*i*MONDRIAN: A Visual Tool To Annotate and Query Scientific Databases

Floris Geerts[1,3], Anastasios Kementsietsidis[1], and Diego Milano[2]

[1] School of Informatics, University of Edinburgh, UK
{fgeerts,akements}@inf.ed.ac.uk
[2] Universitá di Roma "La Sapienza", Italy
diego.milano@dis-uniroma1.it
[3] Hasselt University, Belgium

Abstract. We demonstrate ***i***MONDRIAN, a component of the MONDRIAN annotation management system. Distinguishing features of MONDRIAN are (i) the ability to annotate sets of values (ii) the annotation-aware query algebra. On top of that, ***i***MONDRIAN offers an ***i***ntuitive visual ***i***nterface to annotate and query scientific databases.
In this demonstration, we consider Gene Ontology (GO), a publicly available biological database. Using this database we show (i) the creation of annotations through the visual interface (ii) the ability to visually build complex, annotation-aware, queries (iii) the basic functionality for tracking annotation provenance. Our demonstration also provides a *cheat window* which shows the system internals and how visual queries are translated to annotation-aware algebra queries.

1 Introduction

Modern science relies increasingly on the use of database systems to store huge collections of scientific data. These data are generated from laboratory processes or are copied from other scientific databases. To make sense of these data and decide under which circumstances they can be used, scientists need to know their lineage, i.e., the conditions under which the data were generated, the accuracy of the processes that produced them, or how trust-worthy is the source from which the data were copied. These *metadata* are often stored in scientific databases in the form of annotations. In spite of their importance, existing data formats and schemas are not designed to manage the increasing variety of annotations. Moreover, DBMS's often lack support for storing and querying annotations.

Our work in the MONDRIAN[4] annotation management system [1] is motivated by the pressing needs of biologists, some of which are highlighted by the following example. Consider the relation in Figure 1 which lists triples of identifiers belonging to three distinct biological databases. Each triple associates the identifier *gid* of a gene (in the gene database) with the identifier *pid* of the protein (in the protein database) that the gene produces, where the sequence of the protein is identified by *sid* (in the protein sequence database). Such relations are widely used in the biological domain and offer a

[4] Piet Mondrian: Dutch painter whose paintings mainly consist of color blocks.

quick way to cross-reference and establish associations between independent biological sources [2, 3].

Given such a relation, a biologist often wants to annotate each triple with any evidence that exist and verify its validity. Such evidence might include a reference to an article that mentions that the indicated gene produces the specified protein, or the name of a curator who verified this association. In the figure, we show possible annotations in the form of blocks and block labels. Blocks are used to indicate the set of values for which an annotation exists, while block labels are used to indicate the annotations themselves. In the figure, the annotations indicate the names of curators who verified that a particular association holds. So, in the first tuple, a block indicates Mary's belief that the gene with GDB id *120231* produces protein with id *P21359*. Notice that parts of a triple can be verified by different curators (e.g. see the first tuple), while other parts are yet to be verified (e.g. see the third tuple).

For annotations to be useful, the biologist must be able to query them. For example, she might want tuples that are annotated by either John or Mary. Or, she might want to find which are annotated, and by whom. Often, the lack of annotations is also of interest. For example, a biologist might want the gene-protein (*gid, pid*) pairs that are not annotated, so as to investigate the validity of these yet unverified pairs.

To the best of our knowledge, MONDRIAN is the first system to support the annotation of sets of values, thus allowing for complex annotations such as the ones shown in the figure. Previous works only allowed for annotations to be attached to a particular value of a specific attribute (e.g., see [4]). Single-value annotations are insufficient since they fail to capture the complex relationships of values, relationships which span across attribute boundaries. Another distinguishing feature of MONDRIAN is the ability to query annotations and values alike. MONDRIAN offers an annotation-aware query algebra which we have shown to be both complete (it can express all possible queries over the class of annotated databases) and minimal (all the algebra operators are primitive) [1]. The expressiveness of our algebra goes well beyond the query capabilities of similar systems like, for example, DBNotes [5]. The algebra is simple and intuitive and is able to express all the queries mentioned earlier, and many more (see [1] for the full syntax and examples). For example, query q_1 below retrieves all the tuples that are annotated by either John or Mary, while query q_2 only retrieves tuples that have a gene-protein sequence (*gid, sid*) annotated pair.

$$q_1 = \Sigma_{Mary} \cup \Sigma_{Peter} \qquad q_2 = \Pi^L_{gid,sid}$$

In spite of being simple (and very easy to learn by those familiar with relational algebra), we don't expect that biologists would want to learn yet another query algebra. Instead, it seems natural to offer a visual tool through which a biologists can both annotate data and query them. The objective of this demonstration is to present ***i***MONDRIAN, a tool that offers the above capabilities. Once more, the simplicity of the algebra is to our favor since it facilitates the direct translation of visual queries to algebra queries.

