Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies

Farid Cerbah
Dassault Aviation
Department of Scientific Studies
78, quai Marcel Dassault 92552 Saint-Cloud – France
farid.cerbah@dassault-aviation.fr

Abstract

Relational databases are valuable sources for ontology learning. Previous work showed how precise ontologies can be learned from such structured input. However, a major persisting limitation of the existing approaches is the derivation of ontologies with flat structure that simply mirror the schema of the source databases. In this paper, we present the RTAXON learning method that shows how the content of the databases can be exploited to identify categorization patterns from which class hierarchies can be generated. This fully formalized method combines a classical schema analysis with hierarchy mining in the data. RTAXON is one of the methods implemented in the RDBToOnto tool.

1 Introduction

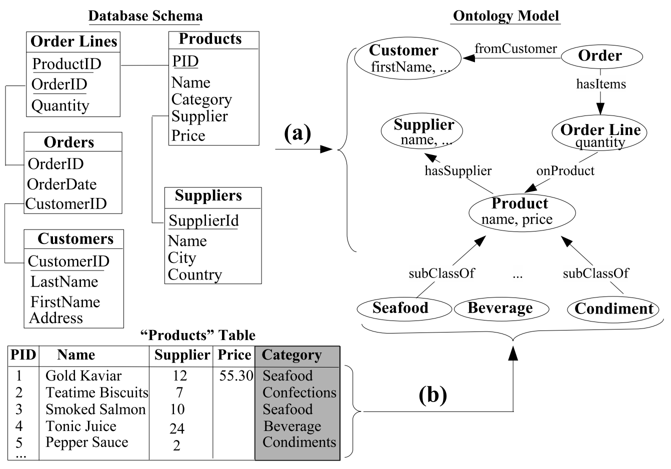
In companies that need to produce and manage technical knowledge on complex engineering assets, as in aerospace and automotive industries, a large proportion of technical corporate repositories are built upon relational databases. These repositories count undoubtedly among the most valuable sources for automatically building highly accurate and effective domain ontologies. However, a major persisting limitation of the existing methods is the derivation of ontologies with flat structure that simply mirror the schema of the source databases. Such results do not fully meet the expectations of users that are primarily attracted by the rich expressive power of semantic web formalisms and that could hardly be satisfied with target knowledge repositories that look like their source relational databases. Ontologies with flat structure is the typical result of learning techniques that exclusively exploit metadata from the database schema without (or just marginally) considering the data. A careful analysis of existing databases shows that additional definition patterns can be learned from the data to significantly enrich the ontology structure. More particularly, class hierarchies can be induced from the data to refine classes derived the relational schema.
In this paper, we define RTAXON, an approach to ontology learning from relational databases that combines two complementary information sources: the schema definition and the stored data. We show how the content of the databases can be exploited to find deeper class hierarchies. RTAXON is implemented in RDBToOnto[1], a comprehensive tool that supports the transitioning process from access to the data to the generation of fine-tuned populated ontologies [5].

2 A Motivating Example

We start by depicting the typical transitioning process on a representative example (figure 1).

The derivations applied to get the target ontology can be divided in two inter-related parts. The first part, named **(a)** in the figure, includes derivations that are motivated by the identification of patterns from the database schema. Each relation (or table) definition from the database schema is the source of a class in the ontology. Such simple mappings from relations to classes are often relevant though several exceptions need to be handled (for instance, some relations are more likely to be translated as class-to-class associations). To complete the class definitions, datatype properties are derived from some of the relation attributes. The foreign key relationships are the most reliable source for linking classes and, in this example, each of the four relationships is translated into an object property. The derivations applied to obtain this upper part of the ontology are well covered by current methods and, if applied on this database sample, most of the methods would provide the result of the **(a)** derivations as final output. However, by looking closer

[1] http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html



"Products" Table

PID	Name	Supplier	Price	Category
1	Gold Kaviar	12	55.30	Seafood
2	Teatime Biscuits	7		Confections
3	Smoked Salmon	10		Seafood
4	Tonic Juice	24		Beverage
5	Pepper Sauce	2		Condiments
...				

Figure 1. An example of ontology building by exploiting both the schema and the data

at the data, we can notice that additional structuring patterns can be exploited to refine the ontology structure. More particularly, the **(b)** part of the derivations shows how the Product class can be refined with subclasses derived from the values of Category column in the Products source table. In the same vein, the Supplier class can be extended with a two-level hierarchy by interpreting the values in both Country and City columns of the corresponding table[2].

