Visualization-based Machine Learning Model for Relational Databases

Dihia Lanasri[1], Carlos Ordonez[2], Ladjel Bellatreche[3], Selma Khouri[1]
[1] ESI, Algiers, Algeria
[2]University of Houston, USA
[3]LIAS/ISAE-ENSMA, Poitiers, France

ABSTRACT: *Transforming several relational tables into a data set to be used as input to a Machine Learning (ML) model is a complex task since the data scientist has to derive many intermediate tables, queries and views in a disorganized manner. This process creates many SQL queries to facilitate the exploration task of the data scientist. Because the provenance of the intermediate results is not reflected, similar SQL queries tend to be written multiple times causing repeated manual work. In this paper, we propose a tool "ER4ML" assisting data scientists in modeling and visualizing the transformations applied to the relational database before obtaining a dataset feeding the ML models. ER4ML is a diagram flow based on a conceptual view of the database schema and transformations based on powerful extensions of ER diagram in UML notations. In addition, ER4ML tracks data provenance, improving query and data set reuse.*

Keywords: ER Diagram, Provenance, Data Transformation, Query, Machine Learning

Received: 12 December 2019, Revised 4 March 2020, Accepted 13 March 2020

DOI: 10.6025/jet/2020/11/2/49-53

Copyright: with Authors

1. Introduction

In order to feed their defined Machine Learning (ML) models, data scientists collect, clean, aggregate and transform important amounts of relational data to elaborate a tabular dataset with many features. Pre-processing relational datasets is a complex process consuming more than 80% of the project time. This Pre-processing needs to execute different SQL queries in a disorganized manner outside the DBMS, without thinking to model the entity-relationship(ER) modifications. These transfromations result in several intermediate temporary tables or views, that are not modeled in the ER diagram. These transformations are valuable knowledge for future analytics reusing similar sets of entities. However, reusing them presents a challenge for data scientists because of the lack of a modeling and GUI tools helping them to understand what has been performed and the intermediate results of SQL queries that construct the ML model. We defend the idea of creating an extended ER model to understand such a set of disconnected tables and views at a high level.

To overcome this issue, we propose a tool "*ER4ML*" aiming to assist data scientists to visualize and understand the different transformations using complex SQL queries (including joins, selections, projections, aggregations and the powerful CASE statement), applied to the relational dataset given as an input to an ML model. ER4ML is based on a defined model "Extended ER" detailed in [5] proposing minimal but powerful extensions to an ER diagram as a conceptual flow representation. It helps to understand a chain of transformations basing on SQL queries going from the physical level(tables) to the logical level (entities), in UML notations. The importance of SQL queries to transform relational data to build a data set for ML is identified in [4]. ER4ML provides a graphical web interface that shows the transformations applied to the ER model and the provenance of each attribute (primary or foreign keys and non-key) with a coloring code to distinguish between source and transformation entities. The transformation entities can be of two main types: denormalization or aggregation, while the final dataset is the result of outer joins (denormalization) tending to create non-3 NF tables. Our system shares some similarities with data reverse engineering and ETL tools, but they have many differences[5]. We establish a connection with existing ER diagrams, instead of building a separate diagram.

This paper is organized as follows: Section 2 discusses the related work. Section 3 presents the architecture of our tool and details its main modules. Section 4 presents our demonstration scenario and Section 5 concludes our paper.

2. Related Work

Based on reverse engineering processes for extracting the database conceptual model from its physical model, some studies defined conceptual view of relational datasets[1] or extending ER modeling by defining some schema transformations[6]; or data warehouse conceptual model by classification on mining models[8]. Our paper focuses on a unique extended ER diagram that unifies the physical and conceptual levels. Other solutions propose to model the ETL/ELT processes at a conceptual level[3] basing on the BPMN model[2]. Our solution represents data processing with transformation entity boxes and SQL queries.

3. Research Application Contribution

3.1. Extended ER Model for Data Transformation

The proposed tool is based on the Extended ER model that defines a database as $D(T, I)$, where $T = \{T_1, \dots, T_n\}$, is a set of n tables and I is a set of integrity constraints (entity for primary key (PK), referential for foreign key (FK)), and represents the chain of transformations as a conceptual flow diagram. We use the term "table" to make a connection to the physical level and SQL querying. SQL queries can be classified into either aggregation or denormalization transformation. We assume well formed queries are used to create datasets, which include a PK to identify rows and at least one non-key attribute.

