Exploring Schema Repositories with Schemr

Kuang Chen
University of California, Berkeley
kuangc@cs.berkeley.edu

Jayant Madhavan
Google, Inc.
jayant@google.com

Alon Halevy
Google, Inc.
halevy@google.com

ABSTRACT

Schemr is a schema search engine, and provides users the ability to search for and visualize schemas stored in a metadata repository. Users may search by keywords and by example – using schema fragments as query terms. Schemr uses a novel search algorithm, based on a combination of text search and schema matching techniques, as well as a structurally-aware scoring metric. Schemr presents search results in a GUI that allows users to explore which elements match and how well they do. The GUI supports interactions, including panning, zooming, layout and drilling-in. We demonstrate schema search and visualization, introduce Schemr as a new component of the information integration toolbox, and discuss its benefits in several applications.

Categories and Subject Descriptors: H.3.3 Information Search and Retrieval: Retrieval models

General Terms: Algorithms, Design

Keywords: Schema search

1. INTRODUCTION

Traditionally, the problem of information integration has been framed to assume the existence of relevant schemas, but without considering the problem of locating the schemas. As such, exiting tools for information integration can begin by tackling the problem of *schema matching*. We argue that there are multiple benefits if information integration starts providing tools earlier in the information life-cycle, and helps in locating relevant schemas. Schema search is a tool for locating schemas (or fragments thereof) in larger collections, and therefore enabling reuse of schemas. In a sense, schema search provides a conduit to the wisdom of crowds and past experience, and improves data ecosystems by nurturing schema compatibility, and consequently, information integration.

Schema search can help overcome the high barrier-to-entry to existing information integration solutions. Current solutions are costly and require both data management and subject-matter expertise. Even if software cost were not an issue, high consulting fees make information integration inaccessible to many organizations. Schemr addresses this problem by enabling data administrators to rely on each others' solutions and expertise by sharing schemas. Small organizations, such as non-profit and grass-roots efforts operating in low-resource settings, particularly benefit because they are more willing to share information.

SIGMOD'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
ACM 978-1-60558-551-2/09/06.

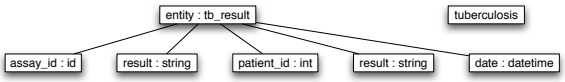

Figure 1: Query graph of a table and a keyword

We ground our work in the challenges and use cases of two such organizations, a nature conservation non-profit, and a large HIV/AIDS treatment program in East Africa. We found that data management in these organizations take place in an ad-hoc manner, and with ad-hoc tools. Data administrators face a vicious cycle: they are overloaded with requests to manually curate data that should be produced by automated processes. Thus, having no time to tackle major system improvements, they create stop-gap solutions. These data administrators said that they would gladly collaborate with others to share schemas and advice, but are hindered by the high-maintenance cost of the stop-gap solutions. They need tools that provide an immediate productivity gain. For these organizations, sharing designs through schema search can provide this bootstrap path, which starts with better data modeling, and leads to better integrated information.

Contributions

- Search algorithm - Schemr's search algorithm combines techniques from document search and schema matching, and employs a holistic *tightness-of-fit* measure to find and rank schemas according to a query's semantic intent.
- Visualizations - Schemr visualizes search results in several views, allowing users to compare multiple results and drill-in to explore a schema with visually encoded similarity measures.
- Open source tool - Schemr is part of an open-source information integration framework, which any organization may use and extend for free.

Example Scenario

We present an example schema search scenario: A rural health clinic wants to persist the results of a new tuberculosis assay. The database administrator (DBA) begins by designing a new table to store the results. She is unsure of the best way to model the new table, and wants to search for related schemas and data examples. She wants to search existing data models by using the keyword "`tuberculosis`" to find schemas relating to the term, and use a partially designed schema to specify results that include elements se-

