ADLER: An Environment for Mining Insurance Data

M. Staudt J.-U. Kietz U. Reimer

Swiss Life, Information Systems Research (CH/IFUE), CH-8022 Zurich, Switzerland
{staudt,kietz,reimer}@swisslife.ch

Abstract

The rapid technical progress of hardware and data recording technology makes huge masses of digital data about products, clients and competitors available even for companies in the services sector. Data homogenization and information extraction are the crucial tasks when trying to exploit its inherent (and often hidden) knowledge for improvements of business processes. This paper reports the current activities at Swiss Life tackling both problems. In particular, we sketch the design of a data analysis environment that applies Knowledge Representation and Machine Learning technology to obtain information from relational data. The main focus lies on explaining the usage and representation of meta data within this environment.

1 Introduction

The exploding amount of available digital data in most companies due to the rapid technical progress of hardware and data recording technology has even more increased the tradeoff between just managing the data on the one hand and analyzing resp. exploiting the knowledge hidden in the data for business purposes on the other hand. The supply side of data management is characterized by huge data collections with a chaotic structure, often erroneous, of doubtful quality and only partially integrated. On the demand side we need abstract and high-level information that is tailored to the user's (mostly management people) needs and can directly be applied for improving the decision making processes, for detecting new trends and elaborating suitable strategies etc. In order to bridge the gap between both sides, i.e. to find a reasonable way for turning data into information, we need (efficient) algorithms that can perform parts of the necessary transformations automatically. There will always remain interactive steps for this data analysis and information gathering task. However, the automatic processing should cover all those parts that can not be handled properly by human beings due to the size of transformation input and output. Basically, we are faced with two fundamental problems:

First, we have to deal with a **data homogenization and integration problem** caused by a (due to historical reasons) great variety of tools and data stores on different system platforms and architectures that do not work together or are at least coupled in a very loose manner only. Not only with respect to technical aspects but also due to semantic dependencies, analysis processes require a preceding integration step both at the schema *and* the data level. These heterogeneous "legacy" systems usually serve for concrete *operational* tasks in a company such as bookkeeping or stock list management and are therefore often called Online Transactional Processing (OLTP) systems. Besides heterogenity availability is an important concern, too: Since detailed data analyses are very time consuming they would slow down the OLTP systems to a degree that they could not serve their primary purposes any more.

A solution to this problem is the construction of central Data Warehouses that run independently from the OLTP systems and take over all data from them relevant for analysis tasks. From a database point of view a Data Warehouse is a collection of materialized views on the data in the source systems. Additional views on the warehouse data realize the different levels of abstractions required for different analysis goals and end user needs. The Swiss Life Data Warehouse intended to reach a homogenized base for data analysis is called MASY (see Section 2).

Second, the **information extraction problem** has to be solved. Starting from a consolidated but

Proceedings of the 4th KRDB Workshop
Athens, Greece, 30-August-1997

(F. Baader, M.A. Jeusfeld, W. Nutt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-8/

nevertheless still immense and confusing collection of *data*, transformation, condensation and restructuring is necessary in order to gain *information* with respect to certain analysis goals out of it. In the data warehouse context the term Online Analytical Processing (OLAP) was formed