An Inductive Database and Query Language in the Relational Model

Lothar Richter, Jörg Wicker, Kristina Kessler, and Stefan Kramer
Technische Universität München, Institut für Informatik I12
Boltzmannstr. 3
D-85748 Garching b. München, Germany
+49 89 289 19411
{lothar.richter, joerg.wicker, stefan.kramer}@in.tum.de, kskessler@web.de

ABSTRACT

In the demonstration, we will present the concepts and an implementation of an *inductive database* – as proposed by Imielinski and Mannila – in the relational model. The goal is to support all steps of the knowledge discovery process, from pre-processing via data mining to post-processing, on the basis of queries to a database system. The query language SIQL (structured inductive query language), an SQL extension, offers query primitives for feature selection, discretization, pattern mining, clustering, instance-based learning and rule induction. A prototype system processing such queries was implemented as part of the SINDBAD (structured inductive database development) project. Key concepts of this system, among others, are the closure of operators and distances between objects. To support the analysis of multi-relational data, we incorporated multi-relational distance measures based on set distances and recursive descent. The inclusion of rule-based classification models made it necessary to extend the data model and the software architecture significantly. The prototype is applied to three different applications: gene expression analysis, gene regulation prediction and structure-activity relationships (SARs) of small molecules.

1. INTRODUCTION

Inductive databases are databases handling data, patterns and models, and supporting the complete knowledge discovery process on the basis of inductive query languages. Many of the recent proposals for inductive databases and constraint-based data mining are restricted to single pattern domains (such as itemsets or molecular fragments) or single tasks, such as pattern discovery or decision tree induction. Although the closure property is fulfilled by many of those approaches, the possibilities of combining various techniques in multi-step and compositional data mining are rather limited. In the demonstration, we present a prototype system, SINDBAD (structured inductive database development), supporting the most basic preprocessing and data mining operations such that they can be combined more or less arbitrarily. One explicit goal of the project is to support the complete knowledge discovery process, from preprocessing to post-processing. The research extends ideas discussed at the Dagstuhl perspectives workshop "Data Mining: The Next Generation" [1], where a system of types and signatures of data manipulation and mining operators was proposed to support compositionality in the knowledge discovery process. In this work, the main idea was to use the simplest possible signature (mapping tables onto tables) as a starting point for the exploration of more complex scenarios.

The development of ideas was guided by several use cases, for instance, the analysis of the NCI DTP HIV data [9] or gene regulation prediction [4]. Starting from concrete scenarios for multi-step, compositional data mining, we identified building blocks necessary for their reconstruction in an inductive query language.

For the development of such a system, various paradigms could have been adopted. In SINDBAD, we chose the relational model, as it possesses several desirable properties, from closure to convenient handling of collections of tuples. Moreover, it is possible to take advantage of mature and optimized database technology. Finally, systems supporting (variants of) SQL are well-known and established, making it easier to get users acquainted with new querying facilities. Thus, we took the same approach as, for instance, Meo et al. [10] and devised a data mining extension of SQL. The extension is more comprehensive than previous approaches by covering discretization, feature selection, pattern discovery, clustering and classification. Similar approaches have been taken by Imielinski and Virmani [7], and Han et al. [6]. For a comprehensive discussion of these query languages and the current lack of preprocessing (and postprocessing) primitives, we refer the reader to a survey by Boulicaut and Masson [2].

2. SINDBAD: CONCEPTS AND IMPLEMENTATION

The goal of the SINDBAD system is to support the whole knowledge discovery process, from pre-processing via data mining to post-processing, on the basis of database queries. The approach adopts the relational model. As each preprocessing and data mining operator returns a table, queries can be arbitrarily nested. In this way, the kind of compositionality needed in multi-step data mining can be achieved

EDBT'07, March 25–30, 2008, Nantes, France.

easily. One of the main ideas of the system is to support the knowledge discovery process by a successive transformation of data. The mining operators were designed in analogy to relational algebra and SQL: For instance, we made heavy use of the extend-add-as operator and devised a feature-select clause in analogy to the select clause. SIQL (structured inductive query language), the query language supported by SINDBAD, is a straightforward extension of SQL. Instead of just adding complicated data-mining operators to SQL, we focused on incorporating small, but extensible and adjustable operators that can be combined to build more complex functions.

