Extracting Graphs from Tables via Conceptual Models

Sebastián Ferrada[1]

[1]IMFD Chile & Data and Artificial Intelligence Initiative, Universidad de Chile, Beauchef 851, Santiago, Chile

Abstract

This poster presents initial progress on a mapping to convert relational databases into knowledge graphs by utilizing the conceptual model of the database as a means of capturing its underlying semantics. We leverage the ERDoc language for defining Entity-Relationship Diagrams, for which we provide semantics. Unlike previous approaches, this method assumes the conceptual model as part of the input and emphasizes the formal definition, semantic correctness, and other properties of the mapping.

Keywords

Data Mapping, Knowledge Graphs, Relational Databases, Conceptual Models

1. Introduction

Knowledge graphs (KGs) model the data of a given domain as a set of entities or objects connected through a rich network of relationships [1]. Similarly, the conceptual model usually conceived when designing a relational database defines the types of entities that will inhabit the database and the types of relationships in which they can participate.

As most data is stored in relational databases, we propose leveraging their conceptual model to capture the underlying semantics and define a mapping procedure that produces a KG from its data. Such a mapping can be useful to be able to apply richer querying [2] and analytics [3] over the mapped data and, further, the mere definition of the mapping can allow for a virtual graph view of the relational data which can be accessed employing query translation.

In this poster, we present progress on the development of such a mapping that transforms a relational database into either an RDF/RDF-star graph or a property graph (PG), using the conceptual model of the input database. We leverage the ERDoc language [4] used to define Entity-Relationship Diagrams (ERDs) [5], which are a common way to design and communicate conceptual models. Our mapping, differently from Stoica et al. [6], is not direct (it requires extra input) but considers the semantics embedded in the conceptual model, yielding a semantically more accurate graph. For instance, in N-to-N relationships, [6] would create a node for each tuple in the table storing the relationship, whereas our approach translates such tuples directly to edges. Similarly, multivalued attributes would each be mapped to a node by [6], whereas our approach produces multivalued properties. Differently from Barret et al. [7], we assume that the conceptual model is part of the input of our mapping, and we shall focus on the formal definition of the mapping and its properties (e.g., information and query preserving [8]).

Posters, Demos, and Industry Tracks at ISWC 2024, November 13–15, 2024, Baltimore, USA

sebastian.ferrada@uchile.cl (S. Ferrada)

https://sferrada.com (S. Ferrada)

0000-0002-9834-8376 (S. Ferrada)

2. Conceptual Models

The conceptual model, defined as ERDs by Chen [5], aims to provide a unified view of data that allows to interoperate the relational model [9] and the network model [10]. As such, it also can allow us to leverage it to interoperate with RDF graphs [11], RDF-star graphs [12] or PGs [13].

ERDoc is a scripting language designed to specify ERDs. The idea for our mapping is to receive an input database along with the ERDoc document that codifies its conceptual model. To formalize our mapping, we provide semantics to a fragment of ERDoc [4].

Definition 1. *An Entity-Relationship Diagram (ERD)* $\mathcal{C}$ *is a tuple* $(\mathcal{E}, \mathcal{R})$ *such that:*

- $\mathcal{E}$ *is a set of* ***entities****. An* ***entity*** *is an expression of the form* $\mathsf{E}\{K_1, \dots, K_n\}\{A_1, \dots, A_m\}$*, where* $\mathsf{E} \in \mathcal{E}$ *is the name of the entity,* $\{K_1, \dots, K_n\}$*, is the non-empty set of prime attribute names of* E*, and* $\{A_1, \dots, A_m\}$ *is the possibly empty set of non-prime attribute names.*
- $\mathcal{R}$ *is a set of* ***relationships****. A* ***relationship*** $\mathsf{R}\langle(\mathsf{E_1}, C_1), \dots, (\mathsf{E_n}, C_n)\rangle\{A_1, \dots, A_m\}$ *is such that* $\mathsf{R} \in \mathcal{R}$ *is the relationship name,* $\mathsf{E_1}, \dots, \mathsf{E_n}$*, elements of* $\mathcal{E}$*, are the participating entities,* $C_1, \dots, C_n$ *are the cardinalities and participation constraints of each participating entity, and* $\{A_1, \dots, A_m\}$ *is the possibly empty set of relationship attribute names.*
- *A* ***weak entity*** $\mathsf{W}\langle(\mathsf{E_1}, \mathsf{R_1}), \dots, (\mathsf{E_l}, \mathsf{R_l})\rangle\{P_1, \dots, P_n\}\{A_1, \dots, A_m\}$*, is such that* $\mathsf{W} \in \mathcal{E}$ *is the name of the weak entity,* $\mathsf{E_i} \in \mathcal{E}$ *for* $i \in \{1, \dots, \mathsf{l}\}$ *are the names of the entities that* W *depends on,* $\mathsf{R_i} \in \mathcal{R}$ *for* $i \in \{1, \dots, \mathsf{l}\}$ *are the names of the relationships through which* W *depends on each* $\mathsf{E_i}$*,* $\{P_1, \dots, P_n\}$ *is the non-empty set of attribute names that are part of the partial key of* W*,* $\{A_1, \dots, A_m\}$ *is the possibly empty set of non-prime attribute names.*

