BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Information Systems

Ing. Petr Kotásek

DMSL: The Data Mining Specification Language

DMSL: Jazyk pro dolování z dat

Short Version of Ph.D. Thesis

Study Field:	Information Technology
Supervisor:	Doc. Ing. Jaroslav Zendulka, CSc.
Opponents:	Prof. Ing. Tomáš Hruška, CSc. Doc. RNDr. Jana Šarmanová, CSc. Doc. RNDr. Jan Rauch, CSc.

Presentation Date: June 26, 2003

KEYWORDS

knowledge discovery in databases, data mining, data preparation, data mining specification language, XML

KLÍČOVÁ SLOVA

získávání znalostí z databází, dolování z dat, příprava dat, jazyk pro dolování z dat, XML

The thesis can be obtained from the Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno.

© Petr Kotásek, 2004
ISBN 80-214-2685-3
ISSN 1213-4198

Contents

Abstract

Our capability to generate and store data has been increasing rapidly in the last years. It is not a problem to store terabytes of data any more. The problem is to melt these huge amounts of relatively primitive information to human-understandable forms – patterns and knowledge. Unfortunately, we are not able to perform this task just by ourselves as the amounts of data are simply too large for our brains to process them. Fortunately, the field of knowledge discovery in databases (KDD) offers a solution: it aims at automated and intelligent extraction of patterns representing implicit knowledge encoded in massive data repositories (databases, data warehouses, WWW, etc.).

Probably the most crucial step in the whole KDD process is the data preparation. Surprisingly, it does not receive much attention among the data mining community, and the thesis tries to fill the gap. We introduce a theoretical framework for the data preparation step of the KDD process, and present an XML vocabulary named the Data Mining Specification Language (DMSL) that is centered around the framework. The wider purpose of DMSL is to provide for platform-independent definition of the whole KDD process, and its exchange and sharing among different applications, possibly operating in heterogeneous environments.

Naše schopnost generovat a ukládat data v posledních letech rychle roste. Už není problém ukládat terabyty dat. Problémem je přetavit tato ohromná množství relativně primitivních informací do forem pochopitelných pro člověka – vzorů a znalostí. Bohužel však není v lidských silách zvládnout tento úkol, protože dat je prostě příliš mnoho, než aby je naše mozky dokázaly zpracovat. Obor dobývání znalostí z databází (DZD) naštěstí nabízí řešení: zabývá se automatizovanou a inteligentní extrakcí vzorů reprezentujících implicitní znalost skrytou ve velkých datových úložištích (databázích, datových skladech, WWW, atd.).

Pravděpodobně nejkritičtějším krokem celého procesu DZD je příprava dat. Překvapivě se jí však nedostává přílišné pozornosti a proto se tato práce snaží existující mezeru zaplnit. Zavádíme teoretický aparát pro krok přípravy dat procesu DZD a představujeme XML jazyk Data Mining Specification Language (DMSL), který je na něm postaven. Širším cílem DMSL je umožnit definici procesu DZD nezávisle na platformě a její výměnu a sdílení různými aplikacemi pracujícími v heterogenních prostředích.

1 Introduction

Nowadays, there are huge amounts of data stored in various repositories (mostly relational databases) and it is behind human capabilities to reasonably process them. It is no longer possible for us to look at data, see any useful patterns in it and consequently derive some potentially useful knowledge from our observations. The process of *knowledge discovery in databases (KDD)* addresses this problem by aiming at the discovery of valid, novel, potentially useful, and ultimately understandable patterns in large amounts of data stored in information repositories [16].

1.1 The KDD Process

The process of KDD is usually defined as a sequence of specific steps. Its definition can vary slightly from one author to another [15, 16, 20, 21, 28, 32], but the general shape remains the same. We define the process of KDD as a sequence consisting of the following steps:

1. *understanding the domain* involves formation of the picture of what exists in the domain, or gathering the relevant facts about the domain:

    - *exploring the problem space* comprises problem identification and precise problem definition, and problem ranking,
    - *exploring the solution space* deals with identification of what the outcome of the KDD process should look like (e.g., reports, graphs, program code, etc.),
    - *specifying the implementation method* details how the solutions to the problems are going to be applied in practice,

2. *data preparation* involves converting data into the shape suitable for mining:

    - *data integration* realizes combination of multiple data sources,
    - *data selection* retrieves task relevant data,
    - *data transformation* reshapes data into the form suitable for mining,[1]
    - *data cleaning* addresses problems arising from noise and inconsistent data,

[1]As a matter of fact, data selection can be seen as a primitive subpart of data transformation. In the following text, when speaking about data transformation and not mentioning data selection, keep in mind that we adopt this view.

- *data survey* builds an overview of data to uncover what is contained in it, whether it is able to provide expected answers, where the danger areas are, and so on,

3. *data mining* is an application of intelligent methods to the prepared data to extract patterns from it,
4. *postprocessing of discovered patterns* includes:
   - *pattern evaluation* where *interestingness measures* are evaluated to identify patterns that represent knowledge,
   - *knowledge presentation* employs visualization and knowledge presentation techniques to present the knowledge to the user,
5. *putting the results into use* is the final step where discovered knowledge is used in order to realize the solutions and implementation methods set in step 1.

Sometimes the KDD process is defined as consisting of the steps 2–4 only (e.g., in [20]). In [32], the term "data exploration process" is used instead of "knowledge discovery in databases".

