ON THE INTEGRATION OF LARGE DATABANKS BY A POWERFUL CATALOGUING METHOD

Zsolt T. KARDKOVÁCS*, Gábor M. SURÁNYI and Sándor GAJDOS

Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H–1117 Budapest, Magyar tudósok krt. 2.
* Office Phone: (+36) 1 4632031
* Office Fax: (+36) 1 4633107
e-mail: {lodoktor, surprof, gajdos}@db.bme.hu
WWW: https://db.bme.hu/DataIntegration

Received: December 14, 2003

Abstract

The integration of huge, disparate information sources has been an outstanding issue for more than a decade. Having realized that the dissimilarity of sources even with the same or related subject is manifold and difficult to tackle, constructing mediators and wrappers has become the common practice. Although under certain circumstances this approach delivers satisfactory results, it lacks the most important property: scalability. Furthermore, it gives no support for discovering similarity, which is often needed whenever no exact match can be returned for a particular search condition.

For these reasons we address the problem of true unification of data sources in the present paper. We assume that the sources share a common schema, i.e. the main objective is the identification of compatible or identical entities. Our novel method accomplishes this task by automatically establishing a catalogue of data elements. Each category of the catalogue holds a set of compatible or identical items. This organization structure has two advantages: there is an intrinsic, fast lookup method, and similarity among data elements can be defined. We prove that the catalogue is theoretically computable even if the schema contains derived attributes.

Keywords: automated data integration, data identity, data similarity, relational database, 0NF schema, catalogue.

1. Introduction

The integration of huge, disparate information sources has been an outstanding issue for more than a decade. However, the exact need for integration and its goal have varied.

At the beginning, the computer infrastructure was developing bottom-up, every department of an enterprise dealt with different aspects of production or service on their own. Under the increasing pressure of competitors the separate databanks were integrated and the integrated system was used as a tool of better management. This kind of integration required only common schemata instead of consistent views of all data elements. Numerous methodologies aiming at view integration were proposed and applied, for a survey see e.g. [1].

Later the Internet gained never forecasted popularity, the world we live in consequently became smaller, and data available to everyone has reached immeasurable amounts. Therefore, in order to cope with this enormous quantity of data, just storing them is no longer satisfactory, relationships among data items need to be discovered and interpreted. Besides contributing a solution for the problem, a great account of efforts towards data integration is YERNENI's thesis [9]. A common property of these approaches is the use of mediators and wrappers, i.e. only a virtual integration is achieved.

In the present paper we address the problem of true integration of data sources, i.e. the problem of automatically establishing common structure for data originating from heterogeneous sources and migrating the data elements into the structure with respect to their identity and similarity. The main advantage of this solution is obvious: the original sources do not have to be retained as legacy systems if unnecessary. We also identify novel methods of data retrieval from the new structure.

For simplicity, however, our proposal presumes that the databanks to be integrated share a common schema as a result of, for example, any method described in [1]. Moreover, to represent this, we exclusively use 0th normal form (0NF) relational schema in which attributes can have multiple (i.e. set) values.[1] The reason for such a representation is that it can easily be produced from either relational, or logical, or object-oriented databases and even from stand-alone data records.

In the next section novel pre-orders for data residing in 0NF schemata are introduced. Section 3 is about how these orders can be interpreted in object-oriented models. It also generalizes them to make then applicable not only to attributes but observing methods as well. A new data structure is built and several retrieval methods are presented in Sections 4 and 5. Section 6 concludes the contribution. Notions are illustrated by means of examples throughout the paper.

2. Substitutions and Covers

Further on in this paper let us assume that there are several finite data stores but they all share a common, extended, 0NF relational schema, denoted by Ω. Defining a satisfactory common schema is usually a difficult task but in theory the union of all attributes of individual schemata is sufficient. Then each element of the data stores contains a NULL value (hence the schema is extended relational) for each attribute which is not defined in the original schema of the element, the rest of attribute values are left intact. The unified relation over Ω is denoted by $\mathcal{U}$. NULL values are consistently considered as empty sets.

