Granada: Relational Database Navigation and Scaling

Jens Kötters and Peter W. Eklund†

†Centre for Cyber Security Research and Innovation
Deakin University, Geelong, Australia.

Abstract. In this paper an application is presented that allows the construction of conjunctive queries (represented as labeled graphs), the presentation of the results of the queries (as tables), and the selection of options for specialization or generalization of the graph queries. The graph labels correspond to the query vocabulary, which is obtained by conceptual scaling of the database. Each scale provides the specialization and generalization options for a particular kind of graph node. At any time, the user is shown the current query, the result table, and a set of options to modify the query.

Keywords: conjunctive queries, pattern concepts, relational databases, navigation, conceptual scaling

1 Introduction

The Granada tool (hosted at `https://github.com/koetters/dbnav`) allows a user to query a relational data source without prior knowledge of the schema. This is achieved through point-and-click extension of graph-based queries, which makes the tool suitable for data exploration as well. Data analysis is currently not supported by the tool, but some functionality could be integrated later. The graph-based queries provide an abstraction from the data source, but we assume a MySQL database in the following (other backends have not yet been implemented). No assumptions regarding the structure or contents of the database have to be made. Navigation with the tool is described in Section 2. Section 3 describes the tool in the context of Formal Concept Analysis (FCA) and our own work within FCA.

2 User Guide

The tool is realized as a web application, using Python 3 server-side. Supported browsers are current versions of Firefox and Chrome, which support modern standards like HTML5 and Javascript ES6. The Granada package contains a file `app.wsgi`, which is the interface to the web server (cf. `https://www.python.org/dev/peps/pep-0333/`). For simplicity, the file can also be run as a script with `python3 app.wsgi`, which starts the app in Python's built-in web server. The initial application screen (Fig. 1) can then be found at `localhost:8081`. The navigate and edit pane (left and middle in Fig. 1) list all databases known to

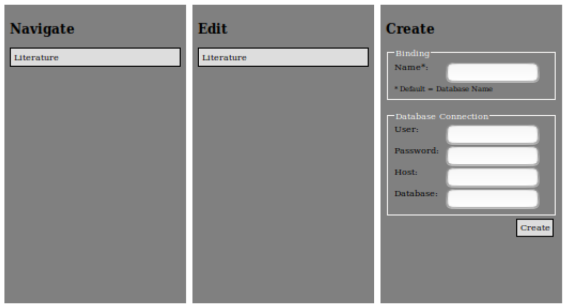
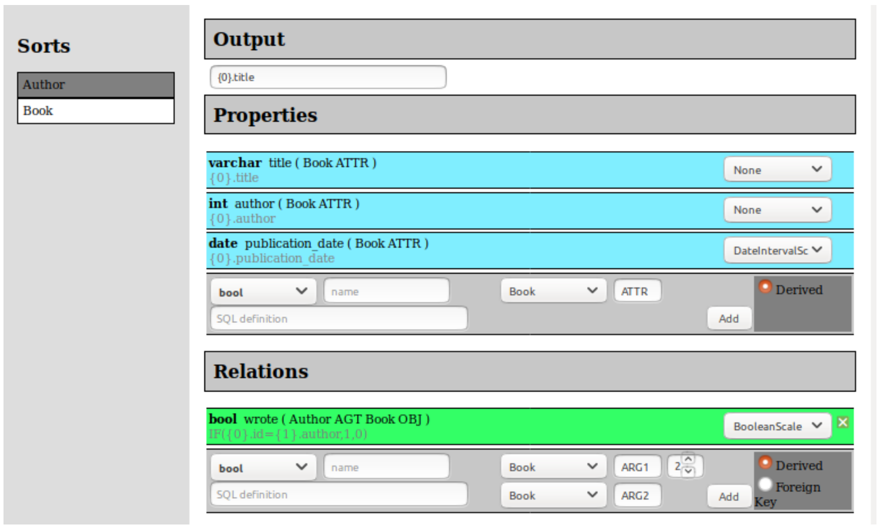


Fig. 1. Granada's Initial Screen

the system (both lists are identical). Initially, the lists will be empty. To include a database in the list, the relevant information (user, password, host and database name) is entered in the create section (right section in Fig. 1). This creates a JSON file on the server with connection information and additional information obtained from the database (e.g. tables and columns). We refer to this file (and the information it contains) as a *binding*; a name for the binding is optionally specified on creation (this defaults to the database name). Once a binding is created, it is listed in both the navigate and edit panes.