The KDD process is inherently **iterative** and **interactive**; e.g., postprocessing of patterns can make the user employ some other kind of patterns, the results of data mining can imply changes in the data preparation step and/or even reveal unknown facts about the domain, etc., while all these activities are directed and attended by the user.

Data mining is a step in the KDD process. However, the shorter term "data mining" is more popular and is often incorrectly used to refer to the whole process, instead of the correct term "knowledge discovery in databases". So, although we agree that data mining is just one step in the KDD process, we will also adopt this misguiding terminology and use the term "data mining" to embrace the whole process.

2 State of Art

Data mining is a fast evolving, interdisciplinary field. It exploits work from a wide range of areas including database technology, information science, statistics, machine learning, neural networks, knowledge-based systems, high-performance computing, data visualization, etc.

Until recently, the vast majority of research activities has been concentrated around the data mining step of the KDD process: design of new data mining techniques and algorithms, performance optimizations, data access, etc., are the most frequent research topics. Naturally, as the field evolves, more and more studies on other topics appear.

2.1 Research Topics in Data Mining

The main research topics in data mining are introduced below. The list is probably not exhaustive, but it definitely contains all the important topics. Some of the topics have been addressed extensively by researchers and highly developed solutions exist for them, while others still represent challenges that need further investigation. The topics should not be looked at as isolated issues – they overlap and are highly interrelated.

The **mining methodology** topics include mining different kinds of knowledge, data preparation, data mining query languages, incorporation of background knowledge, pattern evaluation, mining relational and complex types of data, mining heterogeneous data repositories, architecture of data mining systems, privacy protection and information security, etc.

The **user interaction** topics include interactive mining at multiple levels of abstraction, visual and audio data mining, presentation of results, etc.

The **performance** topics include efficient and scalable mining algorithms, parallel, distributed, and incremental mining algorithms, etc.

2.1.1 Theoretical Models for Data Mining

Research on theoretical foundations of data mining is still immature. At present, there are a number of basic data mining theories, including *data reduction* [5], *data compression* [10, 33], *pattern discovery* (see numerous machine learning and data mining studies on classification, association mining, clustering, and so on), *probability theory* (see statistics studies, such as ones on Bayesian belief networks [22]), *microeconomic view* [26], and *inductive databases* [8, 23] (an inductive database contains relational data and intentionally defined generalizations about the data).

Each of these theories takes a different point of view to capture and explain the basis of data mining. However, they are not mutually exclusive; for instance, pattern discovery can be interpreted as data reduction or data compression.

A systematic and consistent theoretical framework for data mining is necessary because it could provide a coherent environment for development and implementation of data mining technologies. Nevertheless, it is questionable whether such an all-embracing theory is realistic. Data mining is a complex and interdisciplinary process. So, on one hand, it is naturally desirable to have a unifying theoretical platform for it. On other hand, its inherent complexity makes development of such a framework very difficult, if not even impossible. Only time will show whether such a theory is feasible.

More discussion and further references on theoretical models for data mining can be found in [20, 28].

2.1.2 Standardization Initiatives

It holds true in every field that having common standards significantly simplifies development, implementation, maintenance, updating, and integration of applications and systems existing in the field. In data mining, there are a number of established and emerging standards.

The **Predictive Model Markup Language** (**PMML**) [31] is an XML standard for representing data mining and statistical models, as well as some cleaning and transformation operations.

Several data mining APIs have been developed. The **SQL/MM Part 6: Data Mining** [34] specifies an SQL interface to data mining applications and services. The **Java Specification Request-73** (**JSR-73**) [25] defines a pure Java API that supports building and using of data mining models, and access to and processing of data and metadata. The Microsoft's **OLE DB for Data Mining** [30] is a data mining API for Microsoft-based applications.

The **Common Warehouse Model for Data Mining** (**CWM DM**) [13] employs Unified Modelling Language (UML) [35] to specify data mining-related objects, including model representations, model building settings, results of model operations, and so on.

The **Cross-Industry Standard Process for Data Mining** (**CRISP-DM**) [12] is meant to capture data mining projects on the data mining process description level.

More discussion and further references on standardization efforts can be found in [19].

2.2 Languages for Data Mining

2.2.1 Relational-Based Approaches

The **Data Mining Query Language** (**DMQL**) [11, 17, 20] is an SQL-like language for extraction of different types of knowledge (association rules, discriminant rules, classification rules, and characteristic rules) from relational databases and data warehouses at multiple levels of abstraction. It provides for specification of task-relevant data, kind of knowledge to be mined, background knowledge to be used in the mining process, interestingness measures and thresholds for pattern evaluation, and required visual representation of discovered patterns.

MSQL [24] is a rule query language that employs SQL-like syntax and SQL primitives. Its main features include ability to nest SQL, generation of rules from data in response to a query, and further manipulation of results of previous MSQL queries.

The **MINE RULE** operator [9] is designed as an extension to SQL. It supports extraction of association rules from a database and storing them back in a separate relation. The operator provides for specification of task-relevant data, constraints on the structure of rules to be mined, support and confidence thresholds, and taxonomies for mining generalized association rules and for constraining rules structure.

DMQL, MSQL and MINE RULE represent the **relational-based** approach to the problem of data mining languages; they aim at extending the current relational technology to provide support for data mining applications. Comparison of their features and discussion on their relation to the concept of inductive databases can be found in [6]. In [7], a case study on the MINE RULE operator used in the context of inductive databases is presented.