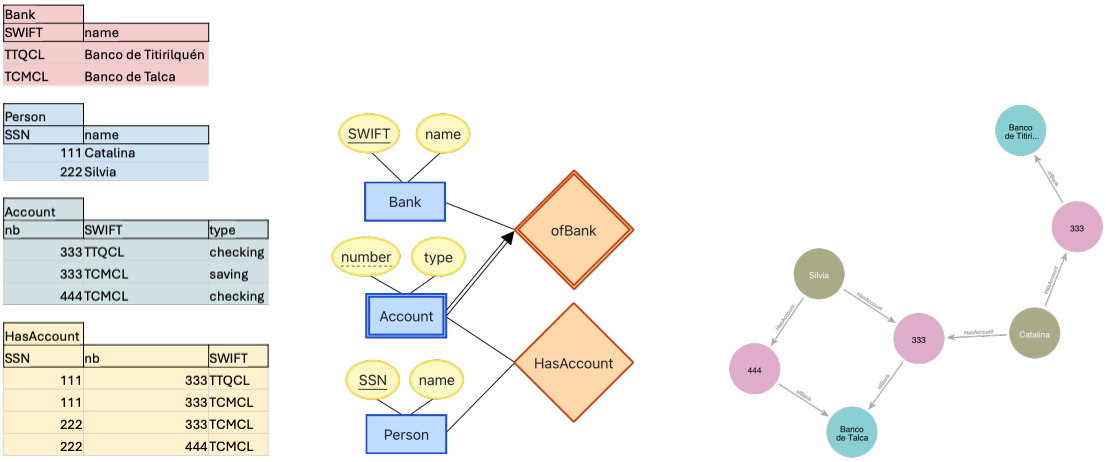
An example of an ERD can be found in Figure 1b, where Person and Bank are Entities, Account is a Weak Entity depending on Bank via relationship ofBank (which means that the primary key of Account is unique only for a given Bank), and hasAccount is an N-to-N relationship.

3. Mapping Relational Databases to Knowledge Graphs

Our mapping $\mathcal{M}$ is such that, given a relational database D with primary and foreign keys Σ (following the definition of Sequeda et al. [8] over a domain of values $\mathcal{V}$), and an ERD $\mathcal{C}$, $\mathcal{M}(\mathsf{D}, \mathcal{C})$ returns a PG $G = (V, E, \sigma, \phi, \lambda)$ (following the definition of Angles [13]). RDF graphs can also be produced later on (e.g., by using the mapping of [14]). We assume that D is in BCNF [15].

Non-weak Entities $\mathsf{E}\{K_1, \dots, K_n\}\{A_1, \dots, A_m\}$ are mapped to a relation $R_\mathsf{E} \in \mathsf{D}$, such that the attributes of R_E are $att(R_\mathsf{E}) = \{K_1, \dots, K_n, A_1, \dots, A_m\}$, and $R_\mathsf{E}[K_1, \dots, K_n]$ is a primary key. To map such a relation to a graph, we take each tuple $t \in R_\mathsf{E}$, where $t = (K_1 \colon v_1, \dots, K_n \colon v_n, A_1 \colon v_{n+1}, \dots, A_m \colon v_{n+m})$, with $v_1, \dots, v_n \in \mathcal{V}$ and $v_{n+1}, \dots, v_{n+m} \in \mathcal{V} \cup \{\texttt{null}\}$, and map it to a node $\eta = f_n(v_1, \dots, v_n)$, where $f_i : \mathcal{V}^i \to \mathcal{V}$ is a function that returns identifiers . Then, we extend the property assigning function σ to contain $\sigma(\eta, p) = v$ for each $p \colon v \in t$. Finally, we extend the label assigning function λ to include $\lambda(\eta) = R_\mathsf{E}$. It can be seen that if f_i is bijective, this mapping is reversible, and thus *information preserving* [8].

The mapping of relationships to the relational model is more nuanced. It depends on the number of participating entities, the presence or absence of attributes, and even the cardinalities



(a) The Tables. (b) The ERD. (c) The Graph.