Capital letters from the beginning of the alphabet are used for elements of the relation (i.e. the tuples). Capital letters from the end of the alphabet stand for variables. The fields (i.e. attributes) of relations are denoted by lower case letters.

[1] According to several interpretations, 0NF relational schemata may also contain compound attributes. From our point of view compound attributes make no difference, results described herein do not apply to simple attributes only.

The dot (.) is used to project the element on its left side to an attribute indicated on the right side. For example, the notation $C.m$ is used to refer to the attribute m of the element C. In graph-related formulae, the symbol v with or without index stands for a vertex of a graph, not for an attribute. $U(m)$ stands for the set of possible values of the attribute m and as a definition

$$U(\mathcal{U}) = \bigcup_m U(m).$$

In the following paragraphs not only set relations but also logical implications are used. Hence, for clarity we note that $\subseteq$ stands for the 'subset or identical' relation and $\supset$ always designates a logical implication.

Definition 1 (ε_m-substitutability) Let C and D be elements of $\mathcal{U}$ over Ω and m an attribute in Ω. Moreover, let ε_m be a pre-order on $U(m)$, i.e. a transitive and reflexive function that maps pairs of subsets of $U(m)$ to truth values. C is substitutable by D concerning m, denoted by $C \preceq_{\varepsilon_m} D$ if

$$\varepsilon_m(C.m, D.m) = \text{true}.$$

Example 1 Let $\Omega = \{\text{relationship}\}$ and $\mathcal{U}$ contain different relationship types. For simplicity, we assume Ω is in 1NF. Let $\varepsilon_{\text{relationship}}$ be interpreted as the order of strength. Then, for example

$$\text{acquaintance} \preceq_{\varepsilon_{\text{relationship}}} \text{friend} \preceq_{\varepsilon_{\text{relationship}}} \text{sibling}.$$

The concept of substitutability can be generalized to a set of attributes as follows.

Definition 2 ($\varepsilon_{\mathcal{M}}$-covering) Let $\mathcal{M} = \{m_1, m_2, \ldots m_i\}$ be a finite set of attribute names in Ω and for each $m \in \mathcal{M}$ ε_m a pre-order on $U(m)$ as previously. C covers D concerning the set of relations $\varepsilon_{\mathcal{M}} = \{\varepsilon_{m_1}, \ldots \varepsilon_{m_i}\}$ and $\mathcal{M}$, denoted $C \sqsubseteq_{\varepsilon_{\mathcal{M}}} D$, if

$$\forall m \quad m \in \mathcal{M} \supset C \preceq_{\varepsilon_m} D.$$

It is simple to show that the generalized relation is a pre-order, too, as the following proposition says:

Proposition 1 *$\varepsilon_{\mathcal{M}}$-covering is a pre-order on any relation over Ω.*

Example 2 $\varepsilon_{\mathcal{M}}$-covering has a greater expressive power than ε_m-substitutability. For instance let $\mathcal{U}$ store acquaintances of a specified person, more precisely their name, their language abilities and the type of relationship to the given person. Furthermore, $\mathcal{M} = \{\text{relationship, languages}\}$, $\varepsilon_{\text{relationship}}$ is defined as in the previous example and $\varepsilon_{\text{languages}} = \subseteq$. If $C = \langle\text{friend}, \{\text{English, German}\}\rangle$ and $D = \langle\text{acquaintance}, \{\text{English}\}\rangle$, then $D \sqsubseteq_{\varepsilon_{\mathcal{M}}} C$ expresses that C is more valuable than D for the given person.

As it may have been noticed, the subset relation is of great importance in the case of 0NF schemata.

3. Object-Oriented Covers

The commonly used object-oriented data model has a characteristic feature compared to other data models: methods are definable. There are basically two types of methods. Some of them only observe the current status of object instances, some of them also modify it. Here we consider only observing methods; extension to other methods is part of our future research plans.