2.2.2 Logic-Based Approaches

In [1, 29], the **logic-based** approach to the problem of data mining languages is presented. The approach is realized in the context of deductive databases and rule-based languages, such as the **Logical Data Language** $\mathcal{LDL}$++ [29] or **Datalog** [1]. The main goal is to provide an intelligent logic-based interface via the definition of *meta-rules* (also called *meta-queries* or *meta-patterns*) that specify general shape of searched patterns as second-order predicates. Meta-rule mining represents a more general and more expressive extension of association rule mining. The main advantage of this approach lies in the fact that the deductive database framework naturally integrates the inductive hypothesis generation and deductive hypothesis verification into a flexible model of interaction. As pointed out in [18], the concept of inductive databases fits naturally in the framework of rule-based languages and deductive databases; a deductive database can easily accommodate both extensional and intensional data.

2.2.3 XML-Based Approaches

In [3], an XML-based environment is proposed to support the overall KDD process. Data mining tasks and their results are specified by means of XML documents. The environment is built around an XML-based query language called **KDDML** that provides for representation of *KDD objects* (rules, classifications, and clusterings) and *KDD queries* (predicates that return either

KDD objects or database objects – tuples of a relation). A KDD query is either a call to an external data mining algorithm (that returns a KDD object) or an invocation of supported operators. Operators operate on KDD objects, database objects, or both KDD and database objects.

The **Predictive Model Markup Language** (**PMML**) [31] provides applications a vendor-independent method of defining data mining and statistical models[2], thus allowing the exchange of models between PMML producers (applications that produce PMML documents) and PMML consumers (applications that consume PMML documents). A PMML document contains the following essential components: data dictionary, transformation dictionary, mining schema, model statistics, and models.

2.2.4 Spatial Data Mining Languages

The **Geo Mining Query Language** (**GMQL**) [27], based on DMQL, is one of the very few attempts in the area of spatial data mining languages [2]. Another attempt is the **Spatial Data Mining Object Query Language** (**SDMOQL**) [4], an OQL-based language designed to support the interactive data mining process in a prototype GIS called INGENS (Inductive Geographic Information System).

3 Motivation and Goals of the Thesis

There are two major issues that motivated the work described in the thesis:

1. As pointed out in [19], agreeing on a common standard for data preparation is one of the major challenges in the data mining standards agenda. Really, in [32], we can read that data preparation is much like weather: everyone talks about it, but no one does anything about it.

   On one hand, everyone in the community agrees that data preparation is one of the most important (and difficult and time-consuming) parts of any data mining project. On other hand, overwhelming majority of existing studies and approaches usually concentrates on data mining techniques and algorithms, data mining query languages, knowledge semantics, optimization techniques, post-processing, pruning strategies, etc. Regarding data preparation, they usually limit themselves to stating that "the task relevant data is obtained by an SQL-like query", "a mineable view is defined", etc. Typically, no attention is paid to data cleaning. Data is usually assumed to have been cleaned and everyone

[2]By a model, PMML understands the output of the data mining process – the knowledge.

acts as if all that has to be done is select the task relevant portion, transform it, and mine it for knowledge.

In this writer's opinion, data preparation is not just **one of the most important steps**. Rather, **it is the most important step** in the whole data mining process: valuable knowledge can be obtained only from data that exposes its semantic content in the right way. This can be very rarely said about the initial original data, so data preparation must take place to expose the semantics of the data to the miner and to the data mining application. If data is not prepared correctly, everything else that follows can turn out to be an enormous waste of time, energy, and money.

2. Data mining is a process that takes place in heterogeneous, modular kinds of environments. Generally speaking, platform-independent, robust, and extensible languages and formats are needed to communicate information in such environments.

   In the field of data mining, there is still no language that would be capable of capturing the whole data mining process.[3] Especially, there is no language for capturing the data preparation step in its whole complexity.

   Nevertheless, as data mining has been here for quite a long time, the state of art has reached the point where an effort towards such a language seems to have a good chance of success: we know enough about data mining to be able to identify general concepts, basic features and essential primitives existing in the field, and to describe them by a language.

Generally speaking, the goal of the thesis is to contribute to the data mining research by developing an approach to the data preparation step of the data mining process, both on the theoretical and application level.

Specifically, arising from the above motives, the goals of the thesis are the following:

1. Developing a theoretical framework for the data preparation step of the data mining process. The framework should cover the following:

   (a) data **selection** and **transformation**; i.e., how existing data are selected and transformed into new data,

   (b) data **cleaning**; i.e., how data are cleaned, or, putting it more specifically, how specific values bearing specific properties are manipulated in order to correct data errors.

[3]Except for an effort towards an XML-based environment described in [3].

The data cleaning process will be covered by the following two concepts:

   i. *value interpretation*; i.e., how values are interpreted (e.g., as valid, invalid, etc.),
   ii. *value treatment*; i.e., how interpreted values are treated (e.g., substituted by other values, ignored, etc.).

2. Developing a platform-independent, robust, and extensible language that will be able to capture the data mining process[4] as a whole. Due to the essential position of the data preparation step in the data mining process, the main focus will be put on the vocabulary for description of this step. Furthermore, the data preparation vocabulary will implement the theoretical framework of goal 1.

   Further data mining primitives that the language should support include domain knowledge, data mining task specification (a data mining query), and knowledge.

4 Data Preparation: The Theoretical Framework

The data preparation formalism is built on the following three mathematical pillars:

- **relations** are used for data representation: data and data mining matrices (and their fields) are modelled as relations,
- **graphs** are used to capture structure of matrix and field dependencies,
- **functions** are used to
  - realize executive functionality; these functions are referred to as *executive functions*,
  - express all the internal relationships existing within the framework; these functions are referred to as *framework functions*.