These are typical examples of subsumption relations that can be found by mining the database content.

3 Related Work

Ontology learning from relational databases is a relatively recent issue. However, it can benefit from early work in the domain of database reverse engineering where several methods have been proposed to extract object-oriented models from relational models (e.g. [3, 8]). The core of the transformation rules for database reverse engineering are still relevant in the context of ontology learning. The most reliable rules have been reused as a starting point and extended in several approaches that have ontologies as target models [9, 1, 7].

Most approaches are based on an analysis of the relational schemas. However, to some extent, the use of the database content has been investigated yet, both in reverse engineering and ontology learning, to find correlations between key values [10, 1]. More particularly, key inclusion may reveal inheritance. In practice, the rules based on the identification of key-based constructs are not the most productive as these modelling schemes are only found in carefully designed databases. In [6], the identification of subsumption relations is based on a precise interpretation of null value semantics. Partitioning a table on the basis of null values may reveal an underlying concept hierarchy.

As a related issue, mapping languages [4, 2] are declarative means that provide convenient ways to map relational models to pre-defined ontologies.

[2]Resulting in subclasses Sweden Supplier $\longrightarrow$ Stockholm Supplier, Göteborg Supplier, etc.

4 Combining Schema and Data Analysis

The primary motivation in the design of the RTAXON method was to combine the most robust rules for exploiting relational schemas with data mining focused on the specific problem of concept hierarchy identification. One of the key issues addressed in this work is the identification of relation attributes that may serve as good categorization sources and we show how these specific learning mechanisms can be coherently integrated into a comprehensive learning approach to ontology construction.

4.1 Preliminary Definitions

We fix some basic notations and definitions that will be used to describe our approach.

A *relational database schema* D is defined as a finite set of relation schemas $D = \{R_1, \dots, R_n\}$ where each *relation schema* R_i is characterized by its finite set of attributes $\{A_{i1}, \dots, A_{im}\}$. A function $pkey$ associates to each relation its primary key which is a set of attributes $K \subseteq R$.

A relation r on a relation schema R (i.e. an *instance* of R) is a set of tuples which are sequences of $|R|$ values. Similarly, a database d on D is defined as a set of relations $d = \{r_1, \dots, r_n\}$. By convention, if a relation schema is represented by a capital letter, the corresponding lower-case letter denotes an instance of the relation schema.

Inclusion dependencies are used to account for correlations between relations. An inclusion dependency is an expression $R[X] \subseteq S[Y]$ where X and Y are respectively attribute sequences of R and S relation schemas, with the restriction $|X| = |Y|$. The dependency holds between two instances r and s of the relation schemas if for each tuple u in r there is a tuple v in s such that $u[X] = v[Y]$. Informally, an inclusion dependency is a convenient way to state that data items are just copied from another relation.

Foreign key relationships can be defined as inclusion dependencies satisfying the additional property: $Y = pkey(S)$. The notation $R[X] \subseteq S[pkey(S)]$ is used for these specific dependencies.

Formal descriptions of ontology fragments are expressed in OWL abstract syntax.

4.2 The Overall Process

The main steps of the process are: database normalization, class and property learning, and ontology population.

- Database Normalization

In early approaches, this stage is not integrated in the learning process. It is quite common to consider as input relational databases that are in some normal form, often 2NF or 3NF. It is assumed that the transformation process can be easily extended to cope with ill-designed databases by incorporating at the early stages of the process a normalization step based on existing algorithms. Though theoretically acceptable, this assumption has some drawbacks in practice as many interesting databases suffer from redundancy problems. More particularly, data duplication between relations is a frequent problem that may have a bad impact on the resulting ontologies. Such data duplications can be formalized as inclusion dependencies. To eliminate the duplications, the database need to be transformed by turning all inclusion dependencies into foreign key relationships. More formally, each attested dependency $R[X] \subseteq S[Y]$ with $Y \neq pkey(S)$ is replaced by the foreign key relationship $R[A] \subseteq S[pkey(S)]$, where A is a newly introduced foreign key attribute, and all non-key attributes in X together with related data in r are deleted from the relation. This preliminary step is semi-automated as the inclusion dependencies to be processed are defined manually and the database transformation is performed automatically.

