Falcons Explorer: Tabular and Relational End-user Programming for the Web of Data

Gong Cheng[1,2], Huiyao Wu[1,2], Saisai Gong[1,2], Weiyi Ge[1,2], and Yuzhong Qu[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] School of Computer Science and Engineering, Southeast University, China
{gcheng,huiyaowu,ssgong,wyge}@seu.edu.cn, yzqu@nju.edu.cn

Abstract. This paper describes an online interface named Falcons Explorer that provides both a tabular perspective and a relational perspective for end users to implicitly program their information needs in a fully guided process of searching and browsing the Web of data.

Keywords: Browsing, end-user programming, relation, search, table

1 Introduction

Triggered by the concept of Linked Data, substantial graph-structured data described by RDF and ontologies has emerged on the Web, and is interlinked, turning into a Web of data. It activates researches on supporting end users to search and browse such graph-structured Web data having well-defined meanings, e.g. to provide end users with a handy service that is able to find "biochemists who won the Nobel Prize for Chemistry and was supervised by a compatriot". Intuitively, to fully utilize the explicitly described semantics of data for retrieving such precise information, one should analogously formalize an information need in a precise manner, i.e. as a graph-structured query such as a SPARQL query. However, this puts end users into great inconvenience because, on the one hand, they are not obliged to have knowledge of graph-structured representation of information, and on the other hand, they are usually not aware of the schema (ontology) as well as identifiers of data items (URIs) used by an application.

Actually, the problem per se is in accordance with the concept of *end-user programming* [3], which is defined as programming (here constructing a graph-structured query) to achieve the result of a program (here graph-structured information) primarily for personal use (here the present end user). End users lack professional skills at programming, and thus require specialized programming environments. To this end, previous researches have proposed solutions of various paradigms, e.g. supporting semi-automatical construction of graph-structured queries by incrementally recommending candidates [5].

Distinguished from existing work, we argue that queries over graph-structured data can be visually shaped in a way different from graphs that (a) is familiar to end users (better than graphs), (b) allows precise description of information needs (better than keyword queries), and (c) has considerable expressive power

(better than plain faceted search). All these requirements together point to *tables*. Table, which widely exists in spreadsheets and relational databases, is surely a popular structure to organize information for ordinary people because it was reported that over 60% of end-user workers use spreadsheets or databases, and approximately half of spreadsheet users use conditional statements (e.g. IF) or formulas [4]. That is, people generally not only deal with tables in their daily work, but also have certain ability to program on tables. In the remainder of this paper, we will show how our system, named *Falcons Explorer*, guides users to smoothly construct complex structured queries by using the devised *tabular perspective* for ontological RDF data. Further, a *relational perspective* is provided for advanced users who have knowledge of relational database to perform more complicated tasks.

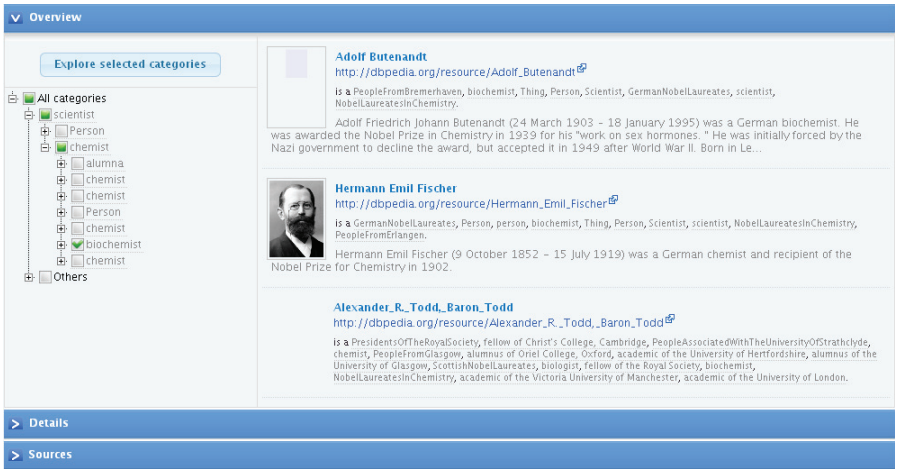

Fig. 1. Browse an entity collection. An overview of the entity collection is presented on the right side. A category hierarchy is provided on the left side for filtering.