The formalism introduces a unified framework for data selection, data transformation, and data cleaning. It is able to capture both the data selection and transformation step and the data cleaning step, while it interconnects them naturally and clearly.

[4]Here, by data mining process, we understand its limited version as presented in [20]; i.e., steps 2–4 of the process description of Section 1.1.

The data selection and transformation functionality is realized by employing

- *matrix* and *field transformation graphs* to capture matrix and field transformation dependencies,
- *matrix operators* and *field functions* (i.e., executive functions) to compute matrices and fields,
- framework functions that assign matrix operators and field functions to matrix and field transformation subgraphs,
- *morphism of transformation graphs* that expresses the correspondence that must exist between a matrix transformation graph and a field transformation graph whose fields belong to matrices present in the matrix transformation graph.

The data cleaning functionality is built on the concepts of value interpretation and value treatment:

- Each scalar value carries its implicit interpretation that depends on its origin (it can either be a value in a data matrix or a value in a data mining matrix that was computed from existing values). The concept of implicit interpretation builds the bridge between data transformation and data cleaning: it is a tool for determining interpretation of a computed value of a data mining field, based on interpretation of values that were used to compute it.
- Each scalar value is assigned its explicit interpretation that is computed using the value itself and possibly other values of fields existing in a given matrix.
- Each scalar value is assigned its definitive interpretation that is determined from its actual interpretation and explicit interpretation.
- Each interpreted scalar value is treated by one of the two supported treatment methods: it can either be left untouched or value substitution can take place.

Conceptually, data cleaning is realized by the same kinds of constructs as are those used for data transformation:

- field transformation graph captures field dependencies for computation of implicit interpretation for scalar values of data mining fields,
- *explicit interpretation graph* captures field dependencies for computation of explicit interpretation for scalar values of data and data mining fields,

- *value substitution graph* captures field dependencies for computation of substitutes for scalar values of data and data mining fields,
- executive functions include:
  - *interpretation functions* for computation of implicit, explicit and definitive interpretation of field values, and interpretation of substitutes,
  - *value treatment functions* for computation of treatment methods to be applied to interpreted values,
  - *value usage functions* that determine what kind of values (originals or substitutes) to use for substitute computation when substituting an interpreted scalar value,
- framework functions assign
  - interpretation functions to
    - data fields for implicit interpretation of data field values,
    - field transformation subgraphs for implicit interpretation of data mining field values,
    - explicit interpretation subgraphs for explicit interpretation of data and data mining field values,
    - data and data mining fields for definitive interpretation of values,
  - value treatment functions to fields for treatment of their interpreted values,
  - value usage functions to fields for specification of kind of values to be used for computation of substitutes for their interpreted values,
  - field functions to value substitution subgraphs for computation of substitutes,
  - interpretation functions to value substitution subgraphs for interpretation of substitutes.

The theoretical framework is rather declarative than procedural. That is, it says what the things look like, not how they are used or interconnected. For instance, it captures explicit and definitive interpretation of values and value treatment, but it does not capture their relationship; it cannot express that explicit interpretation should be considered for secondary definitive interpretation that takes place after value substitution, and so on. This is the job of DMSL.

Providing all definitions of the theoretical framework is outside the scope of this brochure. The following definition is the top-level definition of the data preparation model that shelters everything else.

Definition 4.1 *A **data preparation model** dp is an ordered 5-tuple*

$$dp = \langle DM, DMM, dt, vi, vt \rangle$$

such that

1. *DM is a data model,*
2. *DMM is a data mining model,*
3. *a **data transformation model** dt is an ordered 5-tuple*

   $$dt = \langle \overrightarrow{G}_M, \overrightarrow{G}_F, \gamma, \mu, \phi \rangle$$

   such that

   i) $\overrightarrow{G}_M$ is a matrix transformation graph over DM and DMM,
   ii) $\overrightarrow{G}_F$ is a field transformation graph over DM and DMM,
   iii) $\gamma : \overrightarrow{G}_F \rightarrow \overrightarrow{G}_M$ is the morphism of the transformation graphs,
   iv) $\mu : MG \rightarrow MO$ is the matrix transformation subgraph to matrix operator mapping,
   v) $\phi : FG \rightarrow FF$ is the field transformation subgraph to field function mapping,

4. *a **value interpretation model** vi is an ordered 5-tuple*

   $$vi = \langle \iota_V, \iota_I, \overrightarrow{G}_E, \varepsilon, \delta \rangle$$

   such that

   i) $\iota_V : DF_{DM} \rightarrow IF_V$ is the data field to value-based interpretation function mapping,
   ii) $\iota_I : FG \rightarrow IF_I$ is the field transformation subgraph to interpretation-based interpretation function mapping,
   iii) $\overrightarrow{G}_E$ is an explicit interpretation graph over DM and DMM,
   iv) $\varepsilon : EG \rightarrow IF_V$ is the explicit interpretation subgraph to value-based interpretation function mapping,
   v) $\delta : F \rightarrow IF_I$ is the field to interpretation-based interpretation function mapping,