Figure 1: The proposed mapping process.

and participation constraints. We will summarize two acceptable mappings that preserve BCNF. Let us consider the generic relationship $\mathsf{R}\langle(\mathsf{E_1}, C_1), \ldots, (\mathsf{E_n}, C_n)\rangle\{A_1, \ldots, A_m\}$.

If $n = 2$, $\{A_1, \ldots, A_m\} = \emptyset$, and either C_1 or C_2 are *one and only one* cardinalities, the first mapping applies. W.l.o.g., we assume that C_2 is *one and only one*, and that the participating entities are $\mathsf{E_1}\{K_1, \ldots, K_{n_1}\}\{B_1, \ldots, B_{m_1}\}$ and $\mathsf{E_2}\{J_1, \ldots, J_{n_2}\}\{F_1, \ldots, F_{m_2}\}$, which are mapped to relations $R_{\mathsf{E_1}}$ and $R_{\mathsf{E_2}}$ respectively. To map this relationship, we extend $R_{\mathsf{E_2}}$ so that $att(R_{\mathsf{E_2}}) \leftarrow att(R_{\mathsf{E_2}}) \cup \{K_1, \ldots, K_{n_1}\}$ and $R_{\mathsf{E_2}}[K_1, \ldots, K_{n_1}]$ REF $R_{\mathsf{E_1}}[K_1, \ldots, K_{n_1}]$ is a foreign key. To convert this foreign key to graph elements, for each tuple $t \in R_{\mathsf{E_2}}$, where $t = (K_1\colon v_1, \ldots, K_{n_1}\colon v_{n_1}, J_1\colon v_{n_1+1}, \ldots, J_{n_2}\colon v_{n_1+n_2}, F_1\colon v_{n_1+n_2+1}, \ldots, F_{m_2}\colon v_{n_1+n_2+m_2})$ we create an edge $e = f_{n_1+n_2}(v_1, \ldots, v_{n_1+n_2})$, with $\phi(e) = (f_{n_1}(v_1, \ldots, v_{n_1}), f_{n_2}(v_{n_1+1}, v_{n_1+n_2}))$, and $\lambda(e) = \mathsf{R}$. Yet again, if f_i is bijective, then this transformation is reversible.

For any other case of relationship R (e.g., $n > 2$ or $\{A_1, \ldots, A_m\} \neq \emptyset$), a relation R_{R} for the relationship is created and it contains foreign keys referencing all the relations mapping the participating entities. The primary key of R_{R} depends on the cardinalities and participation constraints [5]. Each tuple $t \in R_{\mathsf{R}}$ can be mapped to an edge only if $n = 2$. Otherwise, t should be mapped to a node, and an edge for each foreign key must be created, as is done in [6].

Weak entities $\mathsf{W}\langle(\mathsf{E_1}, \mathsf{R_1}), \ldots, (\mathsf{E_l}, \mathsf{R_l})\rangle\{P_1, \ldots, P_n\}\{A_1, \ldots, A_m\}$ are mapped into a relation R_{W}, similar to the first relationship case, as the cardinality with which W participates in every $\mathsf{R_i}$ is *one and only one*. R_{W} has attributes $att(R_{\mathsf{W}}) = \mathcal{K} \cup \{P_1, \ldots, P_n, A_1, \ldots, A_m\}$, where $\mathcal{K}$ is the set of all the prime attributes of all the relations $R_{\mathsf{E_i}}$ of each $\mathsf{E_i}$. R_{W} has therefore foreign keys referencing to each $R_{\mathsf{E_i}}$. Each tuple $t \in R_{\mathsf{W}}$ is mapped to a node η with id $f_{|\mathcal{K}|+n}(t[\mathcal{K} \cup \{P_1, \ldots, P_n\}])$, label $\lambda(\eta) = \mathsf{W}$, and each foreign key is mapped to an edge without properties and the label of the respective relationship $\mathsf{R_i}$ going from η to the respective node representing the referenced tuple in $R_{\mathsf{E_i}}$.

Example. In Figure 1a, we present 4 relations that follow the ERD of Figure 1b. These are Person(SSN, name), Bank(SWIFT, name), Account(number, SWIFT, name), and HasAc-

count(SSN, number, SWIFT). Note that Account is a weak entity and HasAccount is an N-to-N relationship. The tuples in Figure 1a are translated to the graph of Figure 1c, following the rules of each case. See how, for instance, the tuple (number: 333, SWIFT: TTQCL, type: checking) from relation Account is mapped to a node with id $\eta = f_1(333)$, label $\lambda(\eta) =$ "Account", properties $\sigma(\eta, \text{number}) = 333$ and $\sigma(\eta, \text{type}) =$ "checking", and to and edge e, such that $\phi(e) = (\eta, f_1(\text{"TTQCL"}))$ and $\lambda(e) =$ "ofBank". This mapping is similar to [6]. However, we can extract the appropriate label for e from the conceptual model. Our mapping presents its difference particularly when mapping the relation HasAccount. The tuple (SSN: 111, number: 333, SWIFT: TTQCL) is mapped, according to [6], to a node with label "HasAccount", with one edge to the node mapping the person with ID $f_1(111)$, and another to the node of the Account with ID $f_2(333, \text{TTQCL})$. Our mapping simply creates **one** edge e', with $\lambda(e') =$ "HasAccount", and $\phi(e') = (f_1(111), f_2(333, \text{TTQCL}))$. This is not only more semantically accurate but also implies the use of fewer joins when querying the resulting graph.

