Intensional Associations in Dataspaces

Marcos Antonio Vaz Salles [#1], Jens Dittrich [*2], Lukas Blunschi [+3]

[#]*Cornell University* [*]*Saarland University* [+]*ETH Zurich*

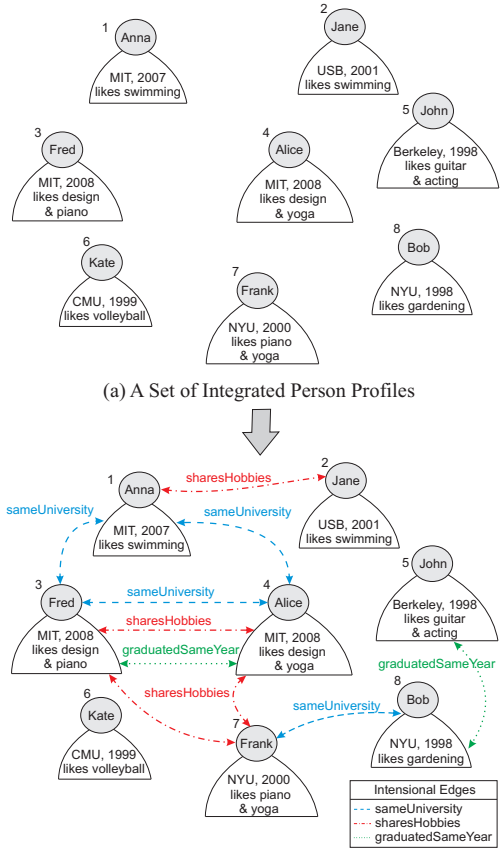
[1]vmarcos@cs.cornell.edu [2]jens.dittrich@cs.uni-sb.de [3]lukas.blunschi@inf.ethz.ch

***Abstract*— Dataspace applications necessitate the creation of associations among data items over time. For example, once information about people is extracted from sources on the Web, associations among them may emerge as a consequence of different criteria, such as their city of origin or their elected hobbies. In this paper, we advocate a declarative approach to specifying these associations. We propose that each set of associations be defined by an *association trail*. An association trail is a query-based definition of how items are connected by intensional (i.e., virtual) association edges to other items in the dataspace. We study the problem of processing neighborhood queries over such intensional association graphs. The naive approach to neighborhood query processing over intensional graphs is to materialize the whole graph and then apply previous work on dataspace graph indexing to answer queries. We present in this paper a novel indexing technique, the grouping-compressed index (GCI), that has better worst-case indexing cost than the naive approach. In our experiments, GCI is shown to provide an order of magnitude gain in indexing cost over the naive approach, while remaining competitive in query processing time.**

I. Introduction

Dataspace systems have been envisioned as a new architecture for data management and information integration [1]. The main goal of these systems is to model, query, and manage relationships among disparate data sources. So far, relationships in these systems have been specified at the set or schema level [2]. In several dataspace scenarios, however, it is important to model associations between individual data items across or within data sources. Such scenarios include social content management [3] and personal information management [4]. In personal information management, for example, it is useful to associate messages and documents received in the same context or timespan.

In this paper, we propose a declarative approach, called *association trails*, to specifying associations among items in a dataspace. An association trail is a query-based definition of how items in the dataspace are connected by virtual association edges to other items. A set of association trails defines a logical graph of associations over the dataspace. As this graph is purely logical and in principle does not need to be explicitly materialized, we call it in this paper an *intensional graph*. In addition, we term the edges of this graph *intensional edges* or *intensional associations*. In contrast to solutions that define associations extensionally [5], [6], [7], association trails define associations logically and in bulk. As a consequence, association trails are especially useful when data sources have no explicit associations defined beforehand. In the following, we illustrate association trails with an example.



(a) A Set of Integrated Person Profiles

(b) An Intensional Graph defined by Association Trails

Fig. 1. A set of person profiles extracted from the Web is transformed by association trails into an intensional graph of associations.

A. Example

While our techniques are applicable to many dataspace scenarios, we will use as a running example for this paper the modeling of an implicit social network. Figure 1(a) shows a set of person profiles extracted from Web sources using information extraction techniques. Each profile states a person's name, along with the university she has attended, her year of graduation, and her hobbies.

