DOCUMENT RESUME

ED 448 742 IR 020 505

AUTHOR Verhoeven, B.; Duval, E.; Olivie, H.
TITLE A Generic Metadata Query Tool.
PUB DATE 1999-10-00
NOTE 7p.; In: WebNet 99 World Conference on the WWW and Internet Proceedings (Honolulu, Hawaii, October 24-30, 1999); see IR 020 454. Some figures contain very small and illegible font.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Interfaces; *Computer Software Development; Computer System Design; Databases; *Information Retrieval; User Needs (Information)
IDENTIFIERS Graphical User Interfaces; *Metadata; *Query Processing

ABSTRACT

This paper discusses a generic query tool that enables an end user to query a metadata store through filters that impose search criteria on attributes. The Metadata Query Tool (MQT) is generic in the sense that it dynamically creates its user interface, based on configuration files that define the metadata scheme and the query functionalities. Although, in principle, the tool can be used to query any relational database, it has been specifically developed to query educational metadata stored in the ARIADNE Knowledge Pool System. The first section of the paper is an introduction, followed by a section that describes the approach to filter-based queries. The third section explains the configurable aspect of MQT, both with respect to the metadata scheme, as well as with respect to the graphical user interface. Some details on the queries generated by MQT are presented in the fourth section. The fifth section covers some user interface aspects. The sixth section gives an overview of related research and tools. The current status is presented in the seventh section. (Contains 11 references.) (MES)

A Generic Metadata Query Tool

B. Verhoeven, E. Duval[1], H. Olivié
Dept. Computer Science
Katholieke Universiteit Leuven (B)
Celestijnenlaan 200A, B-3001 Heverlee. Belgium
E-mail:{Bart.Verhoeven,Erik.Duval,Henk.Olivie}@cs.kuleuven.ac.be

Abstract: This paper discusses a generic query tool that enables an end user to query a metadata store through filters that impose search criteria on attributes. The Metadata Query Tool (MQT) is generic in the sense that it dynamically creates its user interface, based on configuration files that define the metadata scheme and the query functionalities. Although the tool can in principle be used to query any (view on a) relational database, we have developed it to query educational metadata, stored in the ARIADNE[2] Knowledge Pool System [Forte, Wentland & Duval 1997].

1. Introduction

Metadata can be described as "data about data". A good example is a library catalog, which contains information (metadata) about publications (data). Advantages of using metadata to describe electronic resources are numerous [Iannella & Waugh 1997] and include the possibility to efficiently search for data.

Creating an intuitive and powerful Web tool for metadata querying is not an easy task, the more so if we want to develop this tool in a generic way, i.e. without 'hard coding' a particular metadata scheme into it. Moreover, we also want the tool to be flexible in the sense that the query functionality itself must be configurable: the user interface should reflect the important metadata fields and provide a user friendly, yet powerful, mechanism to query the metadata instances. The ideal user interface may depend on the (type of) user and his/her interests and therefor should not be fixed given a certain metadata scheme.

This paper describes how we have developed such a tool. The text is structured as follows. The next section describes the approach of filter based queries. Section 3 explains the configurable aspects of MQT, both with respect to the metadata scheme, as well as with respect to the graphical user interface. Some details on the queries generated by MQT are presented in section 4. The next section covers some user interface aspects. Section 6 gives an overview of related research and tools. Before concluding, we briefly present the current status in section 7.

2. Filter Based Metadata Querying

The metadata we want to query is stored in the ARIADNE Knowledge Pool System [Forte, Wentland & Duval 1997], which basically is a distributed RDBMS. As we wanted to develop a user friendly query tool, we had to develop a reasonably intuitive mechanism, rather than asking the end user to enter SQL-queries. The approach we adopted in MQT is filter based querying (also called content-based filtering) [Mukherjea & Foley 1995]: users define filters on attributes to impose search criteria. A filter on the document title for instance leads to a search for documents, whose title starts with, ends with or contains a search string.