From each category of preprocessing/mining algorithms, we implemented most fundamental representatives. For discretization, we included equal-frequency/equal width, for feature selection a filter approach based on information gain or variance, for pattern mining, the computation of frequent itemsets using APriori, for clustering k-Medoids, and for classification k-nearest neighbor and rule induction (pFOIL, a propositional variant of FOIL [11]). External tools can be integrated via wrappers.

We adopted the `extend` [3] operator to add the results of various data mining operations as new attributes to a given relation. It computes a function for each tuple and adds the result as the value of a new attribute. The most general form of an `extend` clause is given as follows:

```
<extend-clause> ::= extend <relation>
                    add <function>
                    as <att>
```

As an example, consider we want to add a new attribute `gmwt` to a table `p`, defined as the attribute `weight` multiplied by 454 [3]:

```
> extend p
 > add (weight*454)
 > as gmwt
```

In SINDBAD, the `extend` operator adds the result of clustering, instance- or rule-based predictions, and sampling to a table. For clustering/classification, the cluster/class membership is indicated by an additional attribute. In sampling, the sample membership determined by a random number generator is given in the new attribute. In this way, we can split datasets, for instance, into a training set and a test set. For clustering and instance-based learning (k-nearest neighbor), other methods for handling tuples and distances are provided as well.

One of the central concepts of SINDBAD is that of distances between objects. This is not restricted to tuples of a single relation. Using relational distance measures, it is possible to apply clustering and instance-based learning to multi-relational data [12]. Most relational distance measures are based on recursive descent and set distances, i.e., distances between sets of points. In the simplest case, the computation of a distance between two sets of tuples A and B boils down to computing the minimum distance between two elements of each set (single linkage), $d_{SL}(A, B) = \min_{a \in A, b \in B} d(a, b)$. Other measures, also known from hierarchical agglomerative clustering, include complete linkage and average linkage. In Table 1, various parameters for configuring multi-relational distance measures in SINDBAD are shown.

Particularly in biological applications, we often find tables where all attributes are of the same type. In this case, the possibility to transpose a table makes sense. Therefore, we included a primitive for table transposition in SIQL. If a table is transposed, tuple identifiers become attributes, and vice versa.

Table 1: Parameters of multi-relational distance measures

```
(11)> configure
     > multirelational_recursion_depth = 3;
(12)> configure
     > multirelational_exclude_tables = '';
(13)> configure
     > distance_between_instances = euclidean;
(14)> configure
     > distance_between_instance_sets =
     > single_linkage;
```

Table 2 shows an selection of other SIQL operators (for a full list, we have to refer to a more comprehensive publication [8]). Following operators for discretization and pattern mining (frequent itemsets), a feature select clause reminiscent of the select clause in SQL is presented. Feature selection can be done according to various criteria specified in `<fscondition>`, for instance by applying hard thresholds, or by relative thresholds (`in top`).

One of the most recent additions is the inclusion of full-fledged predictive models in the form of rule sets. For simplicity, we chose pFOIL, a propositional variant of the traditional FOIL algorithm [11]. The addition of models required significant extensions of the data model of the system. Models can be composed of component models. The evaluations of *component models* (e.g. class predictions) can be aggregated via *combining functions*. Combining functions can be defined in terms of logical or arithmetic operators. In this way, rule sets, weighted rule sets, trees, linear classifiers, and ensembles can be handled conveniently.

Table 2: Selection from query language definition: discretization, pattern mining, and feature selection

```
<disc-clause> ::=
     discretize (* | <att-list>)
     in <relation>
<pattern-disc-clause> ::=
     frequent itemsets
     in <relation>
<feature-select-clause> :: =
     feature select <conditions-on-tuples>
     from <relation>
     where <fs-condition>
<fs-condition> ::=
     ((variance | infogain <att>)
     ((<|>|=|<=|>=) <real> |
     in top <integer>))
```

The SINDBAD prototype is implemented in Java. For parsing the queries, we used the lexical analyzer generator JFlex (see `http://jflex.sourceforge.net/`) and the parser generator Cup (see `http://www2.cs.tum.edu/projects/cup/`). The implementation supports arbitrarily nested queries. In

the future, we are planning to integrate a full-fledged analysis of parse trees, opening possibilities for query optimization. The system is built on top of PostgreSQL (see `http://www.postgresql.org/`), an open source relational database management system. Most of the inductive queries are broken down and translated into a larger number of less complex non-inductive queries. The implementation of data mining features as PostgreSQL functions seems to be critical for performance.

