ORM Ontologies with Executable Derivation Rules to Support Semantic Search in Large-Scale Data Applications

Márton Búr
marton.bur@relational.ai
RelationalAI Inc.
Zürich, Switzerland

Kurt Stirewalt
kurt.stirewalt@relational.ai
RelationalAI Inc.
Atlanta, Georgia, USA

ABSTRACT

A semantic layer maps complex enterprise data into an ontology with abstract business concepts that are well-known to business users. Chief data officers invest significant effort to create and update these ontologies, while data scientists do feature engineering by combining already existing concepts and features of the domain. However, it is a significant challenge to catalogue and maintain the numerous features pertaining to an ontology, which leads to duplicated features and unnecessary complexity. In this work, we propose to combine ontologies captured using the Object-Role Modeling notation with derivation rules defined in a datalog-like language called Rel, which allows the creation of a semantic layer with feature search capability. Our prototype framework uses the RAI Knowledge Graph Management System, which provides automated and incremental derivation rule evaluation.

ACM Reference Format:
Márton Búr and Kurt Stirewalt. 2022. ORM Ontologies with Executable Derivation Rules to Support Semantic Search in Large-Scale Data Applications. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3550356.3559576

1 INTRODUCTION

Data scientists perform *feature engineering* to distill a set of features that can be analyzed to predict some outcome of interest. Feature engineering at enterprise scale is expensive because:

(1) Enterprise data are rarely designed to be accessible by those with domain knowledge and a business problem to solve but little or no design knowledge of the applications that create and use the data; and
(2) The number of features can grow into the tens of thousands, making the cataloging and maintenance of the feature set a significant software engineering challenge.

Recently, chief data officers have started building *semantic layers* to address the first problem. Semantic layers weave low-level enterprise data into a common ontology that is consumable by those with knowledge of the domain.

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9467-3/22/10.
https://doi.org/10.1145/3550356.3559576

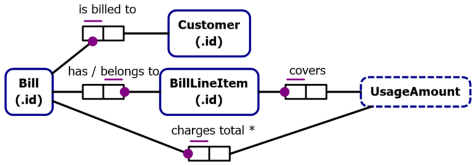

Figure 1: Example ontology and a feature (ORM)

We believe the second problem can be made tractable by means of *semantic search* if data scientists engineer features on top of a semantic layer. A feature is a relationship computed over concepts from the ontology using a recipe that ultimately derives from relationships that are declared in the ontology. A consumer should be able to search for any feature by naming the concept(s) that it ranges over and one or more of the relationships (called *ingredients*) of its recipe. In fact, even if a consumer does not know all of a recipe's ingredients, they can often name at least some, which often suffices to narrow the search to a small enough set of matching features. If the language used to declare ontologies and features provides a means for declaring derived relationships – and if those recipes can be analyzed formally – then data scientists can search for features in large feature sets using concepts and relationships from the ontology. However, up to our best knowledge, there is no ontology modeling language providing these capabilities.

This work uses ontologies declared in the Object Role Modeling notation (ORM) [4] supplemented with feature definitions written in a datalog-like language called Rel to help data scientists search for and reason about feature sets in terms of the business domain. Our solution is novel because we provide a framework that (1) on the meta-level allows loading the ontology into a knowledge graph and creating a semantic layer that provides search functionality across all features, and (2) supplements ORM with executable derivation rules built from the concepts and features of the ontology.

2 REPRESENTING ONTOLOGIES & FEATURES

We use the Object Role Modeling (ORM) notation designed for conceptual modeling to declare an ontology and any features that are derived from the semantic layer that populates the ontology. The first row of Figure 1 depicts the ontology of a fictional enterprise that sends out bills to its customers. ORM depicts concepts as rounded rectangles and relationships as one or more adjacent rectangles (called *roles*) that are linked to the concept that plays the role in the relationship. This ontology declares four concepts – `Customer`, `Bill`, `BillLineItem`, and `UsageAmount` – and three relationships – which verbalize as "Bill is billed to Customer", "BillLineItem covers UsageAmount", and "BillLineItem belongs to Bill".

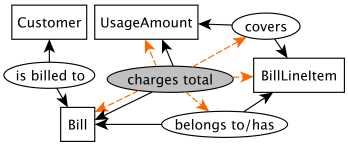

Figure 2: Knowledge graph instance storing metadata

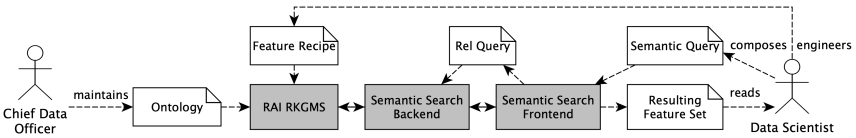

Figure 3: Semantic search workflow

Listing 1: Recipe for `charges total` feature

```
def charges_total[b in Bill] =
    sum[ l in BillLineItem, a:
         belongs_to(l, b) and covers(l, a) ]
```

Any feature that is built atop a semantic layer can be depicted as a *derived relationship* whose roles are played by ontological concepts and whose recipe refers to some combination of relationships from the ontology and/or other features. Figure 1 shows an example feature called "Bill charges total UsageAmount" that a data scientist wishes to derive. The asterisk beside the "charges total" text in the figure indicates that the relationship is derived.