Available filters are defined in one of the configuration files of MQT. Different types of filters are available for different types of attributes, data types, conditions and functions. [Fig. 1] gives a short overview of the different types and their associated functionalities.

Filter type	*Fields*	*Attribute type*	*Conditions*
Basic	Textbox	String	Starts with, *Contains*, Ends with
		Number	<, <=, =, >=, >
		Date	<, <=, =, >=, >
List	Pulldown	NA	NA
Composed	Panel	NA	NA

Figure 1 : Different filters and their associated functionality

Basic filters enable the user to search on (sub-)strings, numbers and dates, based on conventional operators. A list filter contains all applicable values for the associated attribute. The relevant values for such a list filter are retrieved from the database when MQT is launched. Composed filters contain basic and list filters, or other composed filters (see below).

At all times, MQT displays a list of all active filters, which can be deactivated through a simple button associated with each filter. This is illustrated by the simplified screen dump presented in [Fig. 4].

3. Configuration

MQT is developed in a general way: it contains no hard coded references to the metadata scheme, in the form of tables or attributes in the database. This is achieved by a configuration file, which defines the metadata scheme (relations, attributes interrelationships between the relations), as will be explained in section 3.1.

Because not all database attributes are useful in search criteria, the query tool also uses another configuration file, which defines the queries to be supported on the database, as explained in section 3.2.

In this way, MQT can be used to query any metadata store accessible through JDBC (see below).

3.1. Data Scheme Configuration File

This configuration file describes the metadata scheme, in the form of its representation as a relational database scheme. It contains all information needed to generate the queries. More specifically, this configuration file includes the following information:

- Database relations
- For each relation:
  - Attributes
  - Primary key
- References between the database relations

[Fig. 2] illustrates the XML representation of this information.

```
<DATAMODEL>
  <ENTITY>
    <NAME> Pedagogical_Header </NAME>
    <ATTRIBUTE>
      <NAME> Id </NAME>
      <TYPE> STRING </TYPE>
    </ATTRIBUTE>
    <ATTRIBUTE>
      <NAME> Document_Title </NAME>
      <TYPE> STRING </TYPE>
    </ATTRIBUTE>
    <ATTRIBUTE>
      <NAME> Package_Size </NAME>
      <TYPE> INTEGER </TYPE>
    </ATTRIBUTE>
    <KEY> Id </KEY>
  </ENTITY>

  <ENTITY>
    <NAME> User_Comments </NAME>
    <ATTRIBUTE>
      <NAME> Comment_Id </NAME>
      <TYPE> INTEGER </TYPE>
    </ATTRIBUTE>
    <ATTRIBUTE>
      <NAME> Header_Id </NAME>
      <TYPE> STRING </TYPE>
    </ATTRIBUTE>
    <ATTRIBUTE>
      <NAME> Author </NAME>
      <TYPE> STRING </TYPE>
      <MULTIVALUED> YES </MULTIVALUED>
    </ATTRIBUTE>
    <ATTRIBUTE>
      <NAME> Comment </NAME>
      <TYPE> STRING </TYPE>
      <MULTIVALUED> YES </MULTIVALUED>
    </ATTRIBUTE>
    <KEY> Comment_Id </KEY>
  </ENTITY>

  …

  <MAINENTITY> Pedagogical_Header </MAINENTITY>

  <MAINATTRIBUTE> Document_Title </MAINATTRIBUTE>

  <REFERENCES>
    <REFERENCE>
      <FROM>
        <ENTITY> User_Comments </ENTITY>
        <ATTRIBUTE> Header_Id </ATTRIBUTE>
      </FROM>
      <TO>
        <ENTITY> Pedagogical_Header </ENTITY>
        <ATTRIBUTE> Header_Id </ATTRIBUTE>
      </TO>
    </REFERENCE>
    …
  </REFERENCES>

</DATAMODEL>
```

Figure 2: Possible data scheme configuration file

The data model of [Fig. 2] first defines a 'pedagogical header', ARIADNE jargon for an educational metadata instance [Forte, Wentland & Duval 1997]. The ARIADNE approach for semantic interoperability in educational metadata is presented in [Forte et al. 1999].