2 Standing on the Shoulders of Falcons

Falcons Explorer[3] (Explorer for short) is an online interface built on Falcons Object Search[4] (Falcons for short), an entity search engine for the Web of data. Grounded on the search service and an API for data retrieval provided by Falcons, Explorer implements a powerful environment for end users to search and browse, or in other words, to program on a subset of the Web of data comprising

[3] `http://ws.nju.edu.cn/explorer/`.
[4] `http://ws.nju.edu.cn/falcons/objectsearch/`.

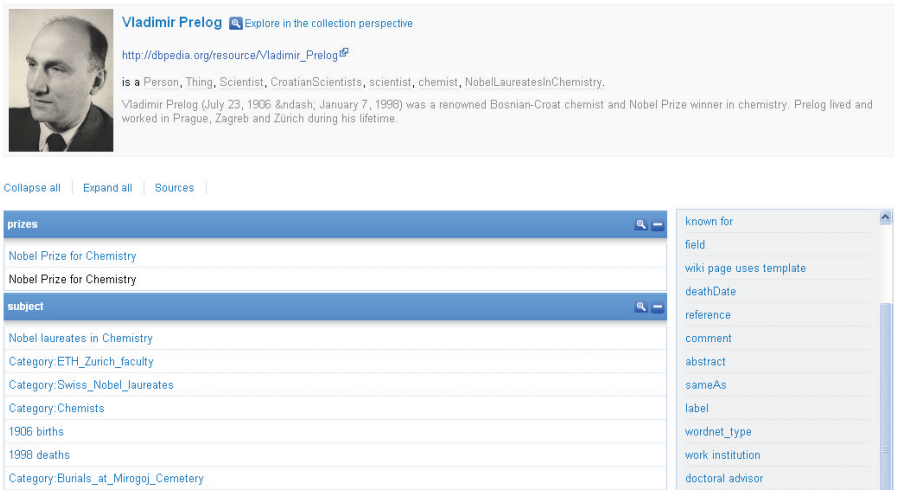

Fig. 2. Browse a single entity. On the left side, related RDF triples are grouped by their properties, an index of which is provided on the right side.

tens of millions of data sources (RDF documents) and billions of RDF triples crawled by Falcons.

The main features of Falcons, namely keyword-based entity search and query refinement with category (class) hierarchies [1] are embedded in Explorer. A typical user of Explorer starts interaction by submitting a keyword query that indicates the name of an entity. Immediately, a collection of entities matched with the keyword query are returned, as shown by Fig. 1. For each entity in the collection, its label, URI, categories, comment, and image (if any) are provided as a summary of its description. If the user finds the search results contain too many irrelevant entities, she will use the provided category hierarchy for filtering, on which multiple selections are allowed, i.e. to keep only the entities belonging to at least one of the selected categories.

Once the user finds the entity being searched for, she will directly click on this entity and then will be forwarded to a new page that serves the detailed description of this entity, as shown by Fig. 2. In this perspective, all the RDF triples describing the present entity are grouped by their properties. An index is also provided for fast lookups. Now the user has three kinds of further navigation. Firstly, she can click on the "Explore in the collection perspective" button, which will trivially treat the present entity as a singleton entity collection to be shown in a page similar to Fig. 1. Secondly, she can click on any entity that is a value of any property, to open a new page similar to Fig. 2 but with the selected entity as the present one. Thirdly, she can explore all the values of any property as a collection in a new page similar to Fig. 1.

It is worth noting that the data presented in both the single-entity and the entity-collection perspectives is derived from the real Web of data, usually

multiple data sources. The user can browse a list of these sources by clicking on the "Sources" button in both Fig. 1 and 2.

