An ER-Flow Diagram for Big Data

Carlos Ordonez
University of Houston[§]
USA

Sikder Tahsin Al-Amin[*]
University of Houston[§]
USA

Ladjel Bellatreche
LIAS/ISAE-ENSMA
Poitiers, France

***Abstract*—ER diagrams have a proven track record to represent data structure and relationships, in many CS problems, beyond relational databases. The ER diagram strengths are abstraction, generality, flexibility, and intuitive visual representation, with few weaknesses; hence its popularity. The main con is the old box-diamond-ellipse-line notation, which has been subsumed by the more modern and simpler UML box-line notation. Given the broad, varied, and dynamic nature of big data ER diagrams are mostly ignored, except when the data sources are databases. It is common wisdom raw big data needs significant pre-processing before computing any analytics, resulting in a long chain of data transformations computed in SQL, Python, or R languages, for instance. On the other hand, flow diagrams remain the main mechanism to visualize major components of a software system or main processing steps of an algorithm, showing rectangles (verbs) connected by arrows (processing order, dependence). In this work, we propose to combine both diagrams into one. We propose a hybrid diagram, which we call ER-Flow, based on modern UML notation, to assist analysts in data pre-processing and exploration. Aiming to introduce a minimal change to the ER diagram, we extend relationships lines with an arrow, indicating processing flow and we annotate entities coming from pre-processing with numbers and transformation labels. We illustrate how our novel ER-Flow diagram can help the user navigate big data at the metadata level, providing an integrated view of data and source code, with many practical benefits.**

***Index Terms*—ER Diagran, Big Data, Metadata, Data Science, Data Transformation**

I. Introduction

A significant effort is required to pre-process data sets in big data analytics because data sets come from diverse sources, they have different structure, they come in different file formats and they are not integrated. Hence data analysts need to collect, integrate, clean, merge, aggregate, and transform data files before they can perform analysis. These data transformations create many intermediate files, tables in a disorganized manner. ER diagrams have a proven track record to represent data structure and relationships, beyond databases. The ER diagram strengths are generality, flexibility, and intuitive visual representation. On the other hand, flow diagrams remain the main mechanism to visualize major components or main processing steps of a software system, but they are less useful to understand complex algorithms. Several closely related works on ER diagram [1], [2], Flow diagram [3], [4], and data transformation [5], [6] have been done by other researchers. In this work, we defend the idea of combining both of them, but making emphasis on the "data" angle. We propose a hybrid diagram, which we call ER-Flow, to assist big data analysts in data pre-processing.

[§]Department of Computer Science, University of Houston, Houston TX 77204, USA
[*]Contact Author: stahsin.cse@gmail.com

II. Definitions

Let $E = \{E_1, ..., E_n\}$, be a set of n entities, linked by relationships. We follow modern UML notation, where entities are represented by rectangles and relationships are shown by lines, with crowfeet on the "many" side. Each entity has a list of attributes, where each attribute can be atomic or multivalued. There exists an identifying set of attributes for each entity (i.e. a primary key, an object id). Intuitively, entities correspond to objects in real life and relationships to actions. Therefore, we will use nouns as entity names and verbs to describe relationships.

Diagram notation extension: We allow relationships to have an arrow on one side indicating data flow direction. This direction can also be interpreted as input and output, going from input entities to output entities. This is a minor, yet powerful, change that enables navigating all data elements in the data lake, as well as having a data-oriented flow of big data processing.

III. Proposed ER-Flow Diagram

Our entity concept is broad: entities can represent a file, matrix, relational table, or dataframe. That is, we go beyond relational databases.