We use the Rel language [5] to precisely define the recipe for each feature. Listing 1 shows how the "charges total" feature from Figure 1 is specified in Rel. The keyword **def** indicates that what follows is the definition of a relationship named `charges_total` that is a feature of concepts of type `Bill` (variable `b` refers to an arbitrary instance of `Bill`). The value to associate with a given `Bill b` is the sum of all `UsageAmounts a` that are reachable from `b` using some `BillLineItem l`. Notice how the recipe refers to relationships `belongs_to` and `covers` present in the ontology.

3 SEMANTIC SEARCH WITH RAI RKGMS

To support semantic search of features, we load both the ontology and metadata about the recipe of each feature into a *knowledge graph* that enables easy navigation among concept and relationship nodes and the feature nodes that are connected via edges according to the recipe metadata. Figure 2 shows the knowledge graph capturing metadata from our running example from Figure 1 and Listing 1. Rectangles denote concepts from the ontology, ellipses show relationships and features. Solid black arrows describe what concepts are connected by a relationship or feature. The engineered feature `charges total` has grey background and dashed orange arrows refer to the ingredients present in its derivation recipe.

The proposed workflow to use our RAI RKGMS-based semantic search service is depicted in Figure 3. There are two personas using the system: chief data officer and data scientist. The responsibility of a chief data officer (left side of Figure 3) is to maintain the ontology and make sure that changes are properly updated in the RKGMS providing the semantic search capabilities. In essence, this activity is about updating the metadata for the asserted facts in the knowledge graph. On the other hand, data scientists (shown in the right side of Figure 3) do feature engineering by defining derivation rules called *feature recipes* and loading these to the RKGMS as metadata and thus enriching the ontology defined by the chief data officer. Once a feature is engineered, the system automatically uses the declarative recipe of the feature to execute it as a rule and populate it with data. Additionally, a data scientist can issue semantic queries using a UI designed for interactive composition of queries over metadata. Results of a query contain a set of features that have recipes using the selected concepts, relationships, and/or features as ingredients.

4 RELATED WORK

The idea to extend and enhance graphical conceptual models with textual languages is not new. The Object Constraint Language (OCL) [1] is an extension to UML to allow precise definition of business rules. However, the grammar can be challenging to learn, which can be discouraging for business users [6].

A role calculus for ORM is proposed by Curland et al. [2] to enable formulation of queries and integrity constraints in a language intelligible to domain experts. The role-based nature of this calculus exploits the attribute-free nature of ORM and relies on its fact-oriented approach to provide straightforward verbalization, but this work does not focus on evaluation of the rules.

Automated computation and incremental maintenance of derived features using declarative graph query specifications for incremental maintenance of view models is presented in [3]. The main difference here compared to our work is the management of explicit vs. implicit traceability information in the results and the underlying data representation (object-oriented vs relational).

5 FUTURE WORK

In the coming weeks, we are deploying our prototype framework and assess its performance and usability. An important question is if we hit any performance boundaries with deeply nested derived relations and graph-like recursive features. Additionally, we are developing a visual graph explorer view to aid data scientists in feature exploration and show verbalizations for derived relationships.

Acknowledgement. The authors would like to thank Sara Henchiri and Vadim Mihalovski their contributions to this work.

REFERENCES

[1] Jordi Cabot and Martin Gogolla. 2012. Object constraint language (OCL): a definitive guide. In *International school on formal methods for the design of computer, communication and software systems*. Springer, 58–90.
[2] Matthew Curland, Terry Halpin, and Kurt Stirewalt. 2009. A role calculus for ORM. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 692–703.
[3] Csaba Debreceni, Ákos Horváth, Ábel Hegedüs, Zoltán Ujhelyi, István Ráth, and Dániel Varró. 2014. Query-driven incremental synchronization of view models. In *2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modeling*.
[4] T. Halpin and T. Morgan. 2008. *Information Modeling and Relational Databases* (second ed.). Morgan Kaufmann.
[5] Kurt Stirewalt. 2022. *Experience report: building enterprise applications using LogiQL and Rel.* https://www.hytradboi.com/2022/experience-report-building-enterprise-applications-using-logiql-and-rel
[6] Mandana Vaziri and Daniel Jackson. 2000. Some shortcomings of OCL, the object constraint language of UML. In *TOOLS (34)*. Citeseer, 555–562.