RDB2Graph: A Generic Framework for Modeling Relational Databases as Graphs

Kang Min Yoo, Sungchan Park, and Sang-goo Lee

Intelligent Data Systems Laboratory
Seoul National University
{kangminyoo, baksalchan, sglee}@europa.snu.ac.kr

Abstract. Graph data mining is highly versatile, as it applies not only to graph data but to relational data, as long as it can be represented as pairs of relationships. However, modeling RDBs as graphs using existing methods is limited in describing semantics of the relational data. In this paper, we propose a two-phased graph-modeling framework that converts any RDB to a directed graph with richer semantics than previously allowed. We implemented the framework and used it for analyzing medical records of diabetes patients.[1]

Keywords: relational database, graph, graph modeling

1 Introduction

Graph data mining is a well-studied area of research because of numerous applications in fields such as social data mining and biochemical analysis [1]. Recently, there has been a growing interest in applying graph mining techniques to relational databases, as viewing them as graph data exposes inherent semantics [3] [4]. Since relational databases are a de facto standard in data warehousing, applying graph mining techniques to mine latent information from the massive relational data seems even more attractive.

However, modeling relational databases as graphs is not straight-forward, because it is challenging to devise appropriate models that expose underlying semantics. W3C formalized the graph-modeling process by defining a new language, R2RML [2], but the language is limited in describing some semantic aspects of graph conversion. For example, it cannot used to describe a vertex that combines several attributes (fig. 1). It also fails at describing semantics not apparent in relational schemata, such as events that could be connected in chronological order (fig. 2).

We propose a new framework to solve the current problem of modeling RDBs as graphs. The framework converts any RDB into a directed graph given some conversion rules (sec. 2). We have also implemented a program based on RDB2Graph (R2G) to analyze medicals records of diabetes patients.

[1] This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea Government(MSIP) (No. 20110030812).

2 RDB2Graph

Given a **relational database** D and a set of modeling rules, the framework generates a directed graph. The output of the framework is **edge set** E and **vertex set** V. Each vertex is a set of key-values, i.e. $\{(k_1, l_1), (k_2, l_2), \ldots, (k_n, l_n)\}$, where each key in $k_1, \ldots, k_n$ corresponds to some attribute and each l in $l_1, \ldots, l_n$ is some value in D. An edge e is directed and denoted by (v_s, v_t), where v_s is the source vertex and v_t is the target vertex. Both v_s and v_t are in V. Given some rule set Γ and D, framework must return E and V such that elements satisfy the specifications of Γ.

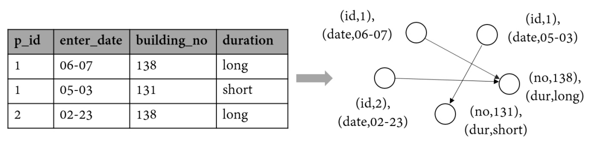
2.1 The Two-Phase Conversion

The conversion takes place in two phases — the first phase discovers relationships within each tuple, while the second phase establishes additional relationships among the vertices constructed from the first phase. Thus, Γ is an union of Γ_1 and Γ_2, the rules for both phases respectively. Splitting in two phases is necessary because it allows incorporation of implicit relationships not apparent from the relational database itself.

Phase I: Tuples to Edges In order to create new vertices and edges from the relational database, a set of rules Γ_1 must specify the followings:

1. a target relation (r)
2. two sets of attributes (C_s, C_t) from relation r to indicate which values of each tuple in r are stored as l in key-value pairs of source or target vertices
3. two sets of key aliases (K_s, K_t) to indicate k in key-value pairs of source or target vertices.
4. a bit (b) to indicate whether the edge is bidirectional or not
5. a selection predicate (p) to filter tuples.

For each c in C_s or C_t, the corresponding value u in each tuple is paired with the corresponding k in K_s or K_t to form a key-value pair (k, u), which is added to the generated source or target vertex. An example of edge generation is presented in figure 2.1.



p_id	enter_date	building_no	duration
1	06-07	138	long
1	05-03	131	short
2	02-23	138	long

Fig. 1. Phase I edge construction based on the rule $C_s = \{p\_id, enter\_date\}$, $C_t = \{building\_no, duration\}$, $K_s = \{id, date\}$, $K_t = \{no, dur\}$, and $b = 0$.

The purpose of specifying the alias sets K_s and K_t is to combine semantically identical attributes together. For example, consider a case where relation r_1 has a foreign key constraint that references a primary key of another relation r_2. Their attribute names might be different, but they are identical semantically. By having the ability to give a common key for the attributes, we are able to generate common vertices. It is apparent from figure 2.1 that some vertices have common key configurations (e.g. $\{id, date\}$). We call such key configurations **vertex schemata**.

Selection predicate p is similar to the counterpart of relational algebra. It can be directly used in SQL queries to filter out tuples. For each rule, the framework retrieves a set of tuples from T that satisfy the predicate and produces exactly one edge of (v_s, v_t), which is added to E. The framework will also attempt to add v_s and v_t into V, if they do not exist already. At the end of phase I, E and V are generated and passed to phase II.