3.2. Layered System Architecture

Figure 3 illustrates the architecture of our system which is designed following the 3-tiers architecture based on the MVC paradigm covering three layers:

Presentation Layer: is the GUI presented to the user for visualizing the database schema, the different transformations and the used SQL queries. The system distinguishes between source and transformation entities. The type of each entity is indicated in its name (eg. Source: EntityName, Denormalization: *Ti and Aggregation: Tj*). The final obtained dataset is named "Dataset: *Tx*" where (*i, j, x*) are sequence numbers. Our diagram flow is automatically generated based on physical model notation complementary to UML notations. Our Extended ER notation does not change UML notation for ER, instead we show transformation entities in color green/Red/blue and ER entities in white. The arrows represent data flow (NOT relationships) but we still link entities (source & transformation) with FKs/PKs. This layer is developed using HTML, BOOTSTRAP, CSS, JavaScript (JS), and GOJS library used to build interactive ER diagrams where the user can hide or show selectively the transformation entities.

Functional Logic layer: is the core of our solution. It contains three main modules of the system: Model extraction, ER initialization, and Transformation modules responsible for generating an extended ER model for each dataset to be given to an ML model. **(i) Model extraction module:** generates an ER model from a relational data source using a specific DBMS connector. Our solution deals with any DBMS, but in our demonstration scenario we consider sqlite3 module for node JS connceting to SQLite DBMS thanks to its simplicity and overhead. This module extracts the structure of the database (tables, their columns, constraints and relationships) and generates two JSON files that define the model and that can be read by GOJS library. The first and the second files contain respectively the relationships and the entities. **(ii) ER initialization module:** takes the two JSON files as input and extracts the detailed structures of nodes and links to initialize the ER model, to be displayed by the presentation module. **(iii) Transformation module:** we assume the user is responsible for specifying the components and results of each transformation and has the prior knowledge to distinguish between fact table & analytic dataset and check the validity of ER models. This module enriches the extended ER model for each new SQL query by: (a) defining the transformation type (Aggregation or Denormalization), (b) defining the SQL query that generates the intermediate table, the query is displayed as a comment next to the transformation entity so to zoom in the transformation applied. (c) creating the transformation entities (named ”TYPE Ti” *i* is an auto-increment sequence number) that are displayed using different colors. The module also allows: (d) associating the *PK*, *FK* & other attributes to the transformation entity. (e) According to the tables joined in the From clause of the query, the module identifies the relationships to existing entities to show the provenance of the transformation entities & their attributes, using an ARROW link between transformation entities, this allows the user to folow the chain of transfromations. (f) creating cardinalities of relationships: one2many and many2one.

Data Layer: after each valid transformation step, the new generated extended ER model is stored as JSON files. The development of this tool is performed on WebStorm IDE. GitHub is used for the versioning of the project. The source code of our solution is available on: https://github.com/dihiaselma/ER4ML