- Class and Property Identification

This is the core step of the ontology learning process where relations of the database are explored to derive parts of the target ontology model. The database schema is the first information source exploited through the application of prioritized rules that define typical mappings between schema patterns and ontology elements, namely classes, datatype and object properties. We give in table 1 three of the most reliable rules which are also employed in several existing approaches. The first trivial rule states that every relation can potentially be translated as a class (though relations can be consumed by more specific rules with higher priority, such as the third rule). The second rule is also a simple mapping, from a foreign key relationship to a functional object property. The third rule is intended to match a relation with a composite primary key and two key-based attributes. Such bridging relations are introduced in databases to link two other relations through key associations. They are turned into many-to-many object properties.

Content of the relations is a second information source allowing to refine with subclasses some of the classes obtained by applying schema-based mapping rules. This important part is described in section 4.3.

- Ontology Population

Final step aims at generating instances of classes and properties from the database content. For a given class, an instance is derived from each tuple of the source relation. Moreover, if refinement into subclasses has been successfully applied on the class at hand, the instances need to be further dispatched into the subclasses.

Relation to Class

Source	Preconditions	Target
$R \in D$	$\neg \exists C \mid R = sourceOf(C)$	class($\boldsymbol{C}_R$)

Foreign key Relationship to Functional Object Property

Source	Preconditions	Target
		ObjectProperty($\boldsymbol{P}_A$
$R_0[A] \subseteq R_1[pkey(R_1)]$	$R_0 = sourceOf(C_0)$	domain(C_0)
	$R_1 = sourceOf(C_1)$	range(C_1)
		Functional)

Composite Key Relation to Object Property

Source	Preconditions	Target
$R_0 \in D$		ObjectProperty($\boldsymbol{P}_R$
$\lvert R_0 \rvert = 2$	$R_1 = sourceOf(C_1)$	domain(C_1)
$pkey(R_0) = \{K_1, K_2\}$	$R_2 = sourceOf(C_2)$	range(C_2))
$R_0[K_1] \subseteq R_1[pkey(R_1)]$		
$R_0[K_2] \subseteq R_2[pkey(R_2)]$		

Table 1. Three reliable rules that match patterns in the database schema. In the **Target** part, the variable in bold holds the Uri of the generated ontology fragment. *sourceOf* assertions provide traceability to control the process

4.3 Extracting Hierarchies from the Data

Our motivating example in section 2 provided illustration of some modelling patterns attested in many databases where specific attributes are used to assign categories to tuples. These frequently-used patterns are highly useful for hierarchy mining as values of these *categorizing attributes* can be exploited to derive subclasses.

Our method for hierarchy mining is focused on exploiting the patterns based on such categorizing attributes. We describe below the pattern identification procedure. Then, we discuss the generation of the subclasses from the identified patterns.

4.3.1 Identification of the categorizing attributes

Two sources are involved in the identification of categorizing attributes: names of attributes and data diversity in attribute extensions (i.e. in column data). These two sources are indicators that allow to find attribute candidates and select the most plausible one.

- Identification of lexical clues in attribute names

When used for categorization, the attributes may bear names that reveal their specific role in the relation (i.e. classifying the tuples). In example of figure 1, the categorizing attribute in the Products relation is clearly identified by its name (Category). The lexical clue that indicates the role of the attribute can just be a part of the name, as in the attribute names CategoryId or Object Type. A list of clues can be set up and used to perform a first filtering of potential candidates.

- Filtering through entropy-based estimation of data diversity

With an extensive list of lexical clues, the first filtering step appears to be effective. However, experiments on complex databases showed that this step often identifies several candidates. The selection among the remaining candidates is based on an estimation of the data diversity in the attribute extensions. A good candidate might exhibit some typical degree of redundancy that can be formally characterized using the concept of entropy from information theory.

Entropy is a measure of the uncertainty of a data source. In our context, attributes with highly repetitive content will be characterized by a low entropy. Conversely, among attributes of a given relation, the primary key will have the highest entropy since all values in its extension are distinct.

Informally, the rationale behind this selection step is to favor the candidate that would provide *the most balanced distribution of instances within the subclasses*.

We give in what follows a formal definition of this step.