Example 1 (Implicit Social Networks) *Users would like to navigate their social dataspaces to find other users related to them or to their topics of interest. Unfortunately, data extracted from loosely-connected sources is poor in associations among users.*

State of the art: Users may search their dataspaces with keyword search engines. These systems have no knowlegde about associations between data items. As a consequence, they

return only items that match the user's specific request and cannot enrich results with other relevant associated information. While previous approaches extend search results with elements in a dataspace or schema graph [7], [8], they are of little use when connections are not explicitly defined as in Figure 1(a). Users could rely, instead, on a recommender system [9]. These systems are, however, limited to hard-coded heuristics such as TF-IDF similarity to recommend related profiles. Moreover, these systems do not model associations among items, so users cannot extend the system by defining associations based on new criteria.

Our goal: We provide a declarative technique to model intensional associations among items in the dataspace. Users or administrators provide to the system declarative association definitions, called *association trails*. Consider that the following association trails are given to the system:

1. People that went to the same university are related.
2. People that graduated on the same year are related.
3. People that share a hobby are related.

Now, users will have the view of the dataspace depicted in Figure 1(b). They are able to browse a rich graph of associations. In addition, search query results may be enriched by items in their neighborhoods in the intensional graph. □

B. Contributions

In summary, this paper makes the following contributions: (1) We present a new declarative formalism, *association trails*, to define an intensional graph of connections among instances in a dataspace (Section III; (2) We propose a new query processing technique, the grouping-compressed index (GCI), for neighborhood queries over such intensional graphs (Section IV); (3) We show experimentally that GCI brings an order of magnitude gain in indexing cost over the naive approach, while remaining competitive in query processing time (Section V).

II. Preliminaries

Data and Query Model. In a nutshell, we assume the dataspace to be represented in a graph data model and queried by end-users with keyword queries. A *dataspace* is represented by a graph $G := (N, E)$, where N is a set of nodes and E is a set of directed edges. Each node $N_i \in N$ is a set of attribute-value pairs $N_i := \{(a_1^i, v_1^i), \ldots, (a_k^i, v_k^i)\}$, where each value is either atomic or a bag of words. As can be easily seen, the data in Figure 1(a) is represented in this data model. Note that $E = \emptyset$ and attribute names have been omitted for visibility.

A *query* Q is an expression that selects a set $Q(G) \subseteq N$. Admissible queries are conjunctions and disjunctions of filters on attribute values and keywords. For example, the query `yoga university=NYU` returns Node 7 in Figure 1(a).

Basic Index Structures. Given the queries above, we assume two basic index structures, commonly found in state-of-the-art search engines. First, an *inverted index* is a mapping from token to the list of node identifiers of nodes containing that token. We represent tokens as concatenations of keywords with the attribute names and translate keyword queries into prefix queries to the index [7]. Second, a *rowstore* is a mapping from node identifier to the information (i.e., attribute-value pairs and edges) associated with that node.

III. Association Trails

This section formalizes association trails and neighborhood queries over intensional graphs.

A. Basic Form of an Association Trail

An association trail defines a set of edges in the intensional graph. For example, in Figure 1(b), a single association trail defines all sharesHobbies edges. We may thus interpret each association trail as defining an *intensional graph overlay* on top of the original dataspace graph.

Definition 1 (Association Trail) *A unidirectional association trail is denoted as*

$$A := Q_L \overset{\theta(l,r)}{\Longrightarrow} Q_R,$$

where A is a label naming the association trail, Q_L, Q_R are queries, and θ is a predicate. The query results $Q_L(G)$ are associated to $Q_R(G)$ according to the predicate θ, which takes as inputs one query result from Q_L and one from Q_R. Thus, we conceptually introduce in the association graph one intensional edge, directed from left to right and labeled A, for each pair of nodes given by $Q_L \bowtie_\theta Q_R$. We require that the node on the left of the edge be different than the node on the right, i.e., no self-edges are allowed.

A bidirectional association trail is denoted as

$$A := Q_L \overset{\theta(l,r)}{\Longleftrightarrow} Q_R.$$

The latter also means that the query results $Q_R(G)$ are related to the query results $Q_L(G)$ according to θ. □

An association trail relates elements from the data sources by a *join predicate* θ. Therefore, association trails cover relational and non-relational theta-joins as special cases. While knowing the form of θ may allow us to improve performance (see Section IV-B), conceptually θ may be an arbitrarily complex function. This means that Definition 1 also models use cases such as content equivalence and similar documents.