Before including operations in covering relations, let us examine a covering relation in the case of two entities, C and D. Assuming that $\varepsilon = \subseteq$, $\mathcal{M} = \Omega$ and $C \sqsubseteq_{\varepsilon_{\mathcal{M}}} D$, all attributes of C occur in D. That means, by definition, a subset relation between the attribute values of the entities holds. In a functional point of view (see e.g. [3]) attributes are also functions but with no parameters, thus covariance on the codomains (a subset relation of return values) is the only condition of subtyping. The subtyping relation is referred to as *type conformance* in the widely used Unified Modeling Language [6].

Observing methods have exactly the same functionality as derived attributes. Therefore, the body (implementation) of these operations actually realizes derivation functions. In the case of deductive object-oriented databases (see e.g. [5]), these functions are also called (derivation) rules since they are encoded in universally quantified implication formulae. In these formulae the heads contain only a predicate symbol corresponding to the function represented. The arity of the predicate symbol is by one greater than the arity of the function.

Example 3 Age of persons is calculated from the year of birth and the current year. This can be formalized as:

$$\forall \text{current\_year}\ \forall \text{age}\ \text{year\_of\_birth+age=current\_year} \supset \pi(\text{current\_year, age}),$$

where π is a predicate symbol, year_of_birth is an attribute (constant), current_year is the parameter and age is being defined.

Clearly, results of observing operations must be able to be reflected in covering relations. As these methods are based on attribute values, extending covering to methods may seem superfluous and trivial. But derived properties can change even if the internal state of an entity is not modified.

Example 4 (previous continued) current_year increases constantly, while year _of_birth is constant, resulting in an also increasing age.

The only possible way to ensure or to check covering in such cases is to consider the derivation rules themselves. In other words, from the derivation rules it must be decidable which entity covers which other ones.

Definition 3 (ε_m-substitutability, logical form) Let C and D be elements of $\mathcal{U}$ over Ω and ε_m a pre-order on $U(m)$. Moreover, let π denote the predicate corresponding to the function that takes external quantities and delivers the value for

m of any entity. π is actually defined by the formula ψ. Since π is based on actual internal attributes, it can vary with the entity in question. Thus, π_C (or π_D) stands for the predicates corresponding to the derivation functions in C (or D), and ψ_C (or. ψ_D) is the actual realization. Attributes become nullary function symbols in the formulae, but the same attribute should be denoted differently (by separate symbols) in ψ_C and ψ_D.

C is ε_m-substitutable by D, denoted $C \preceq_{\varepsilon_m} D$, if

$$\psi_C \wedge \psi_D \wedge \forall \underline{X} \forall Y_C \forall Y_D \, \pi_C(\underline{X}, Y_C) \wedge \pi_D(\underline{X}, Y_D) \supset \varepsilon_m(Y_C, Y_D),$$

where $\underline{X}$ represents all the external quantities affecting m.

Theorem 2 *$\varepsilon_{\mathcal{M}}$-covering is always decidable.*

Proof. According to [2], the formula in Definition 3 belongs to the Bernays-Schönfinkel-Ramsey class and the class is proven to be decidable, i.e. ε_m-substitutability is decidable. Since $\mathcal{M}$ contains only a finite number of attributes, $\varepsilon_{\mathcal{M}}$-covering is always decidable, too. □

4. Building Catalogues

$\varepsilon_{\mathcal{M}}$-covering serves as a basis for the catalogue, which is defined formally as follows.

Definition 4 (Full $\varepsilon_{\mathcal{M}}$-catalogue) Let $G_{\varepsilon_{\mathcal{M}}} = \langle V, E \rangle$ be a graph, where $V = \mathcal{U}$, $\langle v_i, v_j \rangle \in E$ if $v_i \in V$, $v_j \in V$, $v_i \sqsubseteq_{\varepsilon_{\mathcal{M}}} v_j$ and $v_i \neq v_j$. $G_{\varepsilon_{\mathcal{M}}}$ is also called full $\varepsilon_{\mathcal{M}}$-catalogue.

