most graph databases, an edge is directed, meaning that it has a start node and an end node. An edge may also have properties and has a label. In Fig. 2 the edge GenreOf has start node Genre, and the end node Movie.

Property: As said, both nodes and edges may have properties. Properties may have constrains. Possible constraints include:

- Key: Each node has a key which may be one property or a combination of properties. For example, movie-ID is key of Movie, while movie-ID, year and month are key of Monthly Revenue.
- Not-Null: The property must have a value for all instances of its node or edge. For example, title is a Not-Null property of Movie (assuming that every movie must have a title).
- Set: The property may have many values. For example, 'roles' is a set property of the edge between Movie and Actor, since an actor may play many roles (this example is not included in Fig 2.).

Cardinality constraints: The graph database model we use extends existing graph database models by including cardinality constraints between the nodes of each edge. For this we use the same notations as in ERD; i.e., next to each node we write the min and max number of nodes that may participate in each edge type.

For example, Fig. 2 shows a one-to-many relationship between Movie and Monthly revenue, and a many-to-many relationship between Movie and Genre.

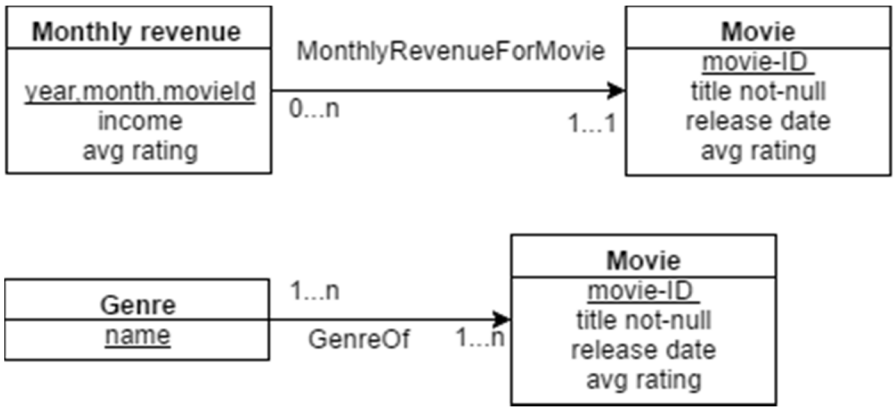

Fig. 2. Examples of nodes, edges, properties, and cardinality constraints

3 The Method for Modeling the Graph Database Schema

The modeling method is based on an ERD of the domain of application which is mapped to the GDBS in two steps: a) adjusting the original ERD to an equivalent ERD that is ready for mapping to a GDBS; and b) mapping the adjusted ERD to the GDBS.

3.1 Adjusting the Original ERD

An ERD may include constructs that cannot be mapped directly to a GDBS, which consists of only nodes and binary edges. Therefore, in the first step we adjust some of the original ERD constructs to an equivalent ERD constructs that can be mapped later on, as follows.

Ternary relationships: A ternary relationship is mapped to a weak entity, with binary relations to the entities involved in the ternary relation. If the relation has properties, they are added to the weak entity. The name of the weak entity may be identical to the name of the original ternary relationship, or be any name that resembles its role.

Aggregation (whole-parts) relationships: An aggregation relationship is mapped to an "ordinary" binary relationship, in which the cardinality of the parts entity is 0:n or 1:n (depending whether the parts entity is mandatory or not), while the cardinality of the whole entity is always 1:1.

Inheritance (is-a) relationships: The inheritance relationships are removed, and the super-entity is merged with the sub-entities. We distinguish between two cases:

a. *Removing the sub-entities and moving their attributes and relationships up to the super-entity*: This mapping is applied when the inheritance relationship is not defined with the "T" (Total cover) constraint and/or not defined with “X” (Exclusive), meaning that there may be super-entities which are not one of the sub-entities and/or that a super-entity may belong to many sub-entities. For example, Person is super-entity of four sub-entities; assuming that there is no "T" and no "X" constraints, the four sub-entities are removed, and their properties and relationships are added to Person.
b. *Removing the super-entity and moving its properties and relationships to each of its sub-entities*: This mapping is applied when there are both "T" and “X” constraints between the sub-entities, meaning that each of the super-entities belongs to one sub-entity only; therefore there is no need to maintain the super-entity.

3.2 Mapping the Adjusted ERD to the GDBS

The mapping process consists of the following steps:

Mapping entities to nodes:

Each entity is mapped to a node; the entity's properties become the node's properties.