If A is an attribute of a relation schema R instantiated with relation r, the diversity in A is estimated by:

$$H(A) = -\sum_{v \in \pi_A(r)} P_A(v) \,.\, \log P_A(v) \qquad (1)$$

$$P_A(v) = \frac{|\sigma_{A=v}(r)|}{|r|} \qquad (2)$$

- $\pi_A(r)$ is the *projection* of r on A defined as $\pi_A(r) = \{t[A] \mid t \in r\}$. This set is the *active domain* of A. In other words, $\pi_A(r)$ is the set of values attested in the extension of A. Each value v of $\pi_A(r)$ is a potential category (to be mapped to a subclass in the ontology).

- $\sigma_{A=v}(r)$ is a selection on r defined as $\sigma_{A=v}(r) = \{t \in r \mid t[A] = v\}$. This selection extracts from the relation r the subset of tuples with A attribute equal to v. In this specific context, the selection extracts from the relation all entries with (potential) category v.

- $P_A(v)$ is the probability of having a tuple with A attribute equal to v. This parameter accounts for the weight of v in A. It can be estimated by the relative frequency of v (i.e. maximum likehood estimation).

Let now $C \in R$ denote the subset of attributes preselected using lexical clues. A first pruning operation is applied to rule out candidates with entropy at marginal values:

$$C' = \{\ A \in C \mid H(A) \in [\ \alpha,\ H_{max}(R)\ .\ (1-\beta)\ ]\ \} \quad (3)$$

- $H_{max}(R)$ is the highest entropy found among attributes of the relations ($H_{max}(R) = \max_{A \in R} H(A)$)
- α and β are parameters such that $\alpha, \beta \in [0,1]$.

As said earlier, $H_{max}(R)$ is often the entropy of the primary key attribute.

If several candidates still remain[3], we ultimately select the attribute that would provide the most balanced organization of the instances. This amounts to look for the attribute whose entropy is the closest to the maximum entropy for the number of potential categories involved:

$$\tilde{H}_{max}(A) = -\log \frac{1}{|\pi_A(r)|} \qquad (4)$$

This reference value, which is derived from the entropy expression (1), is representative of a *perfectly* balanced structure of $|\pi_A(r)|$ categories with the same number of tuples in each category. Note that this value is independent of the total number of tuples ($|r|$).

The final decision aims at selecting the attribute A^* whose entropy is the closest to this reference value:

$$A^* = arg \min_{A \in C'} \delta(A) \qquad (5)$$

Where

$$\delta(A) = \frac{|H(A) - \tilde{H}_{max}(A)|}{\tilde{H}_{max}(A)} \qquad (6)$$

4.3.2 Generation and population of the subclasses

As shown in first rule of table 2, the generation of subclasses from an identified categorizing attribute can be straightforward. A subclass is derived from each value type of the attribute extension (i.e. for each element of the attribute active domain). However, proper handling of the categorization source may require more complex mappings. The second rule in table 2 matches a more specific pattern where values to be used for subclass generation are extracted from another relation. The structuring scheme handled by this rule is encountered in many databases. We give in figure 2 an example where this scheme is applied. In this example, the categorizing attribute CatId in Albums relation is linked through a foreign key relationship to Categories relation in which all allowed categories are compiled. More suitable class names can be assigned by using the values from the second attribute named Description in the Categories relation instead of the numerical key values. In addition, a more exhaustive hierarchy can be derived by considering also the categories that have no associated tuples in the Albums relation, such as Tango category.

Classes of the resulting hierarchy are populated by exploiting the tuples from the same source relation. An instance is generated from each tuple. The extra task of dispatching the instances into subclasses is based on a partitioning of

[3]Note that all candidates can be eliminated. In this case, the first candidate is arbitrarily chosen.

Categorizing Attribute Values to Subclasses

Source	Preconditions	Target
$r \in d$	$R = sourceOf(C)$	$\forall v \in \pi_A(r)$
$A = catAtt(r)$		class($\boldsymbol{C_v}$ partial C)

Categorizing Attribute (Indirect) Values to Subclasses

Source	Preconditions	Target
$r \in d$		
$A = catAtt(r)$		
$R[A] \subseteq S[pkey(S)]$	$R = sourceOf(C)$	$\forall v \in \pi_{B_1}(r)$
$pkey(S) = \{B_0\}$		class($\boldsymbol{C_v}$ partial C)
$S = \{B_0, B_1\}$		
$\lvert\pi_{B_0}(r)\rvert = \lvert\pi_{B_1}(r)\rvert$		

Table 2. Complex rules for hierarchy generation based on identification of categorizing attributes ($A = catAtt(r)$). Within the target part of the rule, the variable in bold holds the Uri of the generated fragment in the ontology.

the tuples according to values of the categorizing attribute. Formally, for each value v of A^*, the corresponding class is populated with the instances derived from the tuples of the set $\sigma_{A^*=v}(r) = \{t \in r \mid t[A] = v\}$.