In any full catalogue there may be elements that cover each other. Moreover the following proposition holds:

Proposition 3 *The strongly connected components of $G_{\varepsilon_{\mathcal{M}}}$ are cliques.*

From the user's point of view the elements of cliques are indistinguishable concerning the attributes in $\mathcal{M}$ of $\varepsilon_{\mathcal{M}}$-covering. Therefore, it is reasonable to consider them as single entities. To this end we adopt a usual technique by introducing an equivalence relation based on $\varepsilon_{\mathcal{M}}$.

Definition 5 ($\triangleleft_{\mathcal{M}}$ partial order) Let an equivalence relation $\equiv_{\mathcal{M}}$ over $\mathcal{U}$ be defined as follows:

$$C \equiv_{\mathcal{M}} D \quad = \quad C \sqsubseteq_{\varepsilon_{\mathcal{M}}} D \wedge D \sqsubseteq_{\varepsilon_{\mathcal{M}}} C.$$

Then $\varepsilon_{\mathcal{M}}$ induces a partial order $\triangleleft_{\mathcal{M}}$ on $\mathcal{U}/\equiv_{\mathcal{M}}$, i.e. on the equivalence classes of $\mathcal{U}$.

$$\begin{tikzcd}
David \arrow[d] & & Valéria \arrow[d] \\
Sam \arrow[dr] & & Mária \arrow[dl] \\
& Joe &
\end{tikzcd}$$

Fig. 1. $\varepsilon_{\mathcal{M}}$-catalogue for Example 5

Definition 6 ($\varepsilon_{\mathcal{M}}$-catalogue) The transitive reduction, i.e. the Hasse diagram of $\triangleleft_{\mathcal{M}}$ is the $\varepsilon_{\mathcal{M}}$-catalogue.

The name catalogue comes from business-to-business (B2B) commerce where content standards establish hierarchies of product categories, specify the attributes of each category and products are classified according to these specifications [7]. Catalogues used in our method act similarly: there is a hierarchy of categories and the categories contain the data elements. But unlike B2B catalogues, here an entity may have unrelated supercategories, which in turn enables more flexible (ambiguous) classification to be modelled. That also means that multiple entry points are conceivable.

Definition 7 (Entry Points of a $\varepsilon_{\mathcal{M}}$-catalogue) A node, v of $\varepsilon_{\mathcal{M}}$-catalogue is an entry point if there is no node v_i such that $v_i \sqsubseteq_{\varepsilon_{\mathcal{M}}} v$.

Example 5 Let us assume that Ω represents names of persons along with the languages they speak and the names of programming languages they know. The contents of $\mathcal{U}$ is:

name	*languages*	*experience*
David	{German}	{Java}
Joe	{English, German, Hungarian}	{C#, Java, Prolog}
Mária	{English, Hungarian}	{C#, Java, Prolog}
Sam	{English, German}	{C#, Java}
Valéria	{Hungarian}	{Prolog}

The $\varepsilon_{\mathcal{M}}$-catalogue defined by $\mathcal{M} = \{\text{languages, experience}\}$ and $\varepsilon_{\text{languages}} = \varepsilon_{\text{experience}} = \subseteq$, is depicted in *Fig. 1.* The entry points of the catalogue are David and Valéria.

5. Searching the Catalogue

The catalogues introduced in the previous section can be considered indexes, because they support efficient, lookup-optimized management methods for the entities involved.

Definition 8 (Search Pattern) τ, a function mapping each attribute m of Ω to $U(m)$ is called the search pattern.

Algorithm 1 (Simple Search in the Catalogue)
simpleSearch$(\mathcal{M}, \tau)$:

- For **all** entry points as node, call simpleSearchNode$(\text{node}, \mathcal{M}, \tau)$.

simpleSearchNode$(\text{node}, \mathcal{M}, \tau)$:

1. If $\tau \equiv_{\mathcal{M}}$ node, emit the elements of node as result.
2. Let nextNode be **any** of the direct successors of node, for which

$$\text{nextNode} \sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau$$

holds. If there is no such nextNode, return.