A straightforward extension to our model is to define θ as a matching function generating several edges between a pair of nodes. This may be useful to model individual matches created by multi-valued attributes, e.g., modeling each hobby match in sharesHobbies by a separate intensional edge.

Use Case 1 (Social Networks) The intensional graph of Figure 1(b) is defined by the following association trails:

$$\begin{aligned}
\text{sameUniversity} &:= \texttt{class=person} \overset{\theta_1(l,r)}{\Longleftrightarrow} \texttt{class=person},\\
&\quad \theta_1(l,r) := (l.university = r.university).\\
\text{graduatedSameYear} &:= \texttt{class=person} \overset{\theta_2(l,r)}{\Longleftrightarrow} \texttt{class=person},\\
&\quad \theta_2(l,r) := (l.gradYear = r.gradYear).\\
\text{sharesHobbies} &:= \texttt{class=person} \overset{\theta_3(l,r)}{\Longleftrightarrow} \texttt{class=person},\\
&\quad \theta_3(l,r) := (\exists h \in l.hobbies : h \in r.hobbies).
\end{aligned}$$

These association trail examples show how to define a logical graph of associations among elements in the dataspace. They use queries that select elements to be related and predicates that specify join semantics among those elements. When taken together, the association trails above result in a multigraph (displayed in Figure 1(b)). This intensional multigraph is actually a view, which can be refined over time by adding more association trails in a pay-as-you-go fashion. □

B. Neighborhood Queries

We focus on a special class of exploratory queries over intensional graphs termed *neighborhood queries*. For simplicity, our presentation in the following sections is focussed on unidirectional association trails, as it is simple to extend our techniques to the bidirectional case.

Neighborhood queries were used by Dong and Halevy to explore a dataspace [7]. They assume that the dataspace graph is given extensionally, i.e., each edge in the graph is explicitly materialized. Unfortunately, their definition does not apply to intensional graphs. As such, we generalize neighborhood queries below to intensional graphs. We first define what a neighborhood is in our context.

Definition 2 (Neighborhood) *Given a query Q and a set of association trails A^*, the neighborhood $N^Q_{A^*}$ of Q with respect to A^* is given by*

$$N^Q_{A^*} := \begin{cases} \emptyset, & \text{if } A^* := \emptyset \\ (Q \cap Q^i_L) \rtimes_{\theta_i} Q^i_R, & \text{if } A^* := \{A_i\} \\ N^Q_{\{A_1\}} \cup N^Q_{\{A_2\}} \cup \ldots \cup N^Q_{\{A_n\}}, & \text{if } A^* := \{A_1, A_2, \ldots, A_n\} \end{cases}$$

where Q^i_L and Q^i_R are the queries on the left and right sides of trail A_i, respectively, and θ_i is the θ-predicate of A_i. □

The definition above states that the neighborhood includes all instances associated through A^* to instances returned by Q. They are obtained by a semi-join that filters all instances in Q^i_R that are connected to elements of Q also appearing in Q^i_L. We ignore self-edges in order to simplify our presentation.

Definition 3 (Neighborhood Query) *For a query Q and a set of association trails A^*, the neighborhood query $\tilde{Q}_{A^*}$ is given by*

$$\tilde{Q}_{A^*} := Q \cup N^Q_{A^*}$$

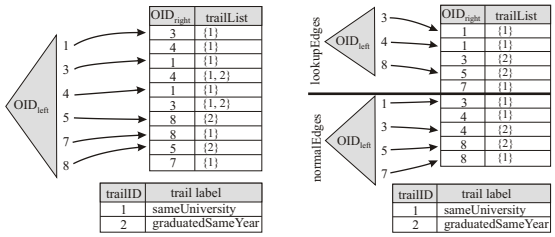
We call the results obtained by Q primary query results *and the results obtained by $N^Q_{A^*}$* neighborhood results. □

IV. Query Processing Techniques

A. Naive Approach

The most intuitive query processing strategy is to explicitly materialize all edges in the intensional graph. At query time, we lookup this materialization to obtain the neighborhoods for each element returned by the original query Q. Other dataspace [7] and graph indexing techniques [5], [6] could also be applied on top of the naive materialization to reduce query time. However, they will make indexing time even larger for the naive approach. As our experiments reveal (Section V), indexing is the most dramatic cost driver for this strategy.