5 Evaluation

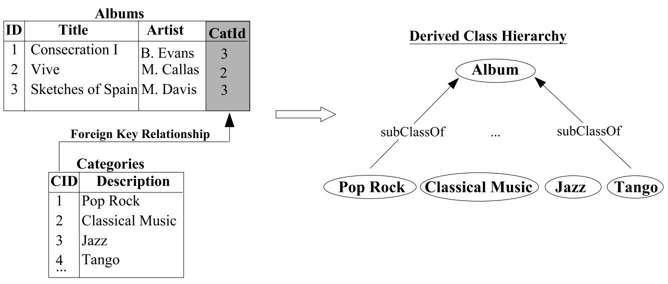
RTAXON has been evaluated on a set of 35 databases from different domains. These databases included 60 categorizing attributes. The method provided exploitable results with a precision of 65% and recall of 60%. In 30% of these cases, several candidates resulted from the first filtering step based on the lexical clues. The conflicts were resolved by invoking the complementary step based on data diversity estimation. 62% accuracy was achieved by this conflict resolution step. To better assess the relevance of the entropy-based selection method used at this stage, we experimented simpler selection methods, such as selecting the attribute with the least number of distinct values. Our method achieved the best overall performance.

Our experiments also include a large-scale case study in the domain of aircraft maintenance (see TAO project website[4]).

6 Conclusion and Further Work

We presented a novel approach to ontology learning from relational databases that shows how well-structured ontologies can be learned by combining a classical analysis of the database schema with a task specifically dedicated to the identification of categorization patterns in the data. The

[4]http://www.tao-project.eu/



Albums

ID	Title	Artist	CatId
1	Consecration I	B. Evans	3
2	Vive	M. Callas	2
3	Sketches of Spain	M. Davis	3

Categories

CID	Description
1	Pop Rock
2	Classical Music
3	Jazz
4	Tango
...	

Figure 2. An example of a categorization pattern where the categories to be employed for hierarchy generation are further defined in an external relation

formalized method is fully implemented and included in the RDBToOnto platform as the main learning component. The method was validated on a representative set of databases.

A major direction for improvement is the extension of the method to deal with the identification of more complex categorization patterns. Our implementation already provides some support for the generation of two-level hierarchies based on two categorizing attributes. However, the pattern identification step is not covered as the two concerned attributes should be given as input to the process.

References

[1] I. Astrova. Reverse engineering of relational databases to ontologies. In *1st European Semantic Web Symposium (ESWS 2004)*, Greece, 2004. Stringer-Verlag.

[2] J. Barrasa, O. Corcho, and A. Gomez-Pérez. R2O, an extensible and semantically based database-to-ontology mapping language. In *Proc. of SWDB 2004*, Toronto, 2004.

[3] A. Behm, A. Geppert, and K. R. Dittrich. On the migration of relational schemas and data to object-oriented database systems. In *Proc. of RETIS 97*, Austria, 1997.

[4] C. Bizer. D2R MAP - a database to RDF mapping language. In *Proc. of WWW 2003*, Budapest, 2003.

[5] F. Cerbah. Learning highly structured semantic repositories from relational databases: The RDBToOnto tool. In *Proc. of ESWC 2008*, Tenerife, 2008.

[6] N. Lammari, I. Comyn-Wattiau, and J. Akoka. Extracting generalization hierarchies from relational databases. a reverse engineering approach. *Data and Knowledge Engineering*, 63, 2007.

[7] M. Li, X. Du, and S. Wang. Learning ontologies from relational databases. In *Proc. of Int. Conference on Machine Learning and Cybernetics*, volume 6. IEEE, 2005.

[8] S. Ramanathan and J. Hodges. Extraction of object-oriented structures from existing relational databases. *ACM SIGMOD*, 26(1), 1997.

[9] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In *ACM Symp. on Applied Computing (SAC 02)*, Madrid, 2002.

[10] Z. Tari, O. A. Bukhres, J. Stokes, and S. Hammoudi. The reengineering of relational databases based on key and data correlations. In *DS-7*, 1997.