Clicking on an item in the edit pane (middle in Fig. 1) displays the associated binding in the edit view (Fig. 2). The tables in the database are listed on the left of Fig. 2, and we refer to them as *sorts*. To the right of the sort list, all binding information related to a particular sort (selected in the list) can be seen. The cyan entries in the properties pane are the table columns, and each is listed with its SQL datatype (similar to how attributes and methods are listed in an API documentation). Green entries in the relations pane represent foreign keys on or into the respective table/sort. Foreign keys are only included in the initial binding if they are explicitly specified in the database. Otherwise, they have to be specified manually, using the form at the bottom of the relations pane. We refer to the entries in either the properties or relations section as *many-valued attributes*. Such attributes can be conceived as unary functions (a foreign key is conceived as a Boolean function on object pairs, i.e. as the characteristic function of the underlying relation). The output pane (top of Figure 2) specifies how a table row is printed as text (by default, the primary key is printed). The



{0}.title

varchar title (Book ATTR)
{0}.title
None

int author (Book ATTR)
{0}.author
None

date publication_date (Book ATTR)
{0}.publication_date
DateIntervalSc

bool wrote (Author AGT Book OBJ)
IF({0}.id={1}.author,1,0)
BooleanScale

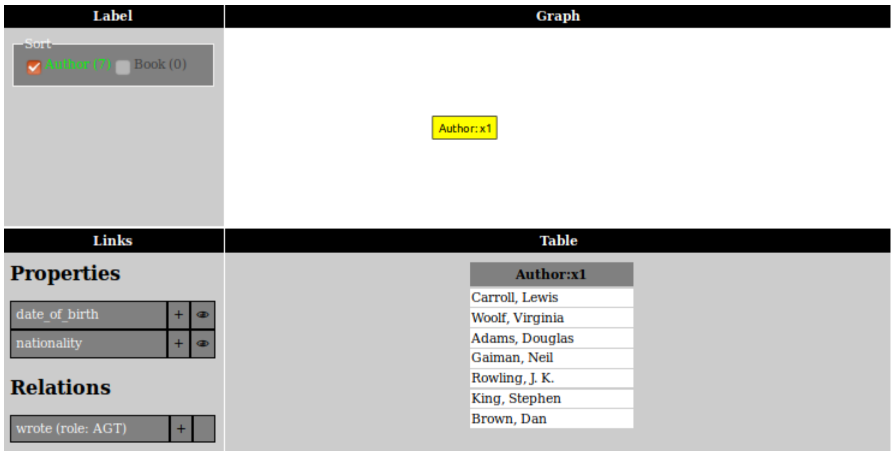
Fig. 2. GRANADA's Binding Editor

drop-down menu on the right-hand side of each attribute is used to configure whether the attribute can be used in queries and, if yes, what search mask is used to specify conditions (in FCA terms, the drop-down menu is used to *scale* the attribute, cf. [4]).

The navigation view (cf. Fig. 3) is opened when the user clicks on a binding in the navigate section in Fig. 1. We assume here that the database has already been configured for navigation, either in the editor view as described above, or automatically by the system, using sensible defaults.

The navigation view is partitioned into four panes, showing the query graph (top right), the result table (lower right) and associated options (left windows). The initial query graph consists of only a single rectangular node, labeled `Any:x1`. It describes the set of all objects in the database (i.e. all rows of all tables). Only in this particular case is an associated result table not shown in the lower right pane, this is because there is no corresponding SQL query for the query graph.