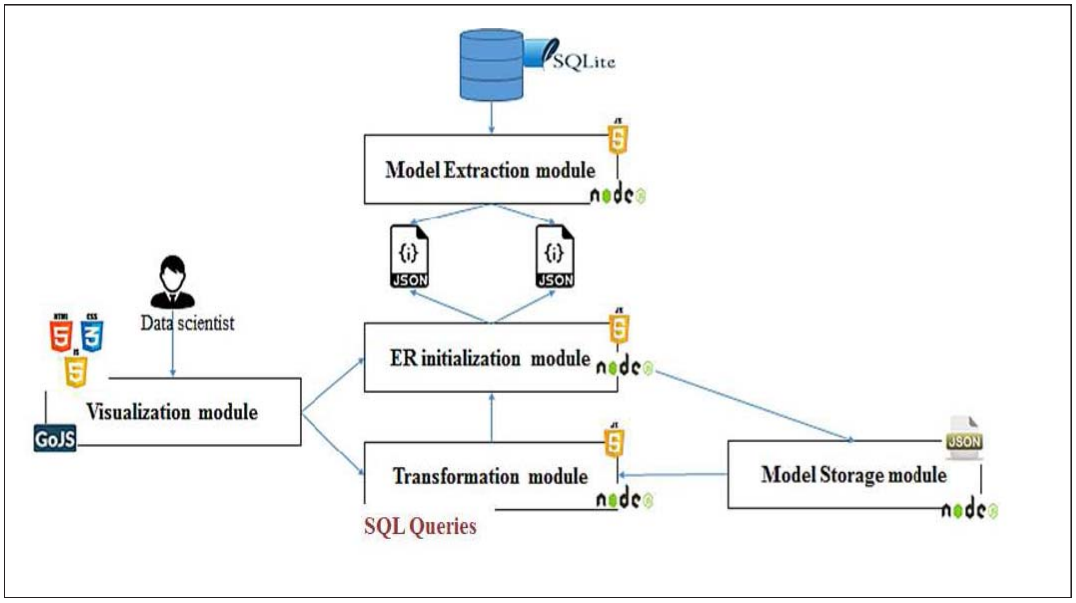
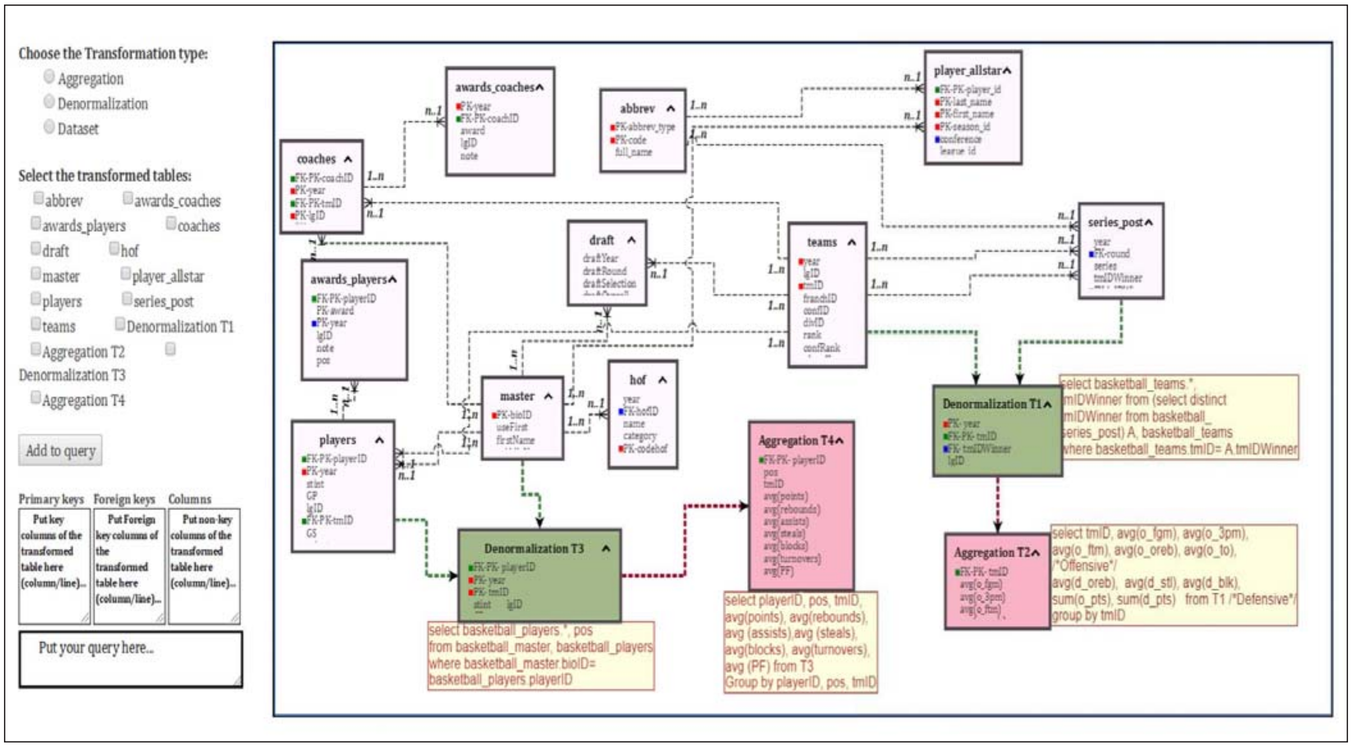