3. Call simpleSearchNode$(\text{nextNode}, \mathcal{M}, \tau)$.

The correctness of the simple search method is obvious except the selection method in step 2 of simpleSearchNode. It says that any of the successor nodes is proper if the criterion holds.

Proof. [indirect] Let us assume that although nextNode is a successor of the current node and nextNode $\sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau$, τ cannot be reached with further steps from nextNode. It means there is no path from nextNode to τ. That contradicts the definition of the catalogue as the transitive closure of a Hasse diagram is the same as the original partial order. □

Because every path to the requested element is equally proper, an advanced search implementation may examine them in parallel. It can be quicker since the search stops as soon as the next element should be after the search pattern in the catalogue. (This does not apply to entry points. All entry points must be processed.)

Elements of a database which satisfy a minimum requirement are easily retrievable from the catalogue, too. Analogously, a maximum requirement can also be specified. These methods are called lower and upper bound queries, respectively.

Algorithm 2 (Lower Bound Search in the Catalogue)
lowerSearch$(\mathcal{M}, \tau)$:

- For **all** entry points as node, call lowerSearchNode$(\text{node}, \mathcal{M}, \tau)$.

lowerSearchNode$(\text{node}, \mathcal{M}, \tau)$:

1. If $\tau \sqsubseteq_{\varepsilon_{\mathcal{M}}}$ node, call lowerSearchAll(node) and return.
2. Call lowerSearchNode(nextNode, $\mathcal{M}$, τ) with all successors of node as nextNode for which

$$\text{nextNode} \sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau.$$

lowerSearchAll(node):

1. Emit the elements of node as result.
2. Call lowerSearchNode(nextNode) with all successors of node as nextNode.

Algorithm 3 (Upper Bound Search in the Catalogue)
upperSearch($\mathcal{M}$, τ):

- For **all** entry points as node, call upperSearchNode(node, $\mathcal{M}$, τ).

upperSearchNode(node, $\mathcal{M}$, τ):

1. If node $\sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau$ does not hold, return.
2. Emit the elements of node as result.
3. Call upperSearchNode(nextNode, $\mathcal{M}$, τ) with all successors of node as nextNode.

As you may have noticed, there is no option at the selection of next element in the lower and upper bound search algorithms. (All successors are mentioned always.) This does not imply that common subcatalogue parts must necessarily be processed multiple times from all supercategories. Already visited parts may be omitted. The implementation of that variant is not especially difficult with system support. Consider, for instance, the tabled evaluation of XSB Prolog [8].

Definition 9 (Data Similarity) Concerning the particular ε_m-substitutability relations, the direct neighbours, i.e. the direct predecessors and successors in an $\varepsilon_{\mathcal{M}}$-catalogue contain the different but most similar elements to a node. They are the items which cover or are covered by the node without any intermediate entity.

Definition of data similarity enables the design of a similarity search algorithm. However, an algorithm can also return results when there is no exact match for the condition in the catalogue.

Algorithm 4 (Similarity Search in the Catalogue)
similarSearch($\mathcal{M}$, τ):

- For **all** entry points as node, call similarSearchNode(node, NULL, $\mathcal{M}$, τ).

similarSearchNode(node, prevNode, $\mathcal{M}$, τ):

1. If node $\sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau$ does not hold, jump to step 10.
2. Select all nextNode successors of node into nextLess for which

$$\text{nextNode} \sqsubseteq_{\varepsilon_{\mathcal{M}}} \tau.$$

3. If nextLess $= \emptyset$, jump to step 6.

4. For each nextNode $\in$ nextLess call similarSearchNode(nextNode, node, $\mathcal{M}, \tau$).
5. Return.
6. If
$$\tau \sqsubseteq_{\varepsilon_{\mathcal{M}}} \text{node} \land \text{prevNode} \neq \text{NULL}$$
holds, emit the elements of prevNode as result.
7. If
$$\tau \sqsubseteq_{\varepsilon_{\mathcal{M}}} \text{node}$$
is not true, emit the elements of node as result.
8. Emit all elements of nextNode successors of node as result for which
$$\tau \sqsubseteq_{\varepsilon_{\mathcal{M}}} \text{nextNode}.$$
9. Return.
10. If $\tau \sqsubseteq_{\varepsilon_{\mathcal{M}}}$ node, emit the elements of node as result.