The only action that is initially available to the user is the restriction of `x1` to one of the sorts (i.e. table names) available in the database. The 'Literature' database, which is used in the examples, consists of two relations/tables `Author` and `Book`. These can be selected in the label view (upper left). The numbers next to the sorts represent the number of rows in the respective tables. Selecting one of the sorts (cf. Fig. 3) updates the query graph (the node label now shows the selected sort, see top right), produces a table which lists all objects of that sort (bottom right), and shows all properties and relations associated with the objects of that sort in the link view (bottom left). Unchecking the sort in the label view leads back to the previous query. This allows one to quickly examine

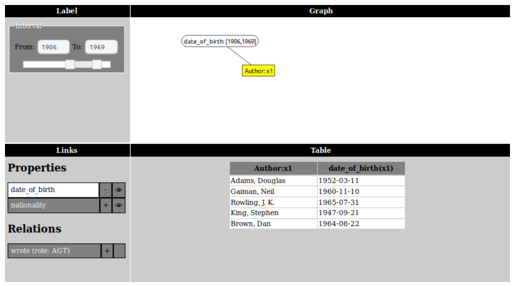


Author:x1
Carroll, Lewis
Woolf, Virginia
Adams, Douglas
Gaiman, Neil
Rowling, J. K.
King, Stephen
Brown, Dan

Fig. 3. Choosing a Sort

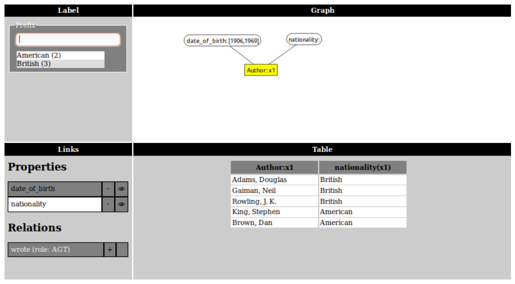
the available sorts, their associated properties and relations. What properties and relations are exactly shown can be specified in the bindings editor (Fig. 2).

The `Author` sort has two properties, `DOB` (date-of-birth) and `nationality`, listed in the link view under properties. The eye icon is an affordance that allows one to view the respective property in the result table (by adding an additional column in each case). Clicking the eye icon again toggles the view on the respective column. This does not modify the graph. Clicking the plus control symbol on the other hand, adds a relation node to the graph (relation nodes are drawn as rounded nodes), which represents a condition on the many-valued attribute. Some initial condition is given, which may or may not restrict the current result set. In the case of the `DOB` attribute, the initial condition specifies that authors are born between 1800 and 2000 (a suitable range is computed which does not restrict the result set). A minus now appears in place of the plus control symbol; clicking on it destroys the relation node (and thus removes the condition). Clicking the button with the property name shows options to modify the condition in the label view (top left), if the property is active (i.e. an associated relation node exists in the graph). For the `DOB` property, the label options are given by a slider with two handles, which allows one to select an interval of permissible values (cf. Fig. 4). The `nationality` attribute is associated with a different kind of scale, which allows the user to specify that the string-valued attribute must begin with a certain prefix. A prefix can be entered in a text field in the label view, or chosen out of a list of occurring values (cf. Fig. 5). The plus symbol on a relation link creates a relation node together with object nodes for the associated objects, with a corresponding sort. For example, clicking on the plus symbol for the `wrote` relation creates the corresponding relation node plus an associated `Book` node (cf. Fig. 6). Clicking on the large button with the relation name on it shows the pairs in the relation that satisfy the query constraints. If a relation between objects has an associated value (an example from [7] is the



Author:x1	date_of_birth(x1)
Adams, Douglas	1952-03-11
Gaiman, Neil	1960-11-10
Rowling, J. K.	1965-07-31
King, Stephen	1947-09-21
Brown, Dan	1964-06-22

Fig. 4. Setting a Property (Interval Scale)



Author:x1	nationality(x1)
Adams, Douglas	British
Gaiman, Neil	British
Rowling, J. K.	British
King, Stephen	American
Brown, Dan	American