Figure 1. The Overall Architecture of ER4ML

4. Demonstration Overview

In our system demonstration we will show our extended, elegant and intuitive ER model obtained in a few seconds from typical SQL queries. It helps understanding complex data transformations. Our demonstration is based on the basketball dataset containing statistics about players, coaches, and teams in men’s professional basketball leagues from 1937 to 2012, which are extracted from the Open Source Sports website. We constructed an SQLite database from these files and proceeded to verification of integrity constraints and PK-FK links. We use 14 transformations4 deriving features using some complex Joins, aggregations and the powerful CASE statement, resulting from the previous work [7] that include queries using CASE statement to obtain the last dataset that will be used for the ML model aiming to predict winning and losing teams. Using our tool, a user will see the ER model corresponding to the defined database, displayed as a dynamic graph. Let us take as an example the first query that represents denormalization. The user chooses ’Denormalization’ transformation type, specifies the query and the set of included tables. Then, she specifies the list of attributes, the cardinalities between entities can also be specified. The new diagram that extends the existing one is generated displaying the first transformation entity as shown in figure 2. The new transformation table is in light green, named ”Denormalization T1”, next to the SQL query defined as a comment bull to help the user understanding the transformation executed. Furthermore, the PK and FK columns are labeled and the cardinalities are displayed above the arrow link colored in dark green. In the case of Aggregation, the transformation entities are in light red while the relationship(arrow) is in dark red. Our tool helps data scientists refine data source in multiple iterations by adding, removing and changing features basing on JOINS and aggregations via the described SQL queries until obtaining the final dataset containing offensive & defensive features.

A detailed demonstration video is available at https://youtu.be/nJkZ8aBa6co which gives a nice overview of different functionalities of our tool. It emphasizes on following points: (1)ER representation of temporary tables, (2)Flow of a chain of data transformations, (3)Conceptual separation between aggregation and denormalization, (4)Complex SQL queries, (5)Reusing existing queries and tables and (6) Connected view of data via primary and foreign keys.



```sql
select basketball_teams.*,
tmIDWinner from (select distinct
tmIDWinner from basketball_
series_post) A, basketball_teams
where basketball_teams.tmID= A.tmIDWinner
```

```sql
select tmID, avg(o_fgm), avg(o_3pm),
avg(o_ftm), avg(o_oreb), avg(o_to),
/*Offensive*/
avg(d_oreb),  avg(d_stl), avg(d_blk),
sum(o_pts), sum(d_pts)   from T1 /*Defensive*/
group by tmID
```

```sql
select basketball_players.*, pos
from basketball_master, basketball_players
where basketball_master.bioID=
basketball_players.playerID
```

```sql
select playerID, pos, tmID,
avg(points), avg(rebounds),
avg (assists),avg (steals),
avg(blocks), avg(turnovers),
avg (PF) from T3
Group by playerID, pos, tmID
```

Figure 2. Denormalization & Aggregation transformations

5. Conclusion

In order to assist data scientists managing and understanding the different transformations applied to a relational database and create a tabular dataset serving as input to their ML models, we developed a new tool "ER4ML" that graphically generates an extended ER model based on UML notations. It models all the intermediate temporary tables or views resulting from SQL queries, as entities enriching the existing database schema. This tool can be seen as an interesting response to the major question discussed during the Dagstuhl seminar 18471 titled Next Generation Domain Specific Conceptual Modeling: Principles and Methods, in last November regarding the integration of conceptual modeling in ML. Despite this, our tool presents some limitations: i) The user cannot view the ML model as entities. ii) The user cannot view data transformations done with external languages like Java, R or Python. iii) Our tool is good for data represented with tables, not for text or images. As a perspective, we are extending our tool to deal with datasets issued from different database formats.

References

[1] Boyd, M., McBrien, P. (2005). Comparing and transforming between data models via an intermediate hypergraph data model. In: JoDS IV, p 69–109. Springer.

[2] El Akkaoui, Z., Maz´on, J. N., Vaisman, A., Zim´anyi, E. (2012). Bpmn-based conceptual modeling of etl processes. In: DaWaK. p 1–14. Springer.

[3] Oliveira, B., Belo, O. (2013). Using reo on etl conceptual modelling: a first approach. In: DOLAP. p 55–60. ACM.

[4] Ordonez, C. (2011). Data set preprocessing and transformation in a database system. *Intelligent Data Analysis* (IDA) 15(4).

[5] Ordonez, C., Bellatreche, L. (2019). Enhancing ER diagrams to view data transformations computed with queries. In:

DOLAP.

[6] Poulovassilis, A., McBrien, P. (1998). A general formal framework for schema transformation. DKE 28(1), 47–71.

[7] Shi, Z., Moorthy, S., Zimmermann, A. (2013). Predicting ncaab match outcomes using ml techniques-some results and lessons learned. In: ECML/PKDD.

[8] Zubcoff, J., Trujillo, J. (2006). Conceptual modeling for classification mining in data warehouses. In: DaWaK. p 566–575. Springer.