3. DEMONSTRATION OVERVIEW

In the demonstration, we will highlight some of the main features of SINDBAD in three real-world applications. In the first application, we test it on the gene expression data from Golub et al. [5], which contains the expression levels of genes from two different types of leukemia. In the second application, the task is to predict gene regulation dependent on the presence of binding sites and the state of regulators [4]. The third application is to predict anti-HIV activity for more than 40,000 small molecules [9].

3.1 Gene expression analysis

Table 3 shows how the AML/ALL gene expression dataset is analyzed step by step. We aim at finding a classifier that predicts the cancer type, either acute myeloid leukemia (AML) or acute lymphoblastic leukemia (ALL), from the gene expression levels of a cell line. The expression levels of a cell's gene are equivalent to metabolic functions of the cell and characteristic for each cell type. The input relation contains attributes stating the expression levels of the genes (that is, one attribute per gene) and one class attribute, which gives the actual tumor subtype (AML or ALL) of the cell. Table 3 shows the input and output of the system without displaying the actual relations.

First, the dataset is loaded, discretized and divided into a training and a test set (queries (20) to (23)). Note that the discretization and labeling as training or test example is done in the second query. The `sample membership` statement conceptually splits a set of examples into two subsets, simply indicated by an additional attribute containing either the value zero or one. The following two queries split the table into two tables based on the previously added information. Queries (24) perform class-sensitive feature selection. As a result, we reduce the dataset to the fifty genes with maximal information gain with respect to the tumor subtype to be predicted. Since the test set should have the same attributes as the training set, we project the former onto the attributes of the latter in query (25). Next, we query for frequent itemsets, that is, co-expressed genes. The co-expressed genes are used to transform the data, because individual genes are usually only predictive in conjunction with other genes. In the following queries, one new attribute per frequent itemset is added to training (27) and test table (28), which specifies which gene occurs in which frequent item set. Then, it uses feature selection to remove the original expression attributes. In this way, each example is represented only by attributes indicating co-expression with other genes. Finally, query (29) induces a k-nearest neighbor classifier on the training table and applies it to the examples in the test table. The predictions are added to the test table as values of the new attribute `predicted_tumor_subtype`. More generally, the k-nn clause adds the values of a predicted attribute to a given test set on the basis of a target attribute of a given training set:

```
extend <testset>
  add knn prediction
    of <targetatt>
    from <trainset>
  as <predictatt>
```

Table 3: Sample run on leukemia gene expression dataset

```
(20)> create table expression_profiles as
    > import ../ALLAML.arff;
(21)> create table
    > train_test_expression_profiles as
    > extend (discretize * in expression_profiles)
    >  add sample membership as test_flag;
(22)> create table
    > train_expression_profiles as
    >  select * from
    >    train_test_expression_profiles
    >   where test_flag = true;
(23)> create table
    > test_expression_profiles as
    >  select * from
    >     train_test_expression_profiles
    >   where test_flag = false;
(24)> create table
    > reduced_train_expression_profiles as
    >  feature select * from
    >    train_expression_profiles
    >   where infogain tumor_subtype in top 50;
(25)> create table
    > reduced_test_expression_profiles as
    >  project test_expression_profiles onto
    >   reduced_train_expression_profiles
    >    attributes;
(26)> create table
    > coexpressed_genes as
    > frequent itemsets in
    >  reduced_train_expression_profiles;
(27)> create table train_set as
    > feature select * from
    >  (extend reduced_train_expression_profiles
    >    add covered by coexpressed_genes
    >    as 'fp')
    > where attribute like 'fp%' or
    >     attribute = 'tumor_subtype';
(28)> create table test_set as
    > feature select * from
    >   (extend reduced_test_expression_profiles
    >     add covered by coexpressed_genes
    >     as 'fp')
    > where attribute like 'fp%' or
    >   attribute = 'tumor_subtype';
(29)> create table
    > classified_test_expression_profiles as
    > extend test_set
    > add knn prediction of tumor_subtype
    > from train_set
    > as predicted_tumor_subtype;
```

3.2 Gene regulation prediction

In the following, we briefly demonstrate multi-relational clustering and classification on gene regulation data [4]. Gene expression is the complex process of conversion of genetic information into resulting proteins. It mainly consists of two steps: transcription and translation. Transcription is the copying of a DNA-template in mRNA mediated by special proteins, so called transcription factors. Translation is the protein formation based on the information coded in the mRNA. The data used here in this work reflects regulatory dependencies involved in the step of transcription. The task on this data is to learn a model that predicts the level of gene expression, i.e., to learn under which experimental conditions a gene is up- or down-regulated. The expression level of a gene depends on certain experimental conditions and properties of the genes such as the presence of transcription factor binding sites, functional categorizations, and protein-protein interactions. The data is represented in five relations (see Table 4). The main table stores the gene identifiers and their expression level, as well as an identifier of the experimental condition. The experimental conditions are given in two separate relations, information about the genes and their interactions to each other in two additional tables.