Given a set of association trails $A^* = \{A_1, A_2, \ldots, A_n\}$, the materialization of the intensional graph can be obtained by the join $Q^1_L \bowtie_{\theta_1} Q^1_R \cup \ldots \cup Q^n_L \bowtie_{\theta_n} Q^n_R$. This materialization can be stored in a join index [10], represented by an inverted list as in Figure 2(a). Clearly, if the queries Q^i_L and Q^i_R each return N nodes, then the association trail A_i may generate up to $O(N^2)$ edges in the multigraph.



(a) Naive Approach (Full Materialization)

(b) Grouping-Compressed Index (GCI)

Fig. 2. Query processing alternatives for association trails sameUniversity and graduatedSameYear.

B. Grouping-Compressed Index (GCI)

The quadratic growth in the size of the naive materialization of Section IV-A is often a consequence of grouping effects in the association trail join predicates. In an equi-join, for example, all nodes with the same value for a join key will join with one another, generating a clique in the graph for that association trail. In a clique of size C, we would naively store C^2 edges in the naive materialization.

As an alternative, we could: (1) explicitly represent the edges from a given node n_1 in the clique to all the other nodes $\{n_2, \ldots, n_C\}$ and (2) for each remaining node in the clique, represent special *lookup edges* $\langle n_j, n_1, lookup \rangle$ that state n_j connects to the same nodes as n_1. Thus, we represent the information in the clique with C normal edges for n_1 plus $C - 1$ edges for the lookup edges of all remaining nodes. In short, a reduction in storage space (and consequently join indexing time) from C^2 edges to $2 \cdot C - 1$ edges. We call this representation *grouping compression*. Figure 2(b) shows our representation for the *grouping-compressed index*.

One important challenge we address in our work is how to query GCI without decompressing each clique to its original C^2 edges. The core idea of our method is to operate in two phases. In the build phase, we create a table with aggregate counts from all lookup edges for the original query results. In the probe phase, we may use this table to avoid looking up normal edges for nodes in the same clique several times. Due to space limitations, we refer the reader to Vaz Salles's PhD thesis for the presentation of our query algorithm [11].

V. Experiments

In this section, we evaluate how the grouping-compressed index (GCI) compares to the naive approach (Naive).

Dataset. We have evaluated our system with a real dataset of biographies and filmographies from IMDb [12]. Person profiles were obtained from the biographies of actors, actresses, writers, and directors, totalling 1,909,796 people. Each profile

included the person's first and last names, birthdate, place of birth, and height. In addition, we imported all the explicit connections between people and the movies they worked in. There were a total of 1,414,654 movies and 14,250,548 person-movie connections. We have created the following (self-explanatory) association trails: sameLastName, sameBirthdate, samePlaceOfBirth, sameHeight, and moviesInCommon. The association trail moviesInCommon generates one intensional edge for each movie connecting two persons, following the extension for multi-valued attributes discussed in Section III-A. The association trails create an intensional graph of person nodes over a dataset in which these connections are not explicit.

Setup. All experiments have been run on a dual AMD Opteron 280 2.4 Ghz dual-core server, with 6 GB of RAM, and a 400 GB ATA 7200 rpm hard disk with 16 MB cache. Association trails have been implemented in the open-source iMeMex Dataspace Management System [13].

Dataset	Orig Data Size [MB]	Total Index Size [MB]	Rowstore Size [MB]	Inv Index Size [MB]	Indexing Time [min]
IMDb	564	924	676	248	48

TABLE 1

SIZE AND CREATION TIME FOR BASIC INDEXES (SECTION II)

We report in Table 1 the indexing time and index sizes taken by the index structures of Section II for the IMDb dataset.