3 Tabular End-user Programming

For browsing an entity collection, Explorer provides a new powerful tabular perspective, as shown by Fig. 3. Basically it contains a table that organizes the properties (columns) of each entity in the collection (rows). Because multi-valued properties are supported, each cell may contain more than one value. Nevertheless, this table is not static but is online programmable. In this section, we will show how users can easily customize the data to be displayed in this table. In this sense, table exhibits its superiority in visualization because all the changes in both the schema and the data are immediately reflected.

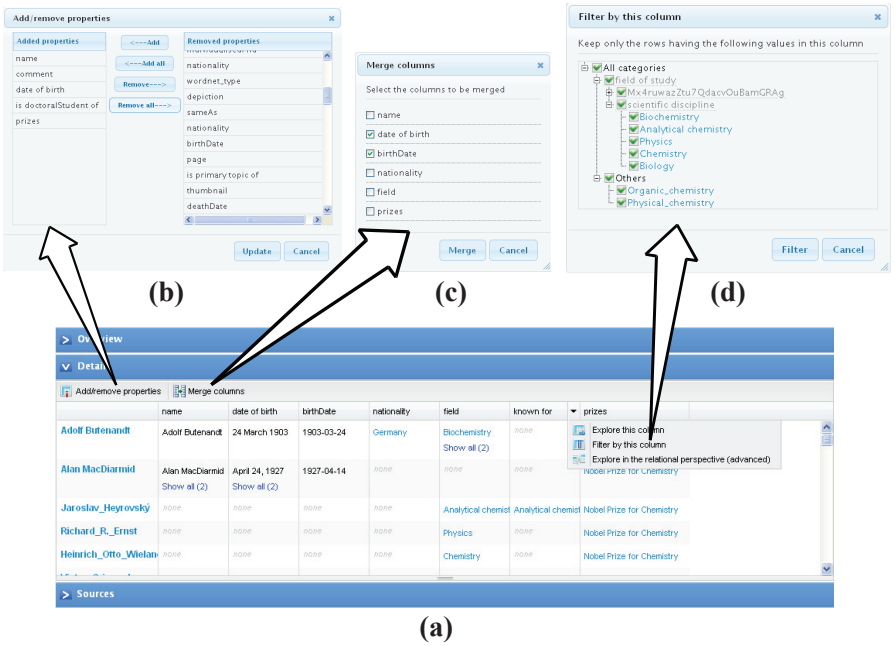

Fig. 3. Details of an entity collection. (a) Related RDF triples are organized into a table, where each row stands for an entity, each column stands for a property, and the cells hold property values. (b) Add or remove properties from the table. (c) Merge columns. (d) Use property values as well as their categories for filtering.

3.1 Programming the Schema

Columns (properties) of this table are programmable.

By default, the system arbitrarily selects several properties for display. By clicking on the "Add/remove properties" button, the user will open a dialog box where she can add more properties applicable to the present entity collection to the table or remove the properties she is not interested in from the table, as shown by Fig. 3(b).

She may further find that it would be better to merge some columns, i.e. not to distinguish between several properties. For example, to obtain the friends of people, one may want to read the values of properties like "knows" and "works with" in a single column. Therefore, she can click on the "Merge columns" button to open a dialog box, as shown by Fig. 3(c), to merge columns. Such operations are completely reversible, i.e. being able to split any column that is formed by a merging operation. In addition, recall that the data presented here often comes from multiple data sources, merging becomes particularly useful in such heterogeneous environment because the user can merge two or more properties (possibly defined by different ontologies) that are believed to be semantically equivalent, e.g. "date of birth" and "birth date".

3.2 Programming the Data

Data (property values) of this table is programmable.

The user can filter the present entity collection by setting restrictions on their properties. As shown by Fig. 3(d), for each column the user can open a dialog box that lists all the values present in the column. She can on the one hand select individual values (entities or literals, e.g. an individual scientist) as restrictions or on the other hand leverage a category hierarchy of these values to impose more general restrictions (e.g. all biochemists).