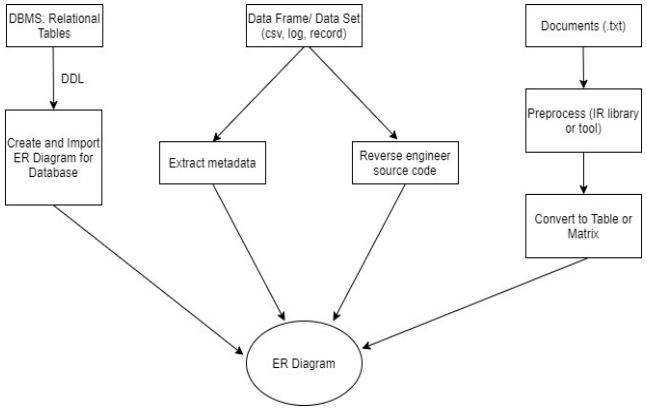

Fig. 1: Generating preliminary ER-Flow diagram.

Entities are classified as source (raw) entities, representing raw data, loaded into the Data Lake and Transformation

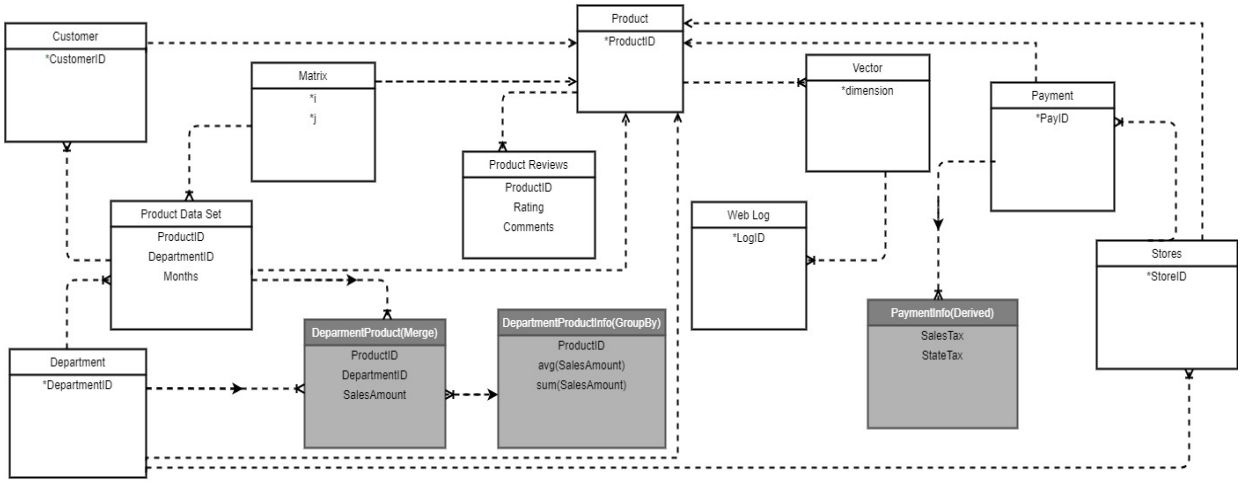

Fig. 2: ER-Flow diagram for a Store Data Lake.

(pre-processing) entities being the output of some tool or programming language (Python, R, SQL).

We focus on representing data transformations for big data analytics, including machine learning, graphs, and even text files (documents). However, our diagram does not represent the "analytic output" such as the parameters of the ML model, IR metrics like precision/recall, graph metrics. We propose these three major categories of data transformations:

1) Merge, which splices multiple entities by some attribute, which is a generalized relational join operator.
2) GroupBy, which partition and aggregates records based on some criteria. Every Data Science language provides operators or functions highly similar to the SQL group by clause.
3) Mathematical, which represent derived attributes coming from a combination of functions and value-level operators (e.g. equations, arithmetic expression, nested function calls).

First, our solution generates a preliminary ER-Flow diagram as follows. This diagram can be polished and customized by the analyst. We show this process in Fig 1.