5. *a* ***value treatment model*** *vt is an ordered 5-tuple*

$$vt = \langle \tau, \omega, \overrightarrow{G}_S, \sigma, \eta \rangle$$

such that

i) $\tau : F \rightarrow TF$ *is the field to value treatment function mapping,*

ii) $\omega : F_{tA} \rightarrow UF$ *is the field to value usage function mapping,*

iii) $\overrightarrow{G}_S$ *is a value substitution graph over DM and DMM,*

iv) $\sigma : SG \rightarrow FF$ *is the value substitution subgraph to field function mapping,*

v) $\eta : SG \rightarrow IF_I$ *is the value substitution subgraph to interpretation-based interpretation function mapping,*

where

- *MG is the set of matrix transformation subgraphs over $\overrightarrow{G}_M$,*
- *FG is the set of field transformation subgraphs over $\overrightarrow{G}_F$,*
- *EG is the set of explicit interpretation subgraphs over $\overrightarrow{G}_E$,*
- *$F = DF_{DM} \cup DMF_{DMM}$ is the set of fields,*
- *$F_{tA} \subseteq F$ is the set of fields treated by the treatAs method,*
- *SG is the set of value substitution subgraphs over $\overrightarrow{G}_S$,*

and

- *MO is a set of matrix operators,*
- *FF is a set of field functions,*
- *IF_V is a set of value-based interpretation functions,*
- *IF_I is a set of interpretation-based interpretation functions,*
- *TF is a set of value treatment functions,*
- *UF is a set of value usage functions.*

A data preparation model consists of five main parts:

1. a data model (a set of data matrices),
2. a data mining model (a set of data mining matrices),
3. a transformation model containing matrix and field transformation graphs over data and data mining models, the morphism of the transformation graphs that represents the connection between the models, and the specification of matrix operators and field functions that compute data mining matrices and fields of the models,
4. an interpretation model that covers the implicit interpretation of scalar values of data and data mining fields, the explicit interpretation of scalar values, and finally the definitive interpretation of scalar values,
5. a treatment model comprising value treatment and value substitution (as a specific case of value treatment).

5 Data Mining Specification Language

DMSL represents an attempt to introduce an XML-based language for the description of the overall data mining process that is centered around the data preparation step. The DMSL language implements the data preparation theoretical framework, it is built on it. It must be pointed out that DMSL is just one of the infinite number of languages that can be used to implement the framework. Generally speaking, there can be many languages (syntaxes) that implement a single theoretical framework. The data preparation formalism constitutes **the primary, autonomous, self-contained theoretical framework**, while **DMSL is dependant on this framework**, and **it could not exist without it**.

5.1 Why XML?

The proposed language could have been defined in many other ways (e.g., BNF, completely new proprietary format, etc.), but XML is already here with all its merits and perfectly suitable for our purpose. DMSL is expected to be used in heterogenous and modular kinds of environments, and therefore it should be platform-independent, extensible and robust. Fortunately, these are exactly the features that XML offers! Moreover, XML is accompanied by a large suite of technologies that will come useful while implementing and using DMSL; otherwise, if DMSL were not an XML-based language, the functionality realized by these technologies would have to be implemented from scratch.

5.2 General Structure of DMSL

DMSL identifies five main primitives that play the major roles in the data mining process:

1. The *data model* represents the initial data.
2. The *data mining model* is the extension of a data model. It accommodates transformations made to the initial data.
3. The *domain knowledge* can be employed by data mining tasks while mining data for knowledge.
4. The *data mining task* specifies that certain data is to be mined for certain type of knowledge.
5. The *knowledge* is the output of an execution of a data mining task.

The intended use of DMSL is to capture the whole evolution process from the original data to the knowledge mined from it. In other words, it is meant to describe and store all the information relevant to data mining projects, and enable applications to exchange and share such projects.

Figure 1 shows how different DMSL elements can refer to (depend on) each other. These references are represented by arrows. Full-line boxes represent elements defined by DMSL or by the add-on languages (a suite of languages for definition of other data mining primitives that accompany DMSL). Dotted-line boxes represent content that is specified by other languages.

Providing complete definition of DMSL is outside the scope of this brochure. The following is the root element of DMSL documents. It serves as a container for storing the five main primitives of data mining projects, plus the `FunctionPool` element which can accommodate functions used in data and data mining models.

```
<!ELEMENT DMSL (Header?, (FunctionPool |
                DataModel | DataMiningModel |
                DomainKnowledge | DataMiningTask | Knowledge)+) >
```

6 Summary of Contributions

To the author's best knowledge, the theoretical framework presented in the thesis represents **the first ever attempt to formalize the data preparation step of the data mining process**. Within the borders of this general contribution, we identify other more specific issues that deserve to be pointed out as significant and/or novel contributions:

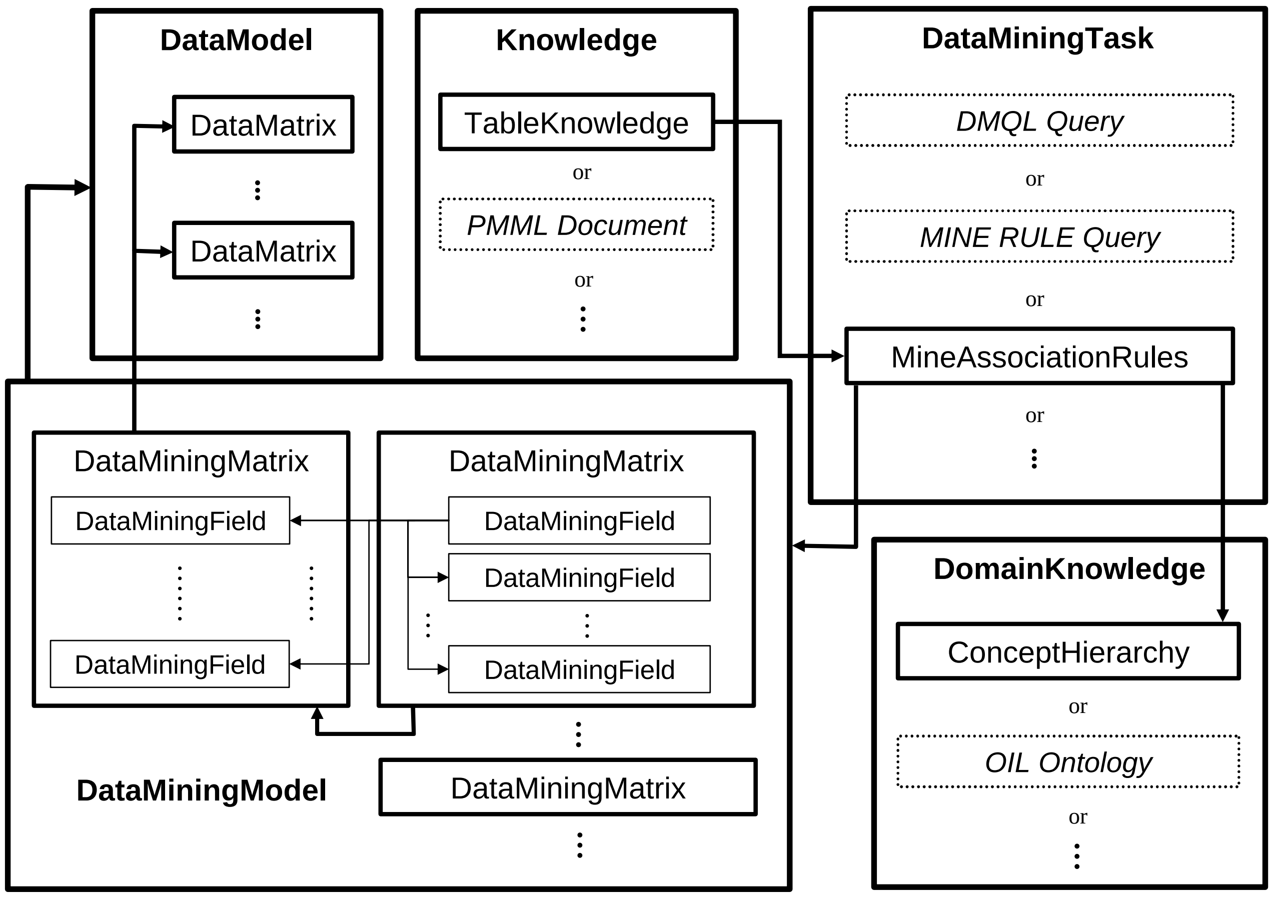

Figure 1: DMSL element dependencies

- **By being built on the mathematical theory of relations, the formalism is widely applicable** as the most widely used and best known data representation form is an $N \times d$ table (also referred to as a data matrix, or, formally, a relation) carrying N d-dimensional rows (also referred to as vectors, or, formally, tuples). Each row represents d measurements (field or attribute values) for one of N objects (individuals).

- **The formalism uses the extended version of the basic form of the relation theory** (that allows only scalar domains and that is the foundation of the relational model); **not only scalars, but also sets of scalars can appear as field values.** It does so by allowing a power set 2^D over a scalar domain D to be a domain of a field. This extension provides for support for transactional data while preserving the original properties of the relational model that allows scalar domains only; the case when only scalars can appear as field values is a special primitive case of the more general case when sets of scalars can appear.

- The formalism represents **a unified framework for data selection, data transformation, and data cleaning**.

  The novel and unique data cleaning functionality is built on the concepts of **value interpretation** and **value treatment**. It is **rich and complex in data cleaning capabilities it offers**, yet **clear and straightforward in its design**. As it is **naturally linked to data selection and transformation** (through the concept of implicit interpretation), it ensures the wide applicability of the data preparation framework.

- **The formalism possesses the closure property.** This is a very important, useful and highly appreciated design principle. Generally speaking, the formalism defines a certain world or domain and certain mechanisms that it uses to work with objects from that domain. Whatever objects from the domain are transformed by mechanisms supported by the formalism, the result is always an object belonging to the domain. The formalism is closed to the outside world – it uses only objects from its domain, and produces only objects that also belong to its domain.

- **The formalism is open and can be easily extended** in various directions if necessary. For instance, it is very easy to add new value interpretations by simply extending the given set of interpretation tags.

The general contribution brought by DMSL is that it provides for platform-independent representation of the data mining process, while the core lies in the data preparation step – data selection, transformation and cleaning. Some of the more specific contributions brought by DMSL stem directly from the characteristics of the underlying theoretical data preparation framework, while some are intrinsic to DMSL itself. The following contributions deserve to be highlighted:

- DMSL enables to represent original data by the data model, and transformations made to the original data by the data mining model that functions as an extension of the underlying data model. Other data mining primitives (namely, data mining tasks) then need to refer to the data mining model only; the underlying data model is encapsulated inside it.

- DMSL fully implements the data representation functionality of the data preparation framework. That is, it supports both scalar and set fields.

- DMSL naturally integrates external and internal data selection and transformation mechanisms – SQL-like statements and DMSL expressions.