3.3 Shifting the Focus

End users usually have indefinite requirements in their minds so that they are often likely to change their focus from time to time. For example, the user may start with the keyword query "Nobel Prize" but soon realize that she is actually looking for Nobel laureates. In this case, Explorer provides two ways to let the user shift her focus. On the one hand, she can directly start a new search with a different keyword query, e.g. "Nobel laureate". This will, when still keeping the previous entity collection on "Nobel Prize", open a new tab on the new query. On the other hand, she can locate the "is awarded to" property on the "Nobel Prize" table, and click on the "Explore this column" button. The system will accordingly take all the entities present in this column as an entity collection to be shown in a new tab. Such action can also be considered as a kind of navigation from an entity collection to another by following one or more (in the case of merging) properties. Surely at any time, the user can always switch her focus over to any other tabs to continue programming.

4 Relational End-user Programming

Whereas the tabular perspective has exhibited considerable expressive power, skilled end users may expect more powerful functions to meet more complex information needs. To this end, Explorer provides a relational perspective, as shown by Fig. 4. Basically, skilled end users are supposed to have knowledge of relational database in preference to graphs and RDF. Therefore, by regarding RDF data as relations (sets of tuples), several key operations on relations such as projection, join, and tie [2] are implemented as fully guided tasks.

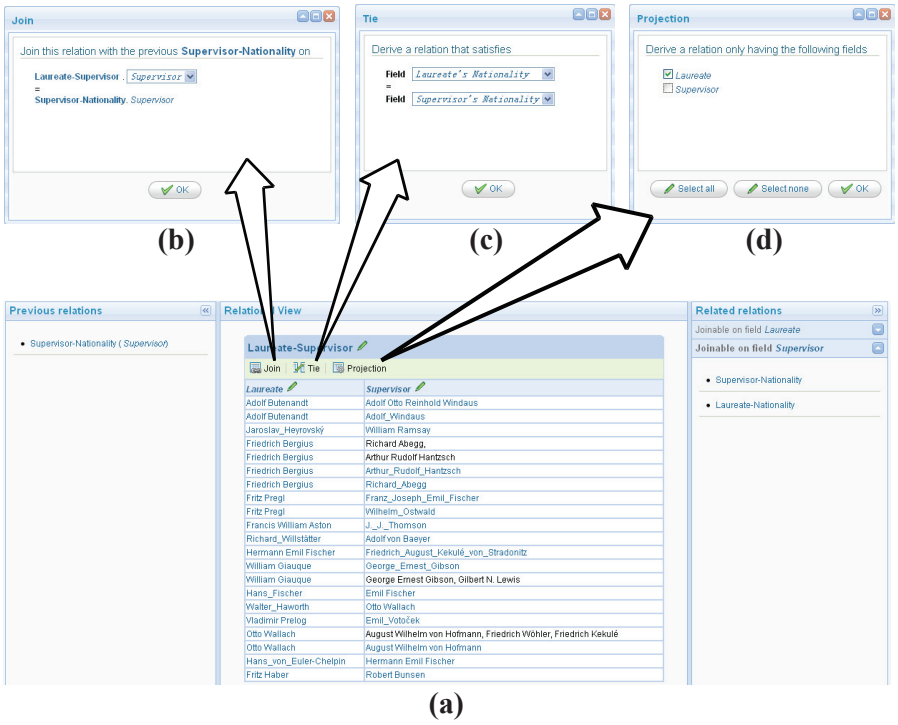

Fig. 4. The relational perspective. (a) The present relation. (b) Join of the present relation with the previous relation on the selected fields. (c) Tie of the present relation on the selected fields. (d) Projection of the present relation on the selected fields.

Initially, for each column in the tabular perspective, the user can click on the "Explore in the relational perspective (advanced)" button to form a binary relation between the present entity collection and their values in this column as the present relation. All the relations and all their fields (domains) can be arbitrarily renamed by the user.

A *projection* of a relation is to derive a new relation by selecting certain fields from it and then removing duplicate tuples. The user can easily achieve this by

clicking on the "Projection" button on the present relation. This opens a dialog box to select interesting fields, as shown by Fig. 4(d).

A *tie* of an n-ary relation is to derive a new $(n-1)$-ary relation from it by firstly filtering the original relation to keep only the tuples whose elements from two specified fields are equivalent, and then performing a projection of the filtered relation to exclude either of the two specified fields to derive an $(n-1)$-ary relation as the result. The user can easily achieve this by clicking on the "Tie" button on the present relation to open a dialog box to select two interesting fields, as shown by Fig. 4(c). We will later illustrate that the tie operation is essential to cyclic joins.

