FLOWER: Data Flow in ER Diagrams

Carlos Ordonez
University of Houston, USA

Ladjel Bellatreche
LIAS/ISAE-ENSMA, France

ABSTRACT

Data Science has brought up significant data processing outside the database realm. ER diagrams have a proven track record to represent data structure and relationships, beyond relational databases. ER diagrams provide abstraction, flexibility, and an intuitive visual representation of data structure richer than metadata. ER diagram notation has evolved, incorporating UML diagram symbols. In data science, data files need significant preprocessing before analysis them, resulting in a long sequence of complex data transformations. Such transformations are computed in diverse languages including Python, SQL, Java and R. We argue flow diagrams remain the main tool to visualize the main modules of a software or computer system or the main steps of an algorithm, showing rectangles with verbs (processing), connected by arrows (showing processing order or flow dependences). We take a bold step: we propose a hybrid diagram, mixing data pre-processing flow and data structure (i.e. FLOW+ER=FLOWER), based on modern UML notation. Since we want to introduce a minimal change to ER diagram notation, we extend relationships lines with an arrow, indicating processing flow and we label entities coming from pre-processing with numbers and transformation operators. Such diagram can be automatically created from programs processing diverse files. Our novel FLOWER diagram can help data scientists navigate data pre-processing, providing an integrated view of pre-processing programs with a data-centric angle.

ACM Reference Format:
Carlos Ordonez and Ladjel Bellatreche. 2023. FLOWER: Data Flow in ER Diagrams. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 3 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 [4], [2], Flow diagram [1], [8], and data transformation [7], [6] have been done by other researchers. In this work, we present a simple and heterodox idea. We propose a hybrid diagram, combining data flow and data structure, which we call FLOWER. The goal is to assist big data analysts in data pre-processing.

2 DEFINITIONS

2.1 ER model

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.

2.2 Database terms

Our proposed diagram extensions are inspired by database principles, extending them to data science and big data. We use the following acronyms: PK (primary key), foreign key (FK).

2.3 Data sets

In our diagram there are input, intermediate and output entities. In this work, we assume the input entities can be a bag (i.e. there may be repeated elements). That is, they do not have a primary key. However, when managing text files, it is common to parse and analyze files, line by line. Therefore, a line number is a primary key. In the case of images, the image file name is commonly a PK as well.

3 PROPOSED HYBRID DIAGRAM

We do not assume input files have keys in the traditional database sense, but we assume data pre-processing programs do produce entity names, keys, attributes and relationships. that can be captured from source code (i.e. file names, variable names, function names). That is, we propose reverse data design: create a diagram (kind of automated model) after data has been pre-processed.

3.1 Extending ER diagram notation

We allow relationships lines to have an arrow indicating data flow direction. This direction can also be interpreted as input and output, going from input entities to output entities. The arrow is shown only for data pre-processing entities (akin to denormalized or derived tables in a relational database). That is, in the particular case of "raw" data sets that can be represented as normalized tables there is no arrow between entities.

The arrow represents:

(1) A processing dependence between two entities.
(2) Data flow direction. This direction indicates one entity is used as input.

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. We emphasize that the entities remain linked by keys. In particular, a traditional ER diagram for a relational database still has PKs and FKs.

3.2 Entites beyond Databases

Our entity concept is broad: entities can represent a file, matrix, relational table, or dataframe. That is, we go beyond relational databases. Our proposed diagram can be considered a metadata representation for big data from a variety angle (i.e. diverse information, in different formats).

Entities are classified as:

- source (raw) entities, representing raw data, loaded into the Data Lake
- Transformation (data preparation, data pre-processing) entities being the output of some system (Hadoop), tool (statistics or machine learning) or some programming language used in data science (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 include (yet) 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 partitions and aggregates records based on some key. We must emphasize Data Science languages (Python, R) provide operators or functions highly similar to the SQL GROUP BY clause. Notice also GROUP BY produces data sets with a PK. In fact, this is the most common mechanism to eliminate and count duplicates in SQL.
(3) Derived expressions, which represent derived attributes coming from a combination of functions and value-level operators (e.g. equations, string manipulation, arithmetic expression, nested function calls).

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

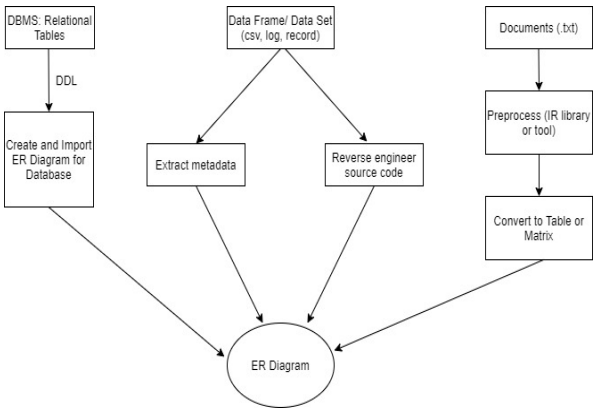

Figure 1: High-level flow diagram.