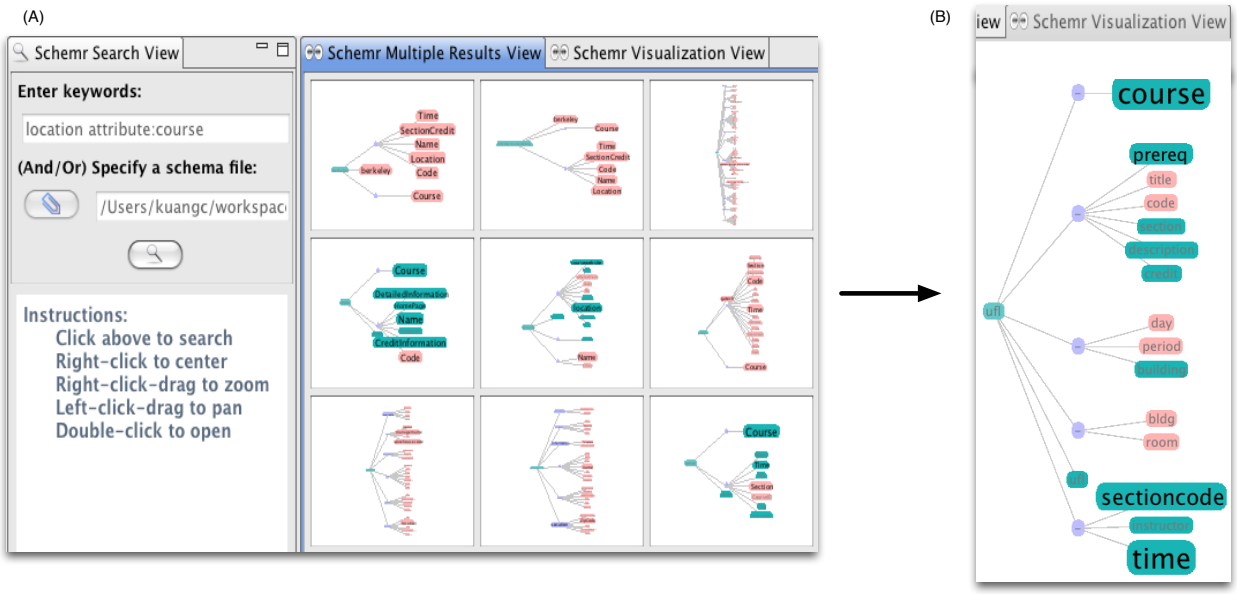

Figure 2: (A) A multiple-results view of search results for a keyword + schema fragment query (B) On drill-in to a search result, the detailed view is visually encoded: color corresponds to schema element types (e.g. entity or attribute); size corresponds to relative normalized quality scores.

mantically equivalent to ones she has already designed.

In Schemr, the user uploads the *DDL* or *XSD* of the unfinished schema, and adds keywords to formulate the query. Schemr parses the input, and creates a *query graph* (Figure 1). Schemr returns a ranked list of n results, presented in a matrix of preview windows. The user can interact with the results or asks for the next n schemas. On drill-in to a particular schema, Schemr creates a detailed view with visual encodings of similarity. Figure 2 shows an example of Schemr's visualizations.

2. SYSTEM DESIGN

Algorithm

Schemr's search algorithm (Figure 3) consists of three phases. The query parser first creates a query-graph from the keyword terms and schema fragments given by user input. In the first phase – Candidate Extraction – Schemr flattens the query-graph into a list of keywords, and quickly retrieves the top candidate schemas from a scalable document index. In the second phase – Schema Matching – Schemr evaluates the top candidate schemas with schema matching techniques [1, 3], scoring the semantic similarity between candidate schema and the query-graph elements. This provides the accuracy necessary for the third phase – Tightness-of-fit Measurement– in which Schemr computes a total score based on the tightness-of-fit among matching elements.

Candidate Extraction: The input query-graph Q is a forest of trees representing of schema fragments and keywords, like in Figure 1. The example shows that Q can represent several graphs, where a keyword is represented as a graph of one item. The query-graph abstraction captures multiple query formats (relational and xml). The system contains a document index of the schema corpus, which we build offline. The document index provides a fast and scalable filter for relevant candidate schemas. We create a list of keywords

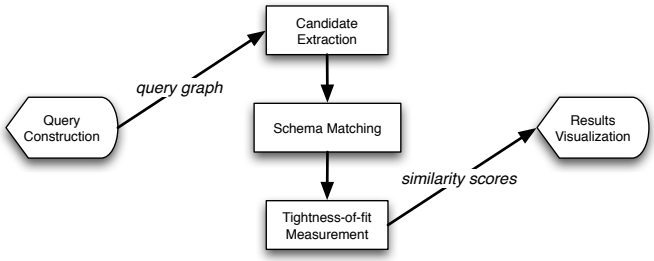

Figure 3: Schema search algorithm data flow

by flattening the query graph Q to query the document index. To preserve recall, the document search algorithm need not match all search terms. Candidate extraction produces the top n schemas for the next phase.

Schema Matching: The top candidate schemas are evaluated against the query-graph using a collection of matchers. For instance, the *name matcher* normalizes terms and computes n-gram overlap; The *context matcher* builds a set of terms from neighboring elements, and tries to capture matches when neighboring-element sets are similar to each other [3]. Each matcher produces a similarity score that is combined, with weights, into a *total-similarity-score*.

Tightness-of-fit Measurement: Schemr's task, in this phase, diverges from the traditional aim of schema matching: rather than generating mappings between elements, we use the total-similarity-scores to create an overall score that captures the semantic intent of schema search. Our principle here is to measure the tightness-of-fit by minimizing the distance between relevant elements in a result schema. The intuition behind our distance measure is as follows: if two result-schema elements are in the same entity, no penalty. If they are in the same entity neighborhood (transitive closure on foreign key), then a small penalty applies. If they are in

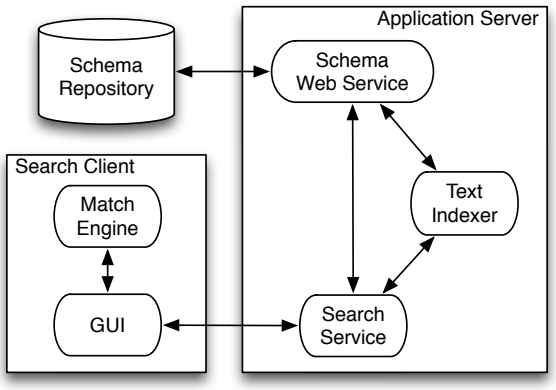

Figure 4: Schemr system architecture diagram

unrelated entities, then a larger penalty applies.