4. Conclusion and Future Directions

In this poster, we present initial progress on a formal mapping that, given a relational database and its conceptual model, produces a knowledge graph that behaves differently from [6]. Initially, we consider property graphs, but RDF and RDF-star graphs can also be produced. Further, we provide semantics for a fragment of the ERDoc language and a formalization of the elements present in an ERD. We are currently defining the translations of the rest of the ERD constructs (class hierarchies, aggregations, multivalued attributes, etc.) and studying the general properties of information and query preservation [8] of the mapping. We note that in the future, we may leverage the work by Barret et al. [7] to automatically obtain the conceptual model for the mapping. We are also working on a mapping algorithm and a tool implementation. Furthermore, we can explore using the conceptual model to generate SHACL constraints [16] to validate the mapping output.

Acknowledgments

Partly funded by ANID, Millennium Science Initiative Program, Code ICN17_002.

References

[1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge Graphs, volume 22 of *Synthesis Lectures on Data, Semantics, and Knowledge*, Springer Nature, 2021.

[2] L. Libkin, D. Vrgoč, Regular path queries on graphs with data, in: Proceedings of the 15th International Conference on Database Theory, ACM, Berlin Germany, 2012, pp. 74–85. doi:10.1145/2274576.2274585.

[3] A. Hogan, J. L. Reutter, A. Soto, In-Database Graph Analytics with Recursive SPARQL, in: The Semantic Web – ISWC 2020, volume 12506, Springer International Publishing, Cham, 2020, pp. 511–528. doi:10.1007/978-3-030-62419-4_29.
[4] M. López, S. Ferrada, A. Hogan, ERDoc: A Web Interface for Entity-Relation Modelling, in: Proceedings of the 3rd International Workshop on Data Systems Education, ACM, 2024.
[5] P. P.-S. Chen, The entity-relationship model—toward a unified view of data, ACM Transactions on Database Systems 1 (1976) 9–36. doi:10.1145/320434.320440.
[6] R. Stoica, G. Fletcher, J. F. Sequeda, On directly mapping relational databases to property graphs, in: 13th Alberto Mendelzon International Workshop on Foundations of Data Management, AMW 2019, CEUR-WS. org, 2019, p. 06.
[7] N. Barret, I. Manolescu, P. Upadhyay, Computing Generic Abstractions from Application Datasets, in: EDBT 2024 - 27th International Conference on Extending Database Technology, volume 27, 2024, pp. 94–107.
[8] J. F. Sequeda, M. Arenas, D. P. Miranker, On directly mapping relational databases to RDF and OWL, in: Proceedings of the 21st International Conference on World Wide Web, ACM, 2012, pp. 649–658. doi:10.1145/2187836.2187924.
[9] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387. doi:10.1145/362384.362685.
[10] C. W. Bachman, Data structure diagrams, ACM SIGMIS Database: the DATABASE for Advances in Information Systems 1 (1969) 4–10. doi:10.1145/1017466.1017467.
[11] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, B. McBride, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, W3C, 2014.
[12] O. Hartig, Foundations of RDF* and SPARQL* : (An Alternative Approach to Statement-Level Metadata in RDF), in: 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web., volume 1912, Montevideo, Uruguay, 2017.
[13] R. Angles, The Property Graph Database Model, in: Proc. Alberto Mendelzon International Workshop on Foundations of Data Management (AMW), volume 2100, CEUR Workshop Proceedings, 2018.
[14] O. Hartig, Foundations to query labeled property graphs using SPARQL, in: Joint Proceedings of the 1st International Workshop on Semantics for Transport and the 1st International Workshop on Approaches for Making Data Interoperable, volume 2447, CEUR Workshop Proceedings, 2019.
[15] I. J. Heath, Unacceptable file operations in a relational data base, in: Proceedings of the 1971 ACM SIGFIDET Workshop on Data Description, Access and Control - SIGFIDET '71, ACM Press, San Diego, California, 1971, p. 19. doi:10.1145/1734714.1734717.
[16] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), Technical Report, W3C, 2017. URL: https://www.w3.org/TR/shacl/.