Phase II: Vertices to Edges Given the graph constructed from the previous phase, E and V, and phase II rules Γ_2, the framework constructs additional edges that satisfy Γ_2. In this phase, each rule of Γ_2 specifies the followings.

1. two sets of vertex schema (K_s, K_t)
2. a bit (b) to indicate whether the generated edge is bidirectional or not
3. a vertex selection predicate (q) to filter vertices

For each rule in Γ_2, vertices with schema K_s or K_t are considered as source or target vertices of an additional edge. Given the vertices, the framework further filters them using vertex selection predicate q. An example of such edge generation is shown in figure 2.1.

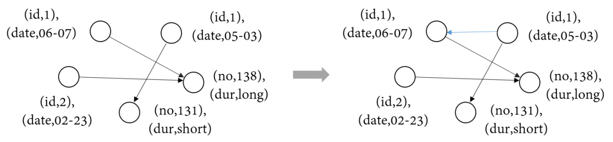
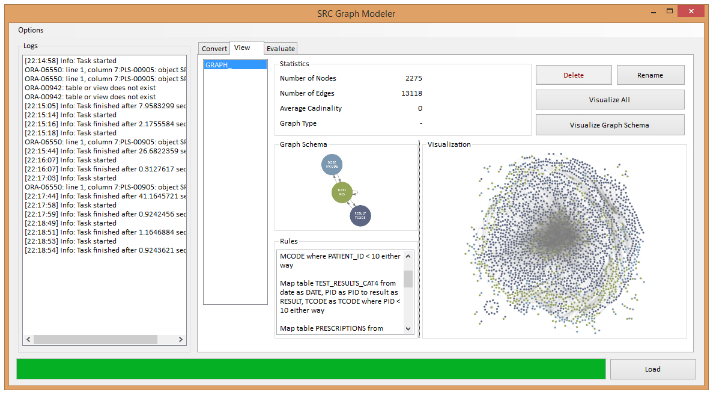


Fig. 2. Phase II edge construction based on the rule $K_s = \{id, date\}$, $K_t = \{id, date\}$, $b = 0$, and q selects source and target vertices such that values of id are equal but the target date is the nearest later date to the source date.

q is different from p of Γ_1 — in p, left-hand side variables are references to some attributes of a relation; in q, left-hand side variables are references to some keys of either K_s or K_t. Phase II produces E and V as well, but with additional edges that satisfy Γ_2.



```
[22:14:58] Info: Task started
ORA-06550: line 1, column 7:PLS-00905: object S
ORA-06550: line 1, column 7:PLS-00905: object S
ORA-00942: table or view does not exist
ORA-00942: table or view does not exist
[22:15:05] Info: Task finished after 7.9583299 sec
[22:15:14] Info: Task started
[22:15:16] Info: Task finished after 2.1755584 sec
[22:15:18] Info: Task started
ORA-06550: line 1, column 7:PLS-00905: object S
[22:15:44] Info: Task finished after 26.6822359 se
[22:16:07] Info: Task started
[22:16:07] Info: Task finished after 0.3127617 sec
[22:17:03] Info: Task started
ORA-06550: line 1, column 7:PLS-00905: object S
[22:17:44] Info: Task finished after 41.1645721 se
[22:17:58] Info: Task started
[22:17:59] Info: Task finished after 0.9242456 sec
[22:18:49] Info: Task started
[22:18:51] Info: Task finished after 1.1646884 sec
[22:18:53] Info: Task started
[22:18:54] Info: Task finished after 0.9243621 sec
```

Number of Nodes	2275
Number of Edges	13118
Average Cadinality	0
Graph Type	-

MCODE where PATIENT_ID < 10 either way

Map table TEST_RESULTS_CAT4 from date as DATE, PID as PID to result as RESULT, TCODE as TCODE where PID < 10 either way

Map table PRESCRIPTIONS from

Fig. 3. SRCGraphModeler is an implementation of RDB2Graph. It was used to analyze medical records of diabetes patients.

2.2 Implementation

Our implementation of the framework, SRCGraphModeler (SGM), is written in C#, and it connects to Oracle 11g based RDB. SGM converts each rule into PL/SQL procedures in order to run them on the database server, improving runtime efficiency. Using SGM, we converted medical records of diabetes patients into various graph models, then we applied graph analysis on the graphs to extract correlation among medications and symptoms.

3 Conclusion

We have presented a graph-modeling framework that enables semantically richer conversions than that attempted by previous works. As future works, we plan to study how transformation of RDB data to graph data affects the information contained in it.

References

1. Charu C. Aggarwal and Haixun Wang. *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*. Springer, 2010.
2. Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language. http://www.w3.org/TR/r2rml, September 2012.
3. Jaehui Park. *A Graph-based Framework for Processing Keyword Queries over Relational Databases*. PhD thesis, Seoul National University, 2012.
4. Subhesh Pradhan, Sharma Chakravarthy, and Aditya Telang. Modeling Relational Data as Graphs for Mining. In *International Conference on Management of Data*, 2009.