Given this input, we can compute the similarity of a gene-condition pair using multi-relational distance measures. The results of k-medoids clustering is shown in Table 5.

The results of k-nearest neighbor classification is shown in Table 6. The target attribute in this case is the increase or decrease in expression level. The class attribute is set to $+1$ if the expression is above a certain threshold, and -1 if it is below. K-nearest neighbor is configured for $k = 10$ and the "majority wins" strategy for prediction. This is a good example for the advantage of the support of multi-relational distance measures over simple propositional distance measures. Multi-relational distances make it possible to analyze complex data with algorithms designed for propositional data in an easy and transparent way without further modifications.

Table 4: Relational schema of gene regulation data. The relation gene is the main table and connects genes with experimental setups and expression levels. The fun_cat relation gives the functional category membership of a gene accordign to the FunCat database. The third relation, has_tfbs indicates occurrence of transcription factor binding sites in respective genes whereas in the regulators table experimental conditions and activated regulators are given. The last table p_p_interaction gives the gene product interaction data.

```
gene(gene_id,          fun_cat(gene_id,
    cond_id,                 fun_cat_id)
    level)
has_tfbs(gene_id,      regulators(cond_id,
    yaac3_01,                ybl005w,
    yacc1_01,                ycl067c,
    yacs1_07,                ydl214c,
    yacs2_01,                ydr277c,
    ...)                     ...)
p_p_interaction(gene1_id,
    gene2_id)
```

Table 5: k-Medoids for gene regulation prediction. The resulting table shows in column 2 the gene identifiers, in column 3 the experimental conditions, followed by the change of expression level and the cluster membership in column 3 and 4.

```
(30)> configure kmedoids_k = 5;
(31)> ...
(32)> extend gene add k medoid membership of gene;
(33)> show table gene;
row|gene_id|cond_id                |level|cluster|
1  |YAL003W|2.5mM DTT 120 m dtt-1  |-1   |2      |
2  |YAL005C|2.5mM DTT 180 m dtt-1  |-1   |3      |
3  |YAL005C|1.5 mM diamide (20 m)  |+1   |5      |
4  |YAL005C|1.5 mM diamide (60 m)  |+1   |1      |
5  |YAL005C|aa starv 0.5 h         |-1   |2      |
...|...    |...                    |...  |...    |
```

Table 6: k-nearest neighbor for gene regulation prediction. This resulting table, column 2 and 3 are the same as in Table 5 followed by the predicted class label in column 4.

```
(40)> configure KNearestNeighbour_K = 10;
(41)> extend gene_test add knn prediction
     > of level from gene_train;
(42)> show table gene_test;
row|gene_id|cond_id                      |class|
1  |YBL064C|aa starv 1 h                 |+1   |
2  |YDL170W|YPD 3 d ypd-2                |-1   |
3  |YER126C|Heat shock 40 minutes hs -1  |-1   |
4  |YJL109C|dtt 240 min dtt-2            |+1   |
5  |YKL180W|Nitrogen Depletion 1 d       |+1   |
...|...    |...                          |...  |
```

3.3 Structure-activity relationships

In the last application, we predict the anti-HIV activity of small molecules using the NCI Developmental Therapeutics Program HIV data [9]. Here, the AIDS antiviral screen data of the National Cancer Institute (see `http://dtp.nci.nih.gov/docs/aids/aids_screen.html`) is used. The data is a collection of about 43,000 chemical structures, which are labeled according to how effectively they protect human CEM cells from HIV-1 infection [14]. We search the data for rules describing a compound's activity against HIV. Hence, the data is prepared and randomly split into test and training set. We chose a representation where each attribute in the training and test relation specifies whether or not a chemical substructure occurs in a substance. The attributes are named f1 to f688, each of them representing a chemical substructure occuring in the antiviral screen data. The numbers refer to the order of their detection by the tree mining algorithm which searched these frequent subtrees in the data set. An additional attribute gives the target label, that is, the compound's effectiveness in protecting against HIV. In Table 7, a protocol of the analysis steps is shown. In the first few queries the datasets are prepared and the FOIL rule in-

duction algorithm is configured (50)-(56). In the main step, rules are learned (57) and displayed (58). The sample rule refers to substructures 683, 262, 219, and 165 to predict a compound as active. Finally, the rule set is applied to a test-set, adding its predictions as an additional attribute (59).