2 The *i*MONDRIAN Demonstration

The demonstration of ***i***MONDRIAN shows how the tool can be used by biologists, throught the lifecycle of annotations, starting from their insertion, to their querying and

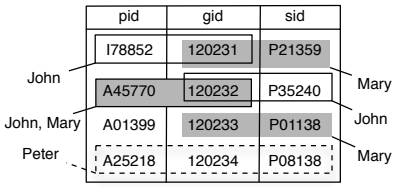

Fig. 1. An annotated relation

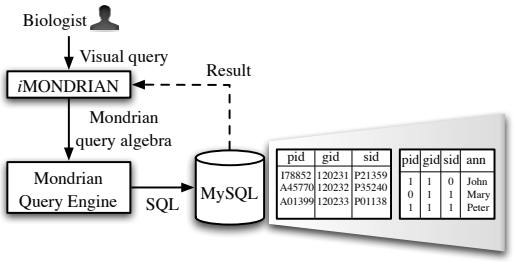

Fig. 2. The MONDRIAN Architecture

ending with their deletion. The demonstration uses data and annotations from Gene Ontology (GO) [6], a publicly available biological database.

2.1 System Architecture

The MONDRIAN architecture, shown in Figure 2. MONDRIAN is built in java and is running on top of MySQL. The ***i***Mondrian component is the front-end through which a user interacts with the system. A visually expressed query is translated to a query written in the MONDRIAN query algebra and this is subsequently translated to SQL and is executed over the underlying RDBMS. One advantage of MONDRIAN queries is that they are *storage-model independent* [1]. That is, MONDRIAN queries are at a level of abstraction that is independent of the chosen representation of annotations. Unlike the executed SQL queries, a change in this representation does not require the reformulation of our queries.

2.2 Demonstrated Functionality

Figures 3 and 4 show the ***i***MONDRIAN interface. For ease of presentation, annotations are represented as colors. Thus, sets of values with the same annotation are colored the same. The same color can appear in a tuple over distinct attributes sets and thus colors do not suffice to tell which attributes are annotated as a set. Therefore, when a user selects an attribute value in a tuple, all the other attributes with which this value is annotated are highlighted. Furthermore, a value can participate in more than one blocks and thus it can have multiple colors. Such values are shown in grey, with a black border, and when a user clicks on them she sees all its colors in a popup window (see Figure 3).

During the demo, we show how users can insert new tuples and annotations. An annotation can be inserted by selecting a set of values and attaching a color to them. This color can be either one that is used already in some tuple or a brand new color (annotation).

The user can query both values and annotations in isolation or in unison. For example, to query annotations, the user can select a color from a value and ask for all the tuples that have the same color (annotation). For example, a visual query v_1 might

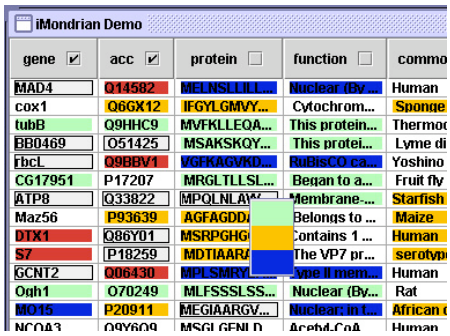

Fig. 3. The ***i***MONDRIAN interface

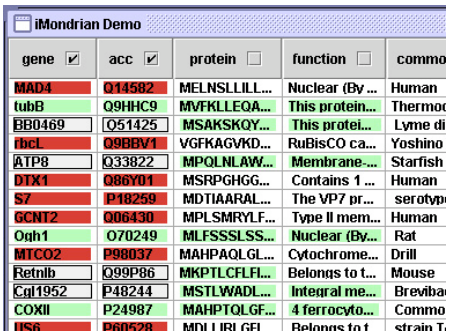

Fig. 4. The result of a visual query

ask for all the annotations with a red or green color. Or, the user can select a number of attribute columns (by clicking check boxes next to each attribute name) and pose a visual query v_2 that returns all the tuples with annotations that involves all of these columns. Each visual query results in a new window containing the query result. The user can pose queries in the result window of a previous query, thus allowing for building of comlex queries. Figure 4 shows the result of applying visual queries v_1 and v_2 to the relation of Figure 3. The composed query, written in the MONDRIAN query algebra, is available from the *cheat window*. This is useful if a user wants to execute periodically the same query. Then, she doesn't have to go through the same steps in the ***i***MONDRIAN interface. She only needs to copy the algebra query from the cheat window and send it directly to the MONDRIAN query engine.

The demo also illustrates how MONDRIAN supports alternative annotation semantics [1]. For example, we discuss *annotation (non-)inheritance* a property that, given an annotation over a set of values, it determines whether, or not, any subset of these values also inherits the annotation. Finally, the demo illustrates some basic provenance functionality, which allows to trace back the origin of annotations.

References

1. Geerts, F., Kementsietsidis, A., Milano, D.: MONDRIAN: Annotating and querying databases through colors and blocks. In: ICDE. (2006) (To appear).
2. Kementsietsidis, A., Arenas, M., Miller, R.J.: Data Mapping in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In: ACM SIGMOD. (2003) 325–336
3. Tan, W.C.: Research Problems in Data Provenance. IEEE Data Engineering Bulletin **27** (2004) 45–52
4. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An Annotation Management System for Relational Databases. In: VLDB. (2004) 900–911
5. Chiticariu, L., Tan, W.C., Vijayvargiya, G.: Dbnotes: a post-it system for relational databases based on provenance. In: ACM SIGMOD. (2005) 942–944
6. Consortium, T.G.O.: The gene ontology (go) database and informatics resource. Nucl. Acids Res **32** (2004) 258–261