Metric	Strategy	
	Naive	GCI
Indexing Time (min)	194	7
Index Size (MB)	4769	170

Results. We report in this section the sensitivity of the methods to the selectivity of the original query Q. First, we show the indexing performance for all methods in the table on the right. In short, Naive's indexing time and index sizes are, respectively, 27 and 28 times larger than GCI, a difference of over an order of magnitude. Figure 3 shows the corresponding query performance of the methods. The query processing times for Naive were highly dependent on query selectivity, increasing sharply as selectivity is lowered. At 10% selectivity, Naive took 2.4 min, a factor 8.8 worse than the 16.2 sec taken by GCI. That behavior is consistent with the fact that lower selectivities imply that a proportionally larger fraction of the fully materialized intensional graph must be processed in order to answer a neighborhood query.

VI. Related Work

Expanding query results by adding context from a neighborhood in the database has been explored in work such as keyword search over relational databases [8] and dataspaces [7]. In contrast to our work, all of this previous work is concerned with extensional graphs and their techniques would have to be revisited when the graph is defined intensionally. The same can be said about recent work on graph indexing [5], [6].

Declarative Networking has been proposed to process multi-hop, recursive queries over a single intensional graph, defined via datalog rules [14]. In contrast, we process *single-hop*, neighborhood queries over *multiple* intensional overlay graphs, defined by association trails. Metadata management techniques, such as [15], are also related to our approach, though they focus on schema-first relational settings while our techniques target schema-later dataspace scenarios.

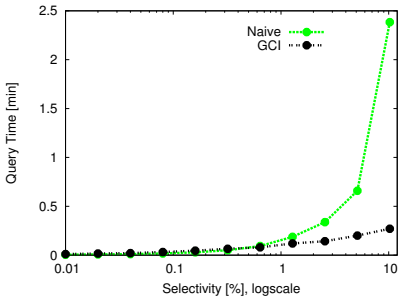


Fig. 3. Query execution time vs. selectivity of original query

VII. Conclusions

In this paper, we have presented *association trails*, a declarative technique to define a logical, intensional graph of associations among instances in a dataspace. Our technique is general and may be applied to a variety of scenarios, such as social networks and personal dataspaces [11]. In addition, we have shown a new neighborhood query processing strategy over intensional association graphs. Our strategy, the grouping-compressed index (GCI), may provide over an order of magnitude gain in indexing time over the naive approach while remaining competitive in query processing time.

Acknowledgements. We thank David Maier for his comments on earlier versions of this work. This work was partially supported by the Swiss National Science Foundation (SNF) under contract 200021-112115.

References

[1] M. Franklin, A. Halevy, and D. Maier, "From Databases to Dataspaces: A New Abstraction for Information Management," *SIGMOD Record*, vol. 34, no. 4, 2005.
[2] M. A. V. Salles *et al.*, "iTrails: Pay-as-you-go Information Integration in Dataspaces," in *VLDB*, 2007.
[3] S. Amer-Yahia, L. V. S. Lakshmanan, and C. Yu, "SocialScope: Enabling Information Discovery on Social Content Sites," in *CIDR*, 2009.
[4] J.-P. Dittrich and M. A. V. Salles, "iDM: A Unified and Versatile Data Model for Personal Dataspace Management," in *VLDB*, 2006.
[5] T. Neumann and G. Weikum, "RDF-3X: a RISC-style Engine for RDF," *JDMR (formerly Proc. VLDB)*, vol. 1, 2008.
[6] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: Sextuple Indexing for Semantic Web Data Management," *JDMR (formerly Proc. VLDB)*, vol. 1, 2008.
[7] X. Dong and A. Halevy, "Indexing Dataspaces," in *ACM SIGMOD*, 2007.
[8] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A System for Keyword-Based Search over Relational Databases," in *ICDE*, 2002.
[9] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE TKDE*, vol. 17, no. 6, 2005.
[10] P. Valduriez, "Join Indices," *ACM TODS*, vol. 12, no. 2, 1987.
[11] M. V. Salles, "Pay-as-you-go Information Integration in Personal and Social Dataspaces," Ph.D. dissertation, ETH Zurich, 2008.
[12] IMDb. `http://www.imdb.com/`.
[13] iMeMex project web-site. `http://www.imemex.org`.
[14] B. T. Loo *et al.*, "Declarative Networking: Language, Execution and Optimization," in *ACM SIGMOD*, 2006.
[15] D. Srivastava and Y. Velegrakis, "Intensional Associations Between Data and Metadata," in *ACM SIGMOD*, 2007.