Table 7: Rule learning applied to NCI HIV data.

```
(50)> configure sampling_method = 'holdout';
(51)> configure sampling_percentage = '0.25';
(52)> configure sampling_keep_ratio_column =
     > 'activity';
(53)> create table hiv_train_test as
     > extend hiv_formatted
     > add sample membership as test_flag;
(54)> create table testset as
     > select * from hiv_formatted
     > where  test_flag = false;
(55)> create table trainset as
     > select * from hiv_formatted
     > where  test_flag = true;
(56)> configure foil_mdl = 'true';
(57)> create table hiv_rules as learn rules
     > for activity in  trainset;
(58)> show table hiv_rules;
(activity = true  <-  f683 = true AND
 f262 = true AND f219 =  true AND
 f165 = true)
...
(59)> extend testset add
      > model prediction of
      >  hiv_rules
      > as learned_activity;
```

4. RELATED WORK AND CONCLUSION

In the demonstration, we present a new and comprehensive approach to inductive databases in the relational model. The main contribution is a new inductive query language in the form of a SQL extension, including pre-processing and data mining operators. SINDBAD and SIQL differ from related work [2] in – among other things – the support of pre-processing features. Also, it is a real prototype, useful for exploring concepts and requirements on such systems. Since it is at the moment far from clear what the requirements of a full-fledged inductive database will be, it is our belief that we can only find out by building such prototype systems.

The most similar work in the literature and on the market is MS SQL Server [13]. For a detailed comparison, we have to refer to a previous publication [8]. However, the main focus of SINDBAD is the successive transformation of data, which is not the case in MS SQL Server. Moreover, the approach presented here seems to be less rigid, especially with respect to feature selection, discretization, and pattern mining. In future work, we are planning to investigate a more elaborate system of signatures and types. Type signatures would be useful to define admissible inputs and outputs of data manipulation and mining operations. Signatures of operators would enable first steps towards the optimization of inductive queries.

5. REFERENCES

[1] R. Agrawal, T. Bollinger, C.W. Clifton, S. Dzeroski, J.-C. Freytag, J. Gehrke, J. Hipp, D.A. Keim, S. Kramer, H.-P. Kriegel, B. Liu, H. Mannila, R. Meo, S. Morishita, R.T. Ng, J. Pei, P. Raghavan, R. Ramakrishnan, M. Spiliopoulou, J. Srivastava, V. Torra, and A. Tuzhilin. Data mining: The next generation. *Report based on a Dagstuhl perspectives workshop organized by R. Agrawal, J-C. Freytag, and R. Ramakrishnan*, 2005.

[2] J.-F. Boulicaut and C. Masson. Data mining query languages. In Oded Maimon and Lior Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 715–727. Springer, 2005.

[3] C. J. Date. *An Introduction to Database Systems*. Systems Programming. Addison Wesley, 4th edition, 1986.

[4] S. Fröhler and S. Kramer. Inductive logic programming for gene regulation prediction. *Machine Learning*, to appear, 2007.

[5] T.R. Golub, D.K. Slonim, P. Tamayo, P. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–7, 1999.

[6] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.

[7] T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Min. Knowl. Discov*, 3(4):373–408, 1999.

[8] S. Kramer, V. Aufschild, A. Hapfelmeier, A. Jarasch, K. Kessler, S. Reckow, J. Wicker, and L. Richter. Inductive databases in the relational model: The data as the bridge. In Francesco Bonchi and Jean-François Boulicaut, editors, *KDID*, volume 3933 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2005.

[9] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 136–143, 2001.

[10] R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.

[11] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239, 1990.

[12] Ramon and Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37, 2001.

[13] Z.H. Tang and J. MacLennan. *Data mining with SQL Server 2005*. Wiley, IN, USA, 2005.

[14] O.S. Weislow, R. Kiser, D.L. Fine, J.P. Bader, R.H. Shoemakerand, and M.R. Boyd. New soluble formazan assay for HIV-1 cytopathic effects: application to high flux screening of synthetic and natural products for aids antiviral activity. *Journal of the National Cancer Institute*, 81:577–586, 1989.