Result-schema elements M, have similarity scores S, and a penalty P. The tightness of fit score is $\sum_M(S \cdot P)$. However, there are many valid configurations of P. We want $P*$ that minimizes the total penalty for a result schema. Thus, $\arg\max_P \sum_M(S \cdot P)$ gives the overall tightness-of-fit score which determines the final search results.

Architecture and Implementation

Schemr's architecture (Figure 4) features a client GUI for entering search terms and reviewing search results. The GUI processes search terms and calls the *Search Service*. On the Schemr server, an offline Lucene [6] *Text Indexer* flattens schemas from the *Schema Repository* to build a document index. Schemr builds on the open-source repository and Search Service create by the Galaxy project [2]. Online, the index filters for candidate schemas, and sends results to the client for matching and ranking in the *Match Engine*. The GUI visualizes the results with encoded quality cues.

Schemr is implemented as an Eclipse Rich Client Platform (RCP) application, and will integrate with the Eclipse Data Tools Platform(DTP) [7] project. Schemr is available as open-source software [8].

Visualizations

Schemr visualizes result schemas in an interactive GUI, supporting operations like panning, zooming, auto-layout, and drilling-in. Schemr features two views (Figure 2): the multiple-results view is a matrix of n results, constituting a *result page*. For each result page, Schemr runs the Schema Matching and Tightness-of-fit algorithms on demand. The schema-visualization view details a particular schema: its similarity matrix is encoded as color and size, in a force-directed layout. To ensure Schemr scales to extremely large schemas, we plan to employ schema-mapping visualization techniques [4], including schema summarization [5].

Applications

Schemr is a part of an open-source information integration framework, OpenII [9]. As a module of OpenII, other framework components enable new schema search applications and scenarios, magnifying Schemr's benefit. For example, integrating Schemr with schema import and export functionality gives users motivation to build metadata repositories. As well, integrating Schemr's search functionality with a codebook that contains data types like units, date/time, and geographic location, would encourage a deeper standardization of data types alongside schema search results. With an OpenII community of users searching the repository, collaboration functionality that provides usage statistics and comments on schemas, would improve schema search results. Finally, integrating Schemr with a schema editor would allow for a new model development process, in which search results are iteratively used to augment a schema. In this process, we can also capture implicit semantic mappings between schema elements, information on schema re-use, and the provenance of new schema entities.

3. DEMONSTRATION SCENARIO

We will demonstrate Schemr's search capabilities over a repository of both relational and semi-structured schemas, small and large, spanning many domains. The data set is a significant collection of public schemas.

Users are able to construct searches that involve keywords, schema fragments or both. They can also choose from provided examples. Optionally, they may choose to see intermediate results, such as the query-graph, candidate schemas, raw similarity scores, and tightness-of-fit scores. In the Schemr GUI, users may explore results in an interactive visualization. First, results will be rendered as a matrix of many small schema trees. Next, the user may drill-in and interactively explore the quality of a single schema result.

4. SUMMARY

Schemr demonstrates an effective approach to schema search and visualization. It uses a novel combination of document based filtering, schema matching, and semantics- and structure-aware scoring. Schemr will be deployed as a standalone tool for organizations to search and share schemas, facilitating the schema design process and paving the way for information integration. As well, Schemr will be a module of the OpenII framework, and improve accessibility and benefit of many information integration applications.

Acknowledgments

We would like to thank Kristin Barker, Harr Chen, Tyson Condie, Joe Hellerstein, Neal Lesh, Peter Mork, Tapan Parikh, Arnie Rosenthal, Len Seligman, Sanjay Unni and Chris Wolf.

5. REFERENCES

[1] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3), 2003.

[2] P. Mork, L. Seligman, M. Morse, A. Rosenthal, C. Wolf, and J. Hoyt. Galaxy: Encouraging data sharing among sources with schema variants. In *To appear in: ICDE*, 2009.

[3] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4), 2001.

[4] G. G. Robertson, M. P. Czerwinski, and J. E. Churchill. Visualization of mappings between schemas. In *CHI*, 2005.

[5] C. Yu and H. V. Jagadish. Schema summarization. In *VLDB*, 2006.

[6] Lucene. `http://lucene.apache.org`.

[7] Eclipse data tools platform (dtp) project. `http://www.eclipse.org/datatools`.

[8] OpenII Google Code Project. `http://code.google.com/p/openii`.

[9] OpenII Project. `http://sites.google.com/site/openinformationintegration`.