Example 6 To demonstrate the usefulness and usability of our similarity definition, consider the following example.

A regular customer of a travel agency is looking for a nice place to spend the summer at. His preferences are:

- a peaceful resort,
- with nearby museums to visit and a beach,
- reachable by plane *or* ship,
- it costs about 1000d.

In order to fulfil the customer's demands the data about all available holiday resorts can be reorganized with $\mathcal{M} = \{\text{properties, accessibility, price}\}$, $\varepsilon_{\text{properties}} = \varepsilon_{\text{accessibility}} = \subseteq$, $\varepsilon_{\text{price}} = \leq$ (the order on real numbers) and the customer's preferences can be represented as

$$\begin{aligned} \tau(\text{properties}) &= \{\text{restful, beach, museums}\}, \\ \tau(\text{accessibility}) &= \{\text{plane, ship}\}, \\ \tau(\text{price}) &= 1000. \end{aligned}$$

Then the best matching candidates are returned by a similarity search. The search also treats the accessibility condition properly, since adequate places reachable by plane *and* ship are listed if there is any, ensuring maximal customer satisfaction.

6. Conclusions

In this paper an automated data reorganization method suitable for true database integration (i.e. without wrappers and mediators) has been proposed. The resulting data structure is in the form of a catalogue, whose nodes comprise elements considered to be identical according to a subset of their attributes. The reorganization method also copes with derived attributes (called observing methods in the object-oriented terminology). We have proved that catalogue construction is a decidable problem even in the presence of derivation rules.

The catalogue presented is non-strict in the sense that ambigous classification is allowed, and that is, in fact, the key to speeding up data retrieval. Search methods have also been described extensively including not only exact but also lower and upper bound queries.

Another significant result of the paper is the ability of responding with similar data elements if requested. This feature is especially beneficial at customer interfaces because exact match to a customer query is a rarity.

Only databanks with common schemata have been discussed in the paper. However, [4] extends this method and concerns data sources with different schemata.

References

[1] BATINI, C. – LENZERINI, M. – NAVATHE, S. B., A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, **18**(4) (1986), pp. 323–364.

[2] BÖRGER, E. – GRÄDEL, E. – GUREVICH, Y., *The Classical Decision Problem*, Springer Verlag, 1997.

[3] CASTAGNA, G., *Object-Oriented Programming: A Unified Foundation*, Birkhäuser, 1997.

[4] KARDKOVÁCS, ZS. T. – SURÁNYI, G. M. – GAJDOS, S., Application of Catalogues to Integrate Heterogeneous Data Banks, in: Robert Meersman and Zahir Tari, eds. *OTM Workshops 2003*, Vol. 2889 of LNCS, pp. 1045–1056, Springer-Verlag, 2003.

[5] KIFER, M. – LAUSEN, G. – WU, J., Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of ACM*, **42**(4) (1995), pp. 741–843.

[6] Object Management Group, *Unified Modeling Language Specification (Version 1.5)*, March, 2003.

[7] OMELAYENKO, B. – FENSEL, D., An Analysis of B2B Catalogue Integration Problems, *Proc. of International Conference on Enterprise Information Systems (2)*, (2001), pp. 945–952.

[8] *The XSB System (Version 2.5)*, 27 June, 2003.

[9] YERNENI, R., *Mediated Query Processing Over Autonomous Data Sources*, Stanford University, August, 2001.