Fig. 5. Setting a Property (Prefix Scale)

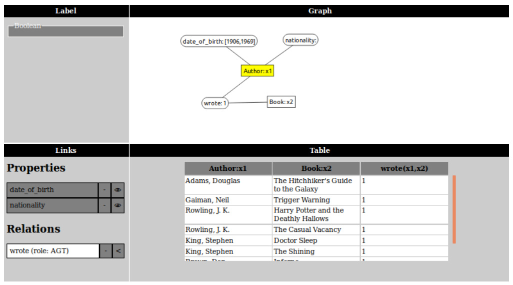
age at which an author wrote book), conditions on that value can be set in the label view. Currently, only three types of scales are implemented: interval scale, prefix scale and foreign key scale. Figures 4, 5 and 6 show examples of their application. The application is still evolving, and a future version will allow to provide implementations of further scales as modular extensions.

3 Related Work

Granada is based on a navigation principle previously described in [5]. In that paper, concepts were not yet formalized, because a suitable notion of extents was still missing. The theoretical foundation was presented in [6], with extents being described by tables (as in relational algebra). Such tables describe relations between objects, and are best thought of as containing only primary keys; the value columns in Granada's table view represent additional information which is not (and does not have to be) covered by the concept model in [6]. Technically, a navigation state (consisting of a query graph and its result table) is a semi-concept, rather than a concept, because Granada does not compute the query closure (i.e. the intent). Bringing concepts back in is a matter of computing the query closure in each navigation state. Providing the user with such information can be considered an additional feature that might be implemented in a future version of Granada. Basic ideas with regard to conceptual scaling, on which

Granada's editor is based, are described in [7]. A preliminary version of the application has been presented at Int. Conf. on Formal Concept Analysis 2017.

Sparklis [3] is a similar tool for interactive query building which also uses natural language features. Relational data exploration has also been considered in the context of Relational Concept Analysis (RCA), see e.g. [1]. An early contribution to querying relational data in the context of FCA, based on the same theoretical notions as the present work, is a relational Toscana extension described in [2].



Author:x1	Book:x2	wrote(x1,x2)
Adams, Douglas	The Hitchhiker's Guide to the Galaxy	1
Gaiman, Neil	Trigger Warning	1
Rowling, J. K.	Harry Potter and the Deathly Hallows	1
Rowling, J. K.	The Casual Vacancy	1
King, Stephen	Doctor Sleep	1
King, Stephen	The Shining	1

Fig. 6. Selecting a relation

References

1. Dolques, X., Le Ber, F., Huchard, M., Nebut, C.: Relational concept analysis for relational data exploration. In: Guillet, F., Pinaud, B., Venturini, G., Zighed, D. (eds.) Advances in Knowledge Discovery and Management. Studies in Computational Intelligence, vol. 615, pp. 57–77. Springer, Cham (2016)
2. Eklund, P., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of toscana. In: Ganter, B., Mineau, G.W. (eds.) Conceptual Structures: Logical, Linguistic, and Computational Issues. pp. 453–467. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
3. Ferré, S.: Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. Semantic Web 8(3), 405–418 (2017)
4. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer, Berlin (1999)
5. Kötters, J.: Object configuration browsing in relational databases. In: Valtchev, P., Jäschke, R. (eds.) Proceedings of ICFCA 2011. LNCS, vol. 6628, pp. 151–166. Springer (2011)
6. Kötters, J.: Concept lattices of a relational structure. In: Pfeiffer, H.D., Ignatov, D.I., Poelmans, J., Gadiraju, N. (eds.) Proceedings of ICCS 2013. LNCS, vol. 7735, pp. 301–310. Springer (2013)
7. Kötters, J., Eklund, P.W.: The theory and practice of coupling formal concept analysis to relational databases. In: Kuznetsov, S.O., Napoli, A., Rudolph, S. (eds.) Proceedings of FCA4AI 2018. CEUR Workshop Proceedings, vol. 2149, pp. 69–80. CEUR-WS.org (2018)