1) Importing ER models available from databases. We assume ER diagrams are available or can be easily constructed for a relational database DDLs or exported from an ER diagram tool as CSV files.
2) Automatic entity and attribute identification from meta-data embedded in the file itself. In particular, we assume CSV files are the default format for spreadsheet data, logs and mathematical software. On the other hand, JSON is another standard file format to exchange data.
3) For plain text files like documents, source code we assume they contain strings for words, numbers, symbols and so on. In this case, we some IR library or tool will pre-process the file and convert it to generic tables or matrices. Then we propose to extract entity and attribute names from the final table or matrix.
4) Automatic data set and attribute name identification for data sets built by Python, R, or SQL code, generalizing a previous approach with SQL queries [7], kind of "reverse engineering".

The diagram data is stored in two JSON files, where the first file contains the relationships and the second one contains the entities and their attributes. In the transformation module, we define the transformation type and create new transformed entities. The analysts may perform several transformations discussed above in the source code that generates a temporary entity. In the case of "Merge", the entity structure may change but the attribute values remain the same, and the "Group by" may use one or more grouping attributes along with or without aggregations (sum, count, avg). In general, aggregations will return numbers, but using only "Group by" will return the attribute values as their types. Mathematical transformations will mostly return derived attributes. Now, the new transformed entities are linked with the original entities using an arrow. After each valid transformation step, we can store the newly generated ER-Flow diagram in JSON files. This ER-Flow diagram can help the analysts to have data-oriented view of the program, navigate source code, reuse functions, and avoid creating redundant data sets.

We show an example in Figure 2 where we show our final ER-Flow diagram. We consider an example of a store for which we show the ER-Flow diagram. In our example the target analytic is a predictive model of product sales considering history sales data, customer information and buyers' opinions. The goal is to produce a data set, which can be used as input for a predictive model like regression, decision trees, SVMs or deep neural networks. Each entity from the original data has an identifying attribute (primary key) and other attributes. From these entities, analysts can generate new entities by doing data transformations as mentioned above. Popular analytic languages like Python and R, both support data transformations (ex: "Merge", "Group by") in pandas and dpylr libraries respectively. Each of the transformations generates a new entity which is named from the input entities and the transformation type is shown inside

the parenthesis as "(TYPE)". The source entities are colored white and the transformed entities are colored grey for better understanding. We can see the flow of the transformed entities as they are linked with an arrow from the source entities.

REFERENCES

[1] J. Mugan, R. Chari, L. Hitt, E. McDermid, M. Sowell, Y. Qu, and T. Coffman, "Entity resolution using inferred relationships and behavior," in *2014 IEEE International Conference on Big Data*. IEEE Computer Society, 2014, pp. 555–560.

[2] G. Guo, "An active workflow method for entity-oriented data collection," in *Advances in Conceptual Modeling - ER 2018 Workshops Emp-ER, MoBiD, MREBA, QMMQ, SCME, Xi'an, China, October 22-25, 2018, Proceedings*.

[3] C. Batini, E. Nardelli, and R. Tamassia, "A layout algorithm for data flow diagrams," *IEEE Trans. Software Eng.*, vol. 12, no. 4, pp. 538–546, 1986.

[4] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell, "Pattern-based analysis of the control-flow perspective of UML activity diagrams," in *Conceptual Modeling - ER 2005*, vol. 3716, pp. 63–78.

[5] M. Sebrechts, S. Borny, T. Vanhove, G. van Seghbroeck, T. Wauters, B. Volckaert, and F. D. Turck, "Model-driven deployment and management of workflows on analytics frameworks," in *2016 IEEE International Conference on Big Data*, 2016, pp. 2819–2826.

[6] M. Pham, C. A. Knoblock, and J. Pujara, "Learning data transformations with minimal user effort," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 657–664.

[7] D. Lanasri, C. Ordonez, L. Bellatreche, and S. Khouri, "ER4ML: an ER modeling tool to represent data transformations in data science," in *Proceedings of the ER Forum and Poster & Demos Session 2019*, vol. 2469, pp. 123–127.