(1) Importing ER diagrams available from existing transactional or data warehouse 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 metadata embedded in the file itself. We assume CSV files are the default file format for spreadsheet data, logs and mathematical software. On the other hand, JSON is another common standard file format to exchange data among many platforms. JSON can be converted to a collection of CSV files and vice-versa.
(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 assume an IR library or tool will pre-process the file and convert it tables, matrices or data frames. Then we propose to extract entity and attribute names from the final table, matrix or data frame.
(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 [3], kind of “reverse engineering".

Following database principles, 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 data transformation module, we define the transformation type and create new transformed entities. Data scientists 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 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 FLOWER diagram in JSON files. This FLOWER 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 FLOWER diagram. We consider an example of a store for which we show the FLOWER 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.

4 RELATED WORK

This is not the first work that tackles visual representation of big data pre-processing. Data transformations represent valuable knowledge to reuse data sets and modern analysts have tons of source code which is impossible to understand. Such transformations have already been proposed for relational databases by extending the ER models with transformation entities [3, 5].

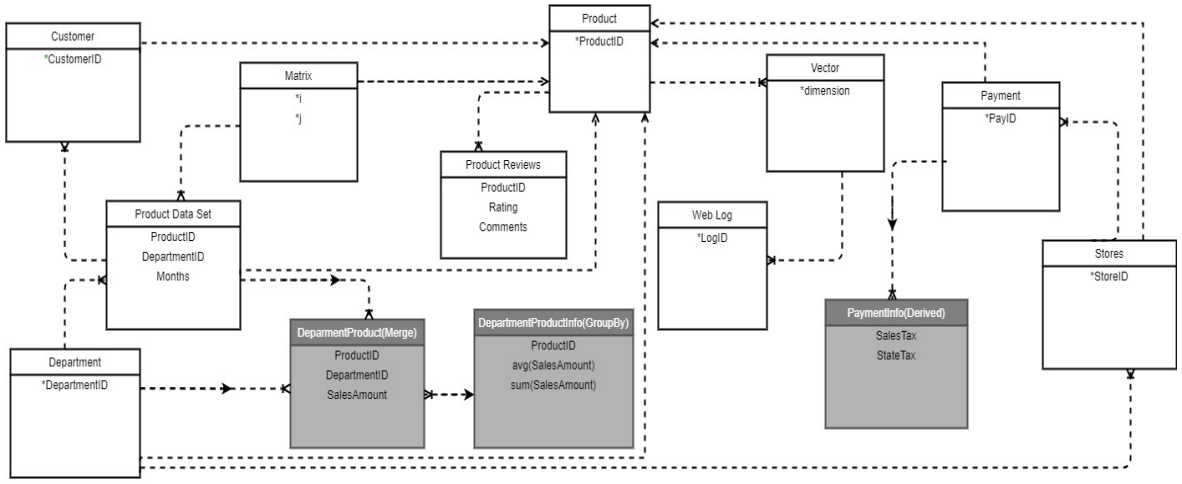

Figure 2: FLOWER diagram for a Store Data Lake (white entities are original raw entities, black entities represent data transformations).

In an alternative line of work, [6] solved data transformation understanding with minimal human interaction. A layout algorithm for automatic drawing of the data flow diagram was presented in [1]. This layout algorithms receives an abstract graph specifying connectivity relations between the elements as input, and produces a corresponding diagram as output. Wohed et. al in [8] analyzed the strengths and weaknesses of control flow specification in Activity Diagrams of UML. The authors of [4] presented a method for entity resolution that infers relationships between observed entities and uses those relationships to aid mapping identities to underlying entities.

5 CONCLUSIONS

We believe data science and big data need new tools to capture data structure and interrelationships, following and extending proven database design techniques: ER diagrams and relational databases. We presented a preliminary idea, suitable for a workshop. Evidently, there is a lot of work to do: automating entity identification from Python/SQL scripts, identifying entity relationships via input/output files in source code, producing a list of entities in UML notation and applying our idea on actual data science projects to get users feedback.

REFERENCES

[1] Batini, C., Nardelli, E., Tamassia, R.: A layout algorithm for data flow diagrams. IEEE Trans. Software Eng. **12**(4), 538–546 (1986)

[2] Guo, G.: 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] Lanasri, D., Ordonez, C., Bellatreche, L., Khouri, S.: 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

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

[5] Ordonez, C., Maabout, S., Matusevich, D.S., Cabrera, W.: Extending ER models to capture database transformations to build data sets for data mining. Data & Knowledge Engineering (2013)

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

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

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