Entities in the configuration file correspond with relations in the database. In the extremely simplified view of [Fig. 2], such an instance contains an identifier (`Id`), a title (`Document_Title`) and the size of the document being described (`Package_Size`). The key attribute of this entity is the identifier. The second entity defines user comments, which consist of an identifier for the comment (`Comment_Id`), the identifier of the

pedagogical header the comment refers to (Header_Id), the author (Author) and the content of the comment (Comment). The latter two attributes are multivalued. The <REFERENCES> section defines the relationship between the user comment and the pedagogical header it refers to.

Besides this kind of information, the configuration file also defines a main relation and a main attribute in this relation. Both of these are used to build the user interface (see below). The main relation is also used to generate queries, which retrieve a result list (list of metadata instances satisfying the search criteria) or a specific metadata instance from the metadata store.

3.1.1. GUI Configuration File

This configuration file describes the attributes the user interface should provide filters for. Filters are logically grouped, so that a well-structured user interface can be constructed. An example GUI configuration file is presented in [Fig. 3].

```
<GUIDATAMODEL>
  <FILTERCATEGORY>
    <DESCRIPTION> General information </DESCRIPTION>
    <TOOLTIP> General information of the document </TOOLTIP>
    <FILTER>
      <NAME> Document title </NAME>
      <ENTITY> Pedagogical_Header </ENTITY>
      <ATTRIBUTE> Document_Title </ATTRIBUTE>
    </FILTER>
    <FILTER>
      <NAME> Language </NAME>
      <ENTITY> Element_Language </ENTITY>
      <ATTRIBUTE> Language_Name </ATTRIBUTE>
      <TYPE> LIST </TYPE>
    </FILTER>
    <FILTER>
      <NAME> User comments </NAME>
      <TYPE> COMPOSITE </TYPE>
      <FILTER>
        <NAME> User comment by </NAME>
        <ENTITY> User_Comments </ENTITY>
        <ATTRIBUTE> Author </ATTRIBUTE>
      </FILTER>
      <FILTER>
        <NAME> User comment </NAME>
        <ENTITY> User_Comments </ENTITY>
        <ATTRIBUTE> Comment </ATTRIBUTE>
      </FILTER>
    </FILTER>
  </FILTERCATEGORY><FILTERCATEGORY>
    <DESCRIPTION> Technical Information </DESCRIPTION>
    <TOOLTIP> Technical information of the document </TOOLTIP>
    <FILTER>
      <NAME> Package Size </NAME>
      <ENTITY> Pedagogical_Header </ENTITY>
      <ATTRIBUTE> Package_Size </ATTRIBUTE>
    </FILTER>
  </FILTERCATEGORY>
</GUIDATAMODEL>
```

Figure 3: Possible GUI configuration file based on the data scheme presented in [Fig. 2]

The simplified configuration file of [Fig. 3] defines two categories of filters, one to query on general information and one for technical characteristics. Both categories are described by tooltips that will be presented to the end user when the mouse is positioned over the category selector (see below).

The general category contains two filters, one to search on document title and one for the language of the document. For each filter, the corresponding attribute in the corresponding entity (i.e. database relation) is defined. Similarly, the filter category for technical data defines one filter, to search on package size.

The general category also contains a composite filter to search on user comments. A composite filter represents a composite metadata field. In this case, the composite filter consists of two sub-filters, to search on the author of a comment and the content of the comment (given by a specific author) respectively.

4. Metadata Queries

The queries generated by MQT can be divided into 2 categories:

1. queries that identify the metadata instances that satisfy the search criteria: the set of these instances is called the result list;
2. queries that retrieve one complete metadata instance from the database.

The query that retrieves the result list is roughly structured as follows:

SELECT <key attributes>,<main attribute>
FROM <main entity>,<relations with active filters on an attribute>
WHERE <conditions imposed by active filter>
AND <join conditions for relations in the from-clause>

The main attribute values are displayed in the result list (see the lower right area in [Fig. 4]). According to the configuration file of [Fig. 2], this would mean that the document title is shown for each document in the result list. The associated key values are used to retrieve the complete metadata instance from the database upon user request.

Retrieving one metadata instance from the database is somewhat more complicated, mainly because the interrelationships between the database relations can be optional and because there can be multiple

interrelationships between the same two relations. The basic approach to solve this problem is to execute separate queries for each relevant relation.

5. User Interface

[Fig. 4] displays a screen dump of the user interface of MQT based on the previously given configuration files. In reality many more categories and filters are present. The query window (upper left, with "Document Query" as title) basically enables the end user to activate or deactivate filters. For each attribute, a text box or pulldown menu can be used to indicate the search value and the operator ("starts with", "=", etc.) to be applied.

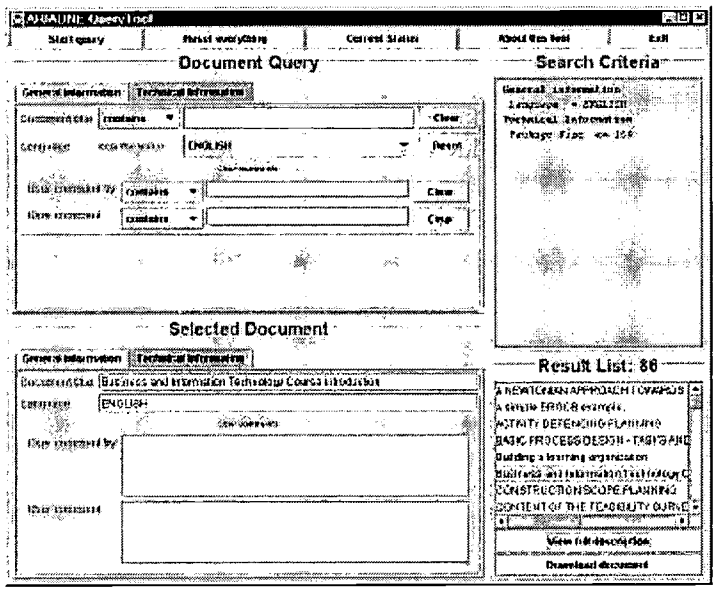

Figure 4: Screen dump of the user interface of MQT given the configuration files of [Fig. 2] & [Fig. 3]

The structure of the query window is identical to that of the data window (lower left, with "Selected Document" title) that displays one particular metadata instance. This was a deliberate design decision, as it makes it more apparent why an instance is included in the result set: the search criteria in the query window can immediately and intuitively be related to the instance values in the data window.

In the upper right area, the active search criteria are displayed in shorthand, so that users have an overview of all active filters at all times. We have added this functionality because of feedback we received on an earlier version of the MQT tool. Without this overview, users quickly lost track of the filters they had applied, and the search values associated with those filters.

Because the number of commands is relatively limited, all commands are available by buttons. Buttons for general commands are situated at the top of the window. These enable an end user to:

- start the query: this means that the query is actually sent to the database, processed, and that the result list is displayed;
- reset everything: all active filters are deactivated and the results list is re-initialized as an empty result list;
- current status: displays the database that MQT is connected to;
- about this tool: provides information about the goal and purpose of the tool;
- exit: the MQT application exits.

Those commands that operate on one specific metadata instance are placed under the result list, as users must select such an instance from that list before applying the command. These commands enable an end user to:

- view the complete metadata instance;
- download the actual document described by the metadata: this command prompts for a user identifier and a password, as only metadata are publicly accessible, but the actual documents are restricted to members of the ARIADNE user group <http://ariadne.unil.ch> (if they are free).

The GUI configuration file (as given in [Fig. 3]) defines the exact layout and configuration of the different graphical components. A filter category in the configuration file corresponds with a Tab panel in the query and

the data window. A filter in the configuration file corresponds with a graphical component in the query window that enables the user to activate/deactivate the filter, as well as with a graphical component that displays the value of a metadata instance in the data window. The type of the graphical component depends on the filter type and the type of the corresponding attribute. The filter type defines the layout and the attribute type determines the available operators.

A composite filter in the GUI configuration file results in a bordered graphical component, which contains the graphical components of the participating filters. The main entity and main attribute are used to indicate the metadata instances that satisfy the search criteria in the result list.

6. Related Research and Tools

Related research can be divided in three different categories: metadata query tools, other query tools (subdivided into XML and web query tools) and general metadata tools.

1. The field of metadata querying is still in full development and not many tools are available. All of the existing metadata query tools we could find either operated on a fixed metadata scheme (Dublin Core, ...) or used some "external" protocol for communicating with the data store (e.g. QUANTUM [Hindall & Haines 1997] uses Z39.50 for querying the database(s)). MQT has no fixed metadata scheme, but it does use JDBC (and SQL) for communicating with the data store.

2a. Since the data scheme configuration file (see [Fig. 2]) can easily be transformed into an XML-DTD, MQT is closely related to XML query tools. XML querying recently became very popular and the ongoing research looks very promising, but it's still in its infancy. One example of such a tool is Innerview <http://www.t2000-usa.com/>, but there still is no uniform, standardised query protocol and mechanism. This might change quickly as some proposals already exist [XML-QL 1998] [XQL 1998].

2b. One more developed research area concerns web querying. There are a lot of sophisticated web search engines available, but there exists no uniformity amongst different tools and the documents on the web don't comply to a standardised, uniform indexing method so that it's very hard to find what you're looking for. This could be facilitated by the use of metadata, but since this is not standardised for the web, web search engines use keywords indexing (and a fixed user interface).

3. Finally, there are general metadata tools. There is one tool in particular we took as a starting point in our design: Reggie [Reggie 1999] is a metadata insertion tool. It uses an input scheme (which ranges from Dublin Core to your own defined scheme) to describe its metadata set and builds its user interface (and associated functionality) based on the selected scheme. Contrary to MQT, Reggie is used to describe data (that is to insert metadata) and it is developed as an applet.

7. Current Status

During the development of MQT, we were confronted with the classical trade-off between functionality and user friendliness. We tried to keep the user interface as intuitive as possible while still supporting quite powerful search possibilities. The user interface also has its limitations though: the result list displays only one attribute for every hit,...

MQT supports any possible combination of active filters, but its functionality is limited in some ways. One of the main limitations is that only the logical 'and' operator is used in the queries: this means that it is impossible to search for documents, whose title starts with 'Behind the scenes' or 'A closer look at'. On the other hand, usability-engineering research consistently shows that users have great difficulty to use boolean operators in the correct way [Shneiderman 1998].

As mentioned before, MQT has been developed in the first place as a query tool for educational metadata in the ARIADNE project [Forte, Wentland & Duval 1997]. All metadata in ARIADNE are stored in a relational database, mainly because of technical reasons [Cardinaels et al., 1998]. MQT is a Java application that uses JDBC to interact with the database. The decision to use a Java application and not an applet or servlet is based on several factors:

1. Servlets generate HTML. Given the current limitation of HTML browsers, this is not a sound development base for an intuitive and powerful user interface.
2. Because of security restrictions, applets cannot communicate with databases that run on servers different from the one that serves the applet.

Java interfaces have been defined for different kinds of filters (see above), attribute types and conditions, so that it is relatively easy to extend the functionality of MQT by adding a new type of filter, attribute type or condition.

Since early 1999, MQT has been deployed within the ARIADNE community and is being used by several hundreds of persons. Based on feedback received in first user trials, we have added the "Search Criteria" window, as mentioned above. Other feedback indicates that this tool indeed serves its major purpose and does enable end users to zoom in fairly quickly on relevant reusable educational resources.

Future work includes the migration from a 2-tier to a 3-tier architecture and the use of introspection of classes in the middleware layer to enable an automatic configuration of the tool.

8. Conclusion

One of the main requirements for the metadata query tool in the Ariadne project [Forte, Wentland & Duval 1997] was that the metadata scheme should not be hard coded, as we wanted to develop a tool that was flexible enough to query metadata stores with different metadata schemes. Using XML configuration files describing the metadata scheme and the required query functionality provided us with the flexibility we wanted to achieve.

Our query approach relies heavily on refinement, through filter based querying. A user can activate and deactivate filters over attributes. Currently, our tool supports basic filters, list filters and composite filters, but new filter types can easily be incorporated.

The approach discussed in this paper leads to a generic query tool, which can be used to query any kind of metadata in any kind of store that can be accessed through JDBC.

9. References

[Cardinaels et al. 1998] K. Cardinaels, K. Hendrikx, E. Vervaet, E. Duval, H. Olivie, F. Haenni, K. Warkentyne, M. Wentland Forte & E. Forte. A Knowledge Pool System of Reusable Pedagogical Elements. *Proceedings of CALISCE98 - 4th International Conference of Computer-Aided Learning and Instruction in Science and Engineering*, pp. 54-62.

[Forte et al. 1999] E. Forte, F. Haenni, K. Warkentyne, E. Duval, K. Cardinaels, E. Vervaet, K. Hendrikx, M. Wentland Forte & F. Simillion. Semantic and Pedagogic Interoperability Mechanisms in the Ariadne Educational Repository. *Accepted for a SIGMOD Record Special Issue on Semantic Interoperability in Global Information Systems*, 1999.

[Forte, Wentland & Duval 1997a] E. N. Forte, M. H. K. Wentland Forte & E. Duval. The ARIADNE project (partI) - Knowledge Pools for Computer Based & Telematics Supported Classical, Open & Distance Education. *European Journal of Engineering Education*, Vol. 22, No. 1, pp. 61-74.

[Forte, Wentland & Duval 1997b] E. N. Forte, M. H. K. Wentland Forte & E. Duval. The ARIADNE project (partII) - Knowledge Pools for Computer Based & Telematics Supported Classical, Open & Distance Education. *European Journal of Engineering Education*, Vol. 22, No. 2, pp. 153-166.

[Hindall & Haines 1997] M. H. Hindall, M. Haines. QUANTUM: A Query and Analysis Tool For Use With Metadata. *Proceedings of the Second IEEE Metadata Conference*, 1997. http://www.computer.org/conferen/proceed/meta97/papers/mtindall/mtindall.html

[Iannella & Waugh 1997] R. Iannella , A. Waugh. Metadata: Enabling the Internet. *Proceedings of CAUSE97 – "The Information Professions and the Information Professional".* <http://archive.dstc.edu.au/RDU/reports/CAUSE97/>

[Mukherjea & Foley 1995] S. Mukherjea & J. D. Foley. Visualizing the World-Wide Web with the Navigational View. *Electronic Proceedings of the Third International World-Wide Web Conference (WWW '95)*, 1995. <http://www.igd.fhg.de/www/www95/papers/44/mukh/mukh.html>

[XML-QL 1998] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu. XML-QL: A Query Language for XML, Submission to the World Wide Web Consortium, 19 August 1998. <http://www.w3.org/TR/NOTE-xml-ql>

[XQL 1998] J. Robie, J. Lapp, D. Schach. XML Query Language (XQL). <http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html>

[Reggie 1999] Reggie: The Metadata editor. <http://metadata.net/dstc/>

[Shneiderman 1998] Ben Shneiderman. *Designing the User Interface. Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 1998. <http://www.aw.com/DTUI>

Acknowledgments

[1] Erik Duval is supported as a post-doctoral fellow by the National Fund for Scientific Research – Flanders (Belgium).

[2] Ariadne is supported by the Telematics Applications of the European Commission and by the Swiss Federal Office for Education and Science.

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").