A weak entity is mapped to a node just like an "ordinary" entity and the key property of this node is composed of the keys of the related "strong" entities of the weak entity, plus the partial key of the weak entity (if this exists).

Mapping relationships to edges:

Each relationship between entities is mapped to an edge connecting the two respective nodes: a start node and an end node. The edge's name may be the name of the relationship. It doesn’t matter which of the two nodes is defined as the start node and which is defined as the end node, because as said, the graph database enables traversing from node to node in any direction. Thus, the selection may be arbitrary.

Mapping cardinality constraints:

To the best of our knowledge, current graph databases do not define cardinality constraints, i.e., min and max number of nodes that may participate in an edge. For example, Neo4j, a leading graph database system, has not (yet) defined such constraints.

4 The Resulting GDBS

The resulting GDBS can be presented in form of a diagram and as DDL statements that can be added to a graph database system. Fig. 3 presents the diagram obtained after applying the mapping option of removing the sub-entities of the super-entity Person (according to Section 3.1.a).

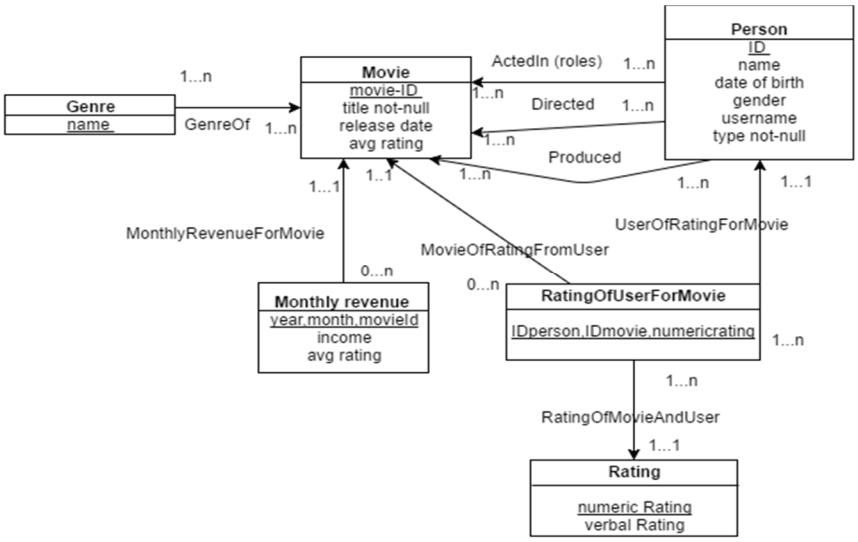

Fig. 3. The GDBS

5 Summary and Future Work

In (Roy-Hubara, & et al, 2017), we introduced a method for modeling a graph database schema, using a rich ERD that represents the domain of an application. The ERD is mapped in two steps to a graph database schema (GDBS) that preserves the semantics and constraints defined in the original conceptual schema.

In the order to evaluate the proposed method we conducted a controlled experiment in which we compared it with alternative methods, measuring differences in quality of the GDBS created by designers, time to complete the design task, and designers' satisfaction with the modeling method. The results of the experiment, which are reported in Roy-Hubara, & et al, 2018), showed differences in the quality of the created GDBS: subjects who used the proposed modeling method produced correct schemas, while subjects who used the ERD only made some errors, but fewer errors than the subjects of the control group who used "ad-hoc" method. With respect to time, we found significant differences: subjects who used the proposed method spent significantly more time than the subjects who used the other two methods.

In the future, we plan to repeat the controlled experiment, using more and homogeneous subjects, who are more trained with graph databases, and utilize design tasks of different size and complexity.

We also plan to implement the proposed method, i.e., to create a domain independent tool to be used as plug-in for graph database systems, that will enable to define a (initial) schema of the application. Such implementation might be adopted by graph database system providers.

Finally, we intend to extend the proposed schema modeling method to model other types of the NoSQL databases.

References

Roy-Hubara, N., Rokach, L., Shapira, B., & Shoval, P. (2017). Modeling Graph Database Schema. *IT Professional,* 19(6), 34-43, November/December 2017.

Roy-Hubara, N., Rokach, L. Shapira, B. & Shoval, P. (2018). Evaluation of a Design Method for Graph Database. *Proc. of EMMSAD-18 Conference*, Tallinn, Estonia, June 2018.

Received: June 15, 2018
Reviewed: June 29, 2018
Finally Accepted: July 08, 2018