This unified view of SQL-like statements and DMSL expressions is enabled by the underlying theoretical framework that uses the concepts of matrix operators and field functions. These concepts are concrete and expressive enough to represent the intended functionality, and, at the same time, general enough to shelter both SQL-like statements and DMSL expressions.

- Regarding nulls processing during data selection and transformation, DMSL works with nulls in the same way as SQL works with them.

- Scalar field transformations are supported by expressions of an external SQL-like language and by DMSL expressions.

- Set field transformations are limited; currently, the only one supported is the identity propagation of set fields through matrices.

- Internal DMSL functions are a very strong tool. Truly, they are the main transformation weapon of DMSL as they enable to model all typical data mining-specific transformations internally within DMSL: value mapping, discretization (binning) and normalization.

  Rather than introducing special purpose elements for respective data mining-specific transformations (as can be seen, for instance, in PMML), DMSL strictly adheres to the underlying theoretical framework and sticks to the most general (while natural, clear and understandable) mechanism for representation of mapping – functions.

- External functions can be used to realize more complicated and/or less typical transformations.

- DMSL fully implements data cleaning for scalar values as defined by the theoretical framework. Moreover, it also extends the data cleaning functionality by introducing ignore treatment methods that can be utilized by data mining tasks, extending scope of treatment methods (they can affect not only treated scalar values but can have broader impact on fields and rows), and introducing interpretation and treatment capabilities to deal with nulls.

- Capabilities of DMSL to transform and clean data are very flexible, as DMSL does not slavishly implement the single-level theoretical framework, but allows for multiple definitions of various features (and provides mechanisms for determining which one to use), naturally combines value interpretation with value treatment (namely, value substitution), and also introduces constructs that simplify various definitions.

- DMSL is open and can be easily extended. This is the consequence of its well-designed modular structure and authentic implementation of the underlying theoretical framework. For instance, it would not be complicated to add new interpretations (both syntactically and semantically), new expressions (due to the wide range of the field functions of the theoretical framework, it should not be a problem to add virtually any expression), etc.

- DMSL can be easily integrated with other languages for description of domain knowledge, data mining tasks, and knowledge.

  Regarding domain knowledge and knowledge, this openness is due to practical reasons: it is convenient and natural to transport domain knowledge and knowledge in one document with the data preparation description and the data mining task, as they are integral parts of every data mining project.

  Regarding data mining tasks, any language that works over tabular data (formally, over relations) can be used since the data model and its extension, the data mining model, are comprised of matrices (relations).

7 Conclusion

With respect to the whole data preparation step of the data mining process as described in Section 1.1, DMSL covers "only" data selection, data transformation and data cleaning, and does not care about data integration and data survey. This is neither a disadvantage nor an omission, this is just an intention.

1. DMSL does not deliberately care about physical data structures that exist under logical data structures it uses – matrices. DMSL is a high-level description tool, and mapping its matrices to physical data structures is solely the problem of the application that uses DMSL. In an integrated data mining environment, there will probably be a module that will facilitate this mapping; it will, for instance, let the user define which DMSL structures map to which physical structures, and so on. In one environment, matrices of a data model can be mapped to tables of a relational database system. In another environment, the same matrices can be mapped to tables of another database system, to comma-separated text files, etc.

2. Data survey is a process that lets the user look into the data, see its structure, understand it, learn about it – just as the whole data preparation

step. It is a heterogeneous process during which the user may use numerous data analysis, statistical, visualization, and other tools. DMSL does not capture this complicated process of looking into the data, it "only" captures the outcome of this process – the final genesis of the prepared data. Why the data is prepared this way and not another is not covered by DMSL; it is the result of the miner's view and knowledge of the data and the domain, the result of experience that he or she gains while preparing the data and understanding the domain. Data preparation does not prepare only the data, it also prepares the miner.

Generally speaking, as DMSL is an XML-based format built on well defined theoretical foundations, it can be easily incorporated into any data mining environment as long as the environment creator wishes to use it. The good feature of DMSL and the underlying theoretical framework is – while their main contribution and power lie in integration of data transformation and data cleaning – that data transformation and data cleaning are still both semantically and syntactically well separated. Thus, it is not a problem to employ only transformation capabilities of DMSL in environments that lack the data cleaning support; data transformation can easily live without data cleaning.

For the complete thesis, document type definitions and examples see [14].

References

[1] S. Abitboul, C. Clifton, R. Motwani, S. Nestorov, D. Tsur, and J. D. Ullman. Query Flocks: A Generalization of Association-Rule Mining. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD'98)*, pages 1–12, Seattle, USA, 1998.

[2] J. Adhikary, J. Han, and K. Koperski. Knowledge Discovery in Spatial Databases: Progress and Challenges. In *Proceedings of the SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 55–70, Montreal, Canada, 1996.

[3] P. Alcamo, F. Domenichini, and F. Turini. An XML Based Environment in Support of the Overall KDD Process. In *Proceedings of the Fourth International Conference on Flexible Query Answering Systems (FQAS2000)*, pages 413–424, Warsaw, Poland, 2000.

[4] A. Appice, D. Malerba, and N. Vacca. SDMOQL: An OQL-Based Data Mining Query Language for Map Interpretation Tasks. In *Proceedings of the Workshop on Database Technologies for Data Mining (DTDM'02), the*

VIII. International Conference on Extending Database Technology (EDBT 2002), Prague, Czech Republic, 2002.

[5] D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. H. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Servcik. The New Jersey Data Reduction Report. *IEEE Data Engineering Bulletin: Special Issue on Data Reduction Techniques*, 20(4):3–45, 1997.