A *(natural) join* of an n-ary relation with an m-ary relation is to derive a new $(n+m-1)$-ary relation by firstly computing the Cartesian product of the two relations, and then performing a tie of the resulting $(n+m)$-ary relation on two specified fields that originate from different relations. To achieve this, in the relational perspective as shown by Fig. 4(a), all the relations that are joinable on each field of the present relation are listed on the right side. By selecting one of them, the current relation will become a previous relation shown on the left side and the selected relation will become the current one. Then the user can join them by clicking on the "Join" button on the present relation to open a dialog box to select a field from all the joinable ones, as shown by Fig. 4(a).

5 Case Study

Now we illustrate how to use Explorer for meeting the complex information need mentioned at the beginning of the paper.

The user starts with a keyword query "Nobel Prize for Chemistry" and finds the entity in question from the results. In its single-entity perspective, she explores all the values of its "is prizes of" property as an entity collection indicating Nobel laureates in Chemistry. In the resulting entity-collection perspective, she firstly leverages the category hierarchy to filter biochemists, and then in the tabular perspective she adds several properties describing the "is supervised by" relationship and several describing "nationality", and merge the two groups of columns individually. The two derived columns are kept in the relational perspective for later use, and are named L-S and L-N individually. Returning to the table, all the entities in the "is supervised by" column are explored as an entity collection. In this new table, several "nationality" properties are added and merged, and the derived column is explored in the relational perspective, named S-SN. Now in the relational perspective, firstly L-S and S-SN are joined on S into a new ternary relation named L-S-SN indicating biochemists as Nobel laureates in Chemistry, their supervisors, and the nationalities of their supervisors. Secondly L-N and L-S-SN are joined on L into a 4-ary relation named L-N-S-SN that further includes the nationalities of biochemists as Nobel laureates in Chemistry. Finally, a tie of L-N-S-SN on N and SN is performed to indicate that biochemists as Nobel laureates in Chemistry and their supervisors have the same nationality.

6 Lessons Learned and Future Work

Firstly, heterogeneity significantly confuses end users. At the schema level, for example, DBpedia uses at least three "birth date" properties identified by different URIs. The proposed merging operation on columns in the tabular perspective helps to semi-automatically solve this problem. We will try to apply similar solutions to category hierarchies. At the data level, heterogeneity (a.k.a. coreference) further undesirably reduces the number of tuples derived from join operations. This inspires us to consider similarity join in future work.

Secondly, pushing all the data about an entity collection to end users not only is cost-inefficient but also sometimes floods users with much boring information. The proposed adding and removing operations on columns provide users a schema-level way to manipulate the scope of data in accordance with their interests. Data filtering at the level of data source should also be a beneficial function. However, it is inappropriate for users to judge the relevance of data sources solely based on their URIs. Thus we are interested in automatically computing informative and/or indicative summaries for data sources.

Acknowledgments

This work was supported by the NSFC under Grant 60773106. We would like to thank Dr. Wei Hu for his suggestions on this work.

References

1. Cheng, G., Qu, Y.: Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. Int. J. Sem. Web Inf. Sys. 5(3), 49–70 (2009)
2. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13(6), 377–387 (1970)
3. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M., Wiedenbeck, S.: The State of the Art in End-User Software Engineering. ACM Comp. Surv., to appear
4. Scaffidi, C., Shaw, M., Myers, B.: Estimating the Numbers of End Users and End User Programmers. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 207–214. IEEE Computer Society, Washington, DC (2005)
5. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From Keywords to Semantic Queries - Incremental Query Construction on the Semantic Web. J. Web Sem. 7(3), 166–176 (2009)

Appendix

This submission to the Open Track is an end-user application grounded on a Semantic Web search engine indexing billions of RDF triples crawled from the real Semantic Web. It provides tabular and relational Web interfaces for end users to program (search, browse, query, etc.) ontological RDF data.