

Aggregate Table-driven Querying Via Navigation Ontologies in Distributed Statistical Databases

Yaxin Bi¹, David Bell¹ and Joanne Lamb²

¹School of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, UK

²CES, University of Edinburgh, Holyrood Road, Edinburgh, EH8 8AQ, UK
{y.bi, da.bell}@qub.ac.uk and j.m.lamb@ed.ac.uk

Abstract. In this paper we describe a query paradigm based on ontologies, aggregate table-driven querying and expansion of QBE. It has two novel features: visually specifying aggregate table queries and table layout in a single process, and providing users with an ontology guide in composing complex analysis tasks as queries. We present the role of the fundamental concept of ontology in the context of the content representation of distributed databases with large numbers of multi-valued attributes, and in query formulation and processing. The methods and techniques developed for representing and manipulating ontologies for query formulation and processing make extensive use of XML and DOM. The core functionalities of content representation, query formulation without prior knowledge about databases, statistical summary and result presentation are integrated into a front-end client within the underpinning MVC architecture, which has been implemented in Java and JAXP.

1 Introduction

Our focus in the present study is on relational databases with large numbers of multi-valued attributes which have been common for several years in a variety of applications – especially for those including statistical databases and scientific databases – and often they are distributed on the Internet. The tasks of content representation, statistical summary and result presentation (publishing) place significant demands on front-end interfaces and integrated access to these databases. The large number of multi-valued attributes means that their meaningful representation is essential for understanding the content of databases and for query formulation. Furthermore, because of the requirements of statistical analysis and the need to publish results in various tabular forms, it must be possible for users to bring both query composition and table layout definition together in a single process of query formulation. Novel query paradigms and data integration mechanisms are thus needed.

Our intention here is to address query paradigms and data integration in the context of a particular application – that of official statistics. The work described includes three major aspects. The first is the support of statistical summary queries in terms of macro (aggregate or summary) data table queries for distributed statistical databases with large number of multi-valued attributes. The second is the enhancement of database content representation by means of domain ontologies. The third aspect is a knowledge-based query paradigm, which integrates ontologies, macro table-driven querying and expansion of Query By Example (QBE) [1], allowing users to visually specify macro table queries and table layout in a single process, and providing users with an ontology guide in composing complex analysis tasks as queries.

Much of the work presented here has been implemented in the European fifth framework MISSON project (Multi-Agent Integration of Shared Statistical Information Over the

[inter]Net). MISSION grew out of ADDSIA [2], [3] and early research project MIMAD [4]. The MISSION system is built on a three-tier architecture composed of the client, library and dataserer. The client is a presentation layer – a front-end interface, which is developed using the advanced technologies of MVC (Model-View-Controller) paradigm [5] and DOM (Document Object Model). The library is a mediation layer holding the ontology repository and it acts as a workspace for software agents. It plays a mediation role between the client and the dataserer, that is, by using agents to receive queries from the client, decompose the queries, send the queries to the dataserers and get the query results back to the client. The dataserer is the local site which holds the physical data storage and management tools for data registering and data access.

In this paper, we present concepts, methods, and techniques developed for the system client and data integration in the MISSION system. These concepts and methods have been employed in the system client and the library in order to implement the fundamental functionality of *browsing*, *query construction*, and *publishing*.

2 Motivation

Relational schemas developed for conventional database systems may not suit the representation of the content of very high dimensional data well. In particular, they are inadequate when raw (micro) data is of value – encoding form with spreadsheet formats, which are accompanied with a considerable amount of metadata for interpretation. In a broad sense, schemas are developed for modelling structured data – they have less ability to define the meaning of attributes used in a given domain.

Query interfaces built on schemas may not contain any information about the application domain and how it operates, and in the limited space of a user interface, there is little chance of the characteristics of the entities being explicitly given. As a result, instead of giving users information about the content of data sources, these interfaces require users to fill out lengthy forms to express their information needs. A common assumption for such interfaces is that users are aware of a system implementation model, for example, a relational schema, in terms of structures and data values [6].

Strong interest in enhancing the content representation and presentation of distributed databases to facilitate statistical analysis and query composition emerges in the HCI, database, and knowledge management research communities. Research in these areas has been documented in a number of publications [6], [7], [8]. In [4], the MIMAD (Micro-Macro Data) model was developed. It provides a means of aggregating high dimensional micro data to macro data for statistical summary, as illustrated in Table 1 (a). This work was extended in ADDSIA to take account of domain knowledge – metadata – in order to achieve interoperation and integration between diverse data sources [2], [3]. In MISSION, these ideas are further developed. We make use of domain ontologies built up from metadata to represent the content of distributed databases and incorporate them into macro table query formulation and processing.

The concept of “ontology” is well-known in knowledge engineering and has been applied to data integration [8]. Basically, in this case, the ontology is used to define the meanings of terms used in data sources, and to ensure the consistent use of terminology in order to cope with semantic heterogeneity reflected in the integration of heterogeneous data sources. It can be argued that the ontology is also a good means of representing the content of data sources, because the ontology hierarchy can correspond to the relational schemas and the meanings of terms defined in the ontology can be used to represent and interpret attributes, as we will see in the following sections.

Table 1. An example of macro object (a: left, b: right)

Gender	Employment	Income_SUM
Male	Full-time (F)	2,122,000
Male	Part-time (P)	1,422,000
Female	Full-time	1,922,000
Female	Part-time	1,122,000

		Gender	
		Male	Female
Employment	F	2,122,000	1,922,000
	P	1,422,000	1,122,000

To achieve the best performance in the exploitation of the ontology for query formulation and processing, by expanding on QBE we develop a simple yet powerful query paradigm. It displays meaningful attributes along with ontologies and provides visual selection among them, and it allows users to specify complicated information needs visually in the form of macro object tables, as shown in Table 3 (b). Our approach is based on the conceptual hierarchy of attributes and relationships, which serves a wide range of user expertise and working styles. We adapt implicit Boolean connectives within such a hierarchy based on two operations: *aggregation* and *generalization*.

3 Data model for ontologies

An ontology is defined as a *shared formal conceptualization of a particular domain* [9], which can be used to specify what concepts represent and how they are related. In practice, an ontology can be regarded as a controlled vocabulary providing a concrete specification of term names and term meanings. An ontology can be represented in various ways, such as description logic, Datalog, and frames. To address issues in the context of content representation of databases, query formulation and processing, we employ XML to represent ontologies, which is an effective approach for the hierarchy generation of the ontologies on the fly, required by the system client.

In the MISSION project, we have developed a DTD for representing domain ontologies. It is a further development of our previous work [2]. The DTD offers a uniform framework for representing concepts over heterogeneous distributed databases. Fig. 1 illustrates a fragment of the DTD definition.

In this definition, the FRAME is the root element, and there are four high level elements under it, i.e. TITLE, SUBJECT, DESCRIPTION, and ONTOLOGY. The first three of these represent the conceptual frame, indicating application domains to which ontologies are related. The last element, ONTOLOGY, is broken down into smaller elements: DESCRIPTION, CLASSIFICATION, and VARIABLE, to cover essential information required in the system. In particular, the element VARIABLE is used to define the meanings of concepts which are associated with the attributes within data sets, and constraints indicating the properties of the concepts. The hierarchy reflected in the elements is an important structure because it depicts the explicit hierarchical relationship between the elements.

```

<!ELEMENT FRAME (TITLE?, SUBJECT, DESCRIPTION?, ONTOLOGY+)>
<!ELEMENT ONTOLOGY (DESCRIPTION, CLASSIFICATION, VARIABLE+)>
<!ELEMENT VARIABLE (SET+)>
...

```

Fig. 1. A fragment of the DTD definition for modeling the ontologies ('+': one or more elements, '?': optional)

Note that, although the depth of the hierarchy embedded in the DTD is at only 4 levels, i.e. FRAME → ONTOLOGY → VARIABLE → SET, this is sufficient to represent the ontologies in our case.

Fig. 2 gives a fragment of the ontology of “Catewe” (Comparative Analysis of Transitions from Education to Work in Europe) encoded in XML. In order to access and manipulate ontology information in the system client, we make extensive use of the DOM model, which is increasingly becoming popular for accessing and manipulating XML documents.

```

<FRAME name = "Catewe">
  <TITLE name = "A Comparative Analysis of Transitions from Education to Work in Europe"/>
  <SUBJECT name = "Education"/>
  <ONTOLOGY name = "Catewe">
    <VARIABLE name = "Country" mnemonic="Land" vartype = "geographical">
      <SET value="Scotland"/>
    ...
  </VARIABLE>
  ...
</ONTOLOGY>
</FRAME>

```

Fig. 2. A fragment of XML data

4 Modelling Client Functions

The MISSION client is an MVC application [5]. Fig. 3 presents a high-level block diagram of the client in terms of the MVC architecture. The client consists of three basic components which correspond to the functions of *browsing*, *query construction*, and *publishing*. These accommodate the functionality for the browsing, query construction, and publication of large volumes of statistical information required in the applications of the MISSION system. Each component is modelled to three distinct forms of functionality called *model*, *view* and *controller*, which inherit from the common MVC class and incorporate customizations for individual tasks. For example, the *browsing* component is broken down into three subcomponents: *BrowserModel*, *BrowserModel*, and *BrowserController*. This architecture effectively takes advantage of the MVC design paradigm, ensuring the explicit isolation of the various functions, abstract representation for each function, and interactions of functions with one another.

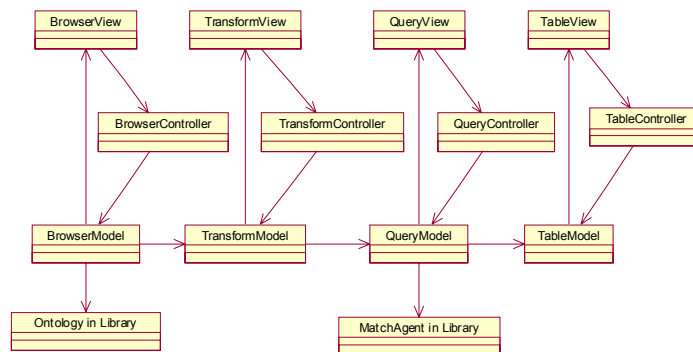


Fig. 3. The MVC and tier partitioning of the MISSION client

5 Visual Query Formulation

In this section, we investigate how concept ontologies are involved in query formulation and processing. Before looking at the process with respect to visual specification of queries, we begin with a brief description of macro table objects.

5.1 Macro table object and table-driven queries

A macro object is defined as follows [4]:

MacroObject: $\{C_1, C_2, \dots, C_n; N_1, N_2, \dots, N_m\}, count, sum, sum-of-squares$

This representation abstractly defines macro table objects, where C_1 to C_n are categorical attributes, N_1 to N_m are numeric attributes, and *count*, *sum*, *sum-of-squares* are common primitive operations to be performed on the numeric attributes. In addition, the operations of *max* and *min* might also be included if applications demand these. In this expression, if the numeric attribute is empty, the *count* as a default operation which will be then applied to categorical attributes.

For queries which produce macro objects as illustrated in Table 1 (a), we need a mechanism to formulate queries in a tabular QBE style. A query table here is defined as an instance of a QueryView class that is managed by the Query Constructor component. A complete table configuration consists of three separate expressions in terms of *horizontal header*, *vertical stub* and *data cells*. The first two of the expressions define the configuration of the x and y axes of the table, partitioning the table into rows and columns. The third expression defines the data cells that hold aggregated values, corresponding to the Cartesian product of the value sets of categorical attributes. The table configuration provides the layout definition, and implicitly restricts attributes to be placed in either the *header* or *stub* only. For example, in formulating the macro object query (see Table 1(a)), the attribute *Gender* is placed on the header and *Employment* is put on the stub, and the cells hold aggregated values produced using the *sum* operation, as illustrated in Table 1 (b).

5.2 The client interface

Fig. 4 presents a screenshot of the user interface, including the three constituents: *browsing*, *query constructor* and *publishing* functions. Notice that the *publishing* function is implicitly specified with the configuration of the query table. These constituents are clearly important for supporting the broader user activities in flexible ways. As seen from the screenshot, the two functions of *browsing* and *query construction* are displayed on two parallel sub-windows. Users can interleave the functions through their views and track their progress. On the top of the query constructor, there exist two combo-boxes, one is to store numerical attributes (users can choose from them to compose queries), and the other presents a choice from the aggregation functions *count*, *sum*, *sum-of-squares*, which will be integrated into macro table queries.

The browsing component is the starting point for system access. It provides users with interactive and incremental functions, allowing users to discover a domain of interest and to interrogate the ontologies to find the fundamental concepts in which they are interested. This function offers a very important means for novices to learn about the data sources through the frames and ontologies. It fulfils the significant demand for the support in the formulation of valid queries without pre-knowledge about the databases. For example, the

user can first find the *frames* that are available in the connected library. Generally knowledge of each frame can be gained through the descriptive information: *title*, *subject* and *description* as described in Fig 2. Once the frame has been chosen, the user can go straight to the concepts held in the ontologies to select the concepts in which he/she is interested. This process does not require the user to have pre-knowledge about the data sources – it can be learned during the browsing session.

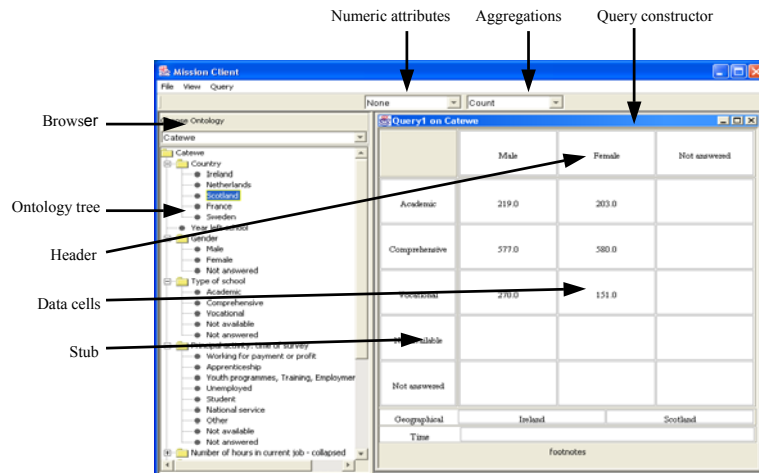


Fig. 4. A screenshot of the system client

The BrowserView presents ontologies as trees. The user can choose one of the ontology trees. For example, the user can choose the local ontologies of Catewe Scotland or the master ontology of Catewe as shown in Fig. 4. The ontology tree is composed of the nodes that are iteratively constructed with the ontology objects, but only the properties of the names and set values are visible. The remaining details within the objects are less useful for the user in understanding the content of databases, and so they only are used in the query process. On the other hand, the view can be switched from one to another, individual nodes and group nodes can be zoomed in to and out from independently without restrictions, according to users' preferences. This makes it possible to allow users to access multiple ontologies of data sources with the same style and avoid the screen being crowded by too many objects. This significantly enhances content representation for distributed data sources.

The idea of the macro table-driven query is to bring query composition and table layout definition together in a single process of query formulation. Defining a layout and style on the one hand and a query on the other hand are usually seen as two different tasks which should be done in two different ways. However, the two tasks are indeed related. It is difficult to define a layout without also defining the related queries. In the system client, the *query construction* component expands the QBE style as a way of query formulation, and incorporates the layout definition into the query formulation process, which acts as the client publishing function. The operator selection on the top field, such as *count*, determines which type of aggregation will be performed for the formulated query, and the configuration of the concepts placed on the QueryView implicitly determines the layout definition of query results. So, query formulation and layout definition are integrated in a single process.

5.3 An Example

Given an information need is to count “different types of school by gender within the countries of Scotland and Ireland”. The user begins with the client interface as shown in Fig. 4. He/she first opens the Browser to connect to a library and obtain a list of ontologies such as *Catewe*, *Catewe Scotland*, etc. which are stored in the combo-box on the BrowserView. For the above request, the user chooses the master ontology of *Catewe*. Then, as illustrated in Fig. 4, he/she starts to navigate the ontology information to locate the three concepts “country, type of school and gender”, respectively. Once these concepts are obtained, the user starts to drag an individual node of *Scotland* and *Ireland* from the ontology tree and drop them on the *Geographical Filed*, to drag *gender* and drop it on the header and place *type of school* on the stub, and then select the *count* operator. When complete, the query can be sent for execution. Simultaneously the visual expression formulated in the query constructor is internally converted into the expression of the table query language.

Looking at the query expression in Fig. 5, this expression explicitly demonstrates how the concepts drawn from the ontology hierarchy are incorporated into query formulation. For example, the concept of Gender, its type, its alias of Sex and a set of values have been included in the expression. Internally the query expression is passed as an object to a host library for query processing. More details for this can be found in [10].

```
<TQUERY>
<COMPUTE operand="table"/>
<WITH operand="generalization">
  <VARIABLE name="Type of school" mnemonic="SCHTYPE" vartype="geographical">
  <VARIABLE name="Gender" mnemonic="Sex" vartype="categorical">
    <SET label="male" value=""/>
  ...
<BROKENDOWNBY operand="aggregation">
  <VARIABLE name="Gender"/>
  <VARIABLE name="Type of school"/>
  ...
```

Fig. 5. The internal expression of the query corresponding to Fig. 4

6 Conclusion

This paper describes major concepts, methods and techniques developed for the system client and data integration in the MISSION project. We describe the use of the fundamental concept of ontology in the content representation of distributed databases, query formulation and processing, and underpinning methods and techniques for the macro objects, and the underlying client architecture of MVC. Although these have been developed specifically for statistical databases in the MISSION project, they can be readily tailored to general applications.

From the technology point of view we seek to use a novel query paradigm to reduce the burden on novices in understanding contents of databases and identifying pertinent attributes. We also present a user-friendly graphical interface to facilitate query formulation by the expansion of QBE style querying and table definition in a single process. For more advanced users, this front-end interface offers a powerful means of composing complex queries and publishing their analysis results, thereby supporting decision making and sharing expertise.

In the current state of the project, an ontology is automatically generated when a dataset and associated metadata are registered to the system. An exception in the frame description which data providers have to type in. With respect to ontology construction, there are three major classes in the metadata – identical, overlapping, and exclusive. The current approaches to coping with these issues are a) if two datasets have identical metadata, then the datasets will share the same ontology, b) if two datasets have different metadata, each dataset will have an individual ontology, and c) if two datasets have metadata with some overlap, then a master ontology has to be created using the two sets of metadata. The last approach involves complicated processing, but the possibility of handling such cases do exist. Some solutions have been proposed, in particular, the idea of mapping the local ontologies to a reference ontology which will be imported from public classification repositories. These solutions will be discussed in the other paper in future.

Acknowledgement

The work is partially supported by the MISSION project (IST 1999-10655) and partially supported by the ICONS project (IST-2001-32429). These are funded by the European Framework V. The authors would like to acknowledge the contributions made by the MISSION client development team.

References

1. Zloof, M.: Query by Example. AFIPS, 44, (1975).
2. Bi, Y., Murtagh, F. and McClean, S.I.: Metadata and XML for Organising and Accessing Multiple Statistical Data Sources, Proceedings of ASC International Conference, Edinburgh, (1999) 393-404.
3. Scotney, B.W., McClean, S.I., Rodgers, M. C.: Optimal and Efficient Integration of Heterogeneous Summary Tables in a Distributed Database. The Journal of Data and Knowledge Engineering, Vol. 29. (1999) 337-350.
4. Sadreddini, M. N. Bell, D. A. and McClean, S. I.: A Model for Integration of Raw Data and Aggregate Views in Heterogeneous Statistical databases. Database Technology, Vol. 4 (2), (1992) 115-127.
5. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994).
6. Tanin, E., Plaisant, C., Shneiderman, B.: Broadening Access to Large Online Databases by Generalizing Query Previews, Proceedings of the Symposium on New Paradigms in Information Visualization and Manipulation, (2000) 80-85.
7. Levy, A. Y., Rajaraman, A., Ordille, J. J.: Querying Heterogeneous Information Sources Using Source Descriptions. Proceedings of the 22nd VLDB Conference, Bombay, India. (1996).
8. Wache, H. V ogele, T. Visser, U. Stuckenschmidt, H. Schuster, G. Neumann, H., Hubner, S.: Ontology-based integration of information – a survey of existing approaches. In Stuckenschmidt, H. (ed.): IJCAI-01 Workshop: Ontologies and Information Sharing, (2001) 108-117.
9. Gruber, T.: A translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Vol. 5(2), (1993) 199-220.
10. McClean, S., Páircéir, R., Scotney, B., Greer, K.: A Negotiation Agent for Distributed Heterogeneous Statistical Databases in SSDBM (2002) 207-217.