[6] M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. A Comparison between Query Languages for the Extraction of Association Rules. In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2002)*, pages 1–10, Aix-en-Provence, France, 2002.

[7] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Querying Inductive Databases: A Case Study on the MINE RULE Operator. In *Proceedings of the 2nd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD98)*, pages 194–202, Nantes, France, 1998.

[8] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modelling KDD Processes within the Inductive Database Framework. In *Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, pages 293–302, Florence, Italy, 1999.

[9] S. Ceri, R. Meo, and G. Psaila. A New SQL-like Operator for Mining Association Rules. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, pages 122–133, Mumbai (Bombay), India, 1996.

[10] S. Chakrabarti, B. Dom, and S. Sarawagi. Mining Surprising Patterns Using Temporal Description Length. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, pages 606–617, New York, USA, 1998.

[11] J. Chiang, Y. Fu, W. Gong, J. Han, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, W. Wang, B. Xia, and O. R. Zaïane. DBMiner: A System for Mining Knowledge in Large Relational Databases. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 250–255, Portland, USA, 1996.

[12] Cross-Industry Standard Process for Data Mining (CRISP-DM), December 2002. URL: `http://www.crisp-dm.org`.

[13] Common Warehouse Model for Data Mining (CWM DM), December 2002. URL: `http://www.omg.org/technology/documents/formal/cwm.htm`.

[14] Data Mining Specification Language (DMSL), December 2002. URL: `http://www.fit.vutbr.cz/~kotasekp/dmsl`.

[15] M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2002. ISBN 0-13-088892-3.

[16] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press/The MIT Press, 1996.

[17] Y. Fu, J. Han, K. Koperski, W. Wang, and O. R. Zaïane. DMQL: A Data Mining Query Language for Relational Databases. In *Proceedings of the SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 27–34, Montreal, Canada, 1996.

[18] F. Giannotti, G. Manco, and F. Turini. On Query Languages for Data Mining. Technical report, CNUCE-CNR, Pisa, Italy, December 2002. URL: `http://www-kdd.cnuce.cnr.it/papers/vldb_submitted1.ps`.

[19] R. L. Grossman, M. F. Hornick, and G. Meyer. Data Mining Standards Initiatives. *Communications of the ACM*, 45(8):59–61, 2002. ISSN 0001-0782.

[20] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, Inc., 2001. ISBN 1-55860-489-8.

[21] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001. ISBN 0-262-08290-X.

[22] D. Heckerman. Bayesian Networks for Knowledge Discovery. In *Advances in Knowledge Discovery and Data Mining*, pages 273–305. AAAI Press/The MIT Press, 1996.

[23] T. Imieliński and H. Mannila. A Database Perspective on Knowledge Discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[24] T. Imieliński and A. Virmani. MSQL: A Query Language for Database Mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999. Kluwer Academic Publishers, ISSN 1384-5810.

[25] Java Data Mining API (JDMAPI), Java Specification Request-73, December 2002. URL: `http://jcp.org/jsr/detail/073.jsp`.

[26] J. M. Kleinberg, C. Papadimitriou, and P. Raghavan. A Microeconomic View of Data Mining. *Data Mining and Knowledge Discovery*, 2(4):311–324, 1998.

[27] K. Koperski. A Progressive Refinement Approach to Spatial Data Mining. Master's thesis, Warsaw University of Technology, 1999.

[28] H. Mannila. Methods and Problems in Data Mining. In *Proceedings of the 7th International Conference on Database Theory (ICDT'97)*, pages 41–55, Delphi, Greece, 1997.

[29] B. Mitbander, W.-M. Shen, K. L. Ong, and C. Zaniolo. Metaqueries for Data Mining. In *Advances in Knowledge Discovery and Data Mining*, pages 375–398. AAAI Press/The MIT Press, 1996.

[30] OLE DB for Data Mining, December 2002.
URL: `http://www.microsoft.com/data/oledb/dm.htm`.

[31] Predictive Model Markup Language (PMML) Version 2.0, January 2003.
URL: `http://www.dmg.org/pmml-v2-0.htm`.

[32] D. Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann Publishers, Inc., 1999. ISBN 1-55860-529-0.

[33] J. R. Quinlan and R. L. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80(3):227–248, 1989.

[34] SQL/MM Part 6: Data Mining, December 2002. ISO/IEC 13249,
URL: `http://www.iso.org`.

[35] Unified Modelling Language (UML), December 2002.
URL: `http://www.uml.org`.

Author's Curriculum Vitae

Name:	Petr Kotásek
Born:	Brno, 1975
Education:	Brno University of Technology, Czech Republic (1993–2002): Faculty of Electrical Engineering and Computer Science (1993–1998), M.Sc. in Computer Science and Engineering (1998), PhD student at the Faculty of Electrical Engineering and Computer Science (1998–2001) and the Faculty of Information Technology (2002).
Study Stay:	Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique (ESIEE), Paris, the Tempus Mobility Grant, elaboration of the M.Sc. thesis (March–June 1998).
Teaching:	Computer exercises of the *Data Modelling and Database Design* course (1998–2002).
Research Projects:	Grant holder of the FRVŠ (University Development Fund) grant no. 1620/2000 "Knowledge Discovery in Databases and Knowledge Representation" (2000), team member in the EuroMISE-Cardio Center (2001–2002), a project held by the Institute of Computer Science of the Academy of Sciences of the Czech Republic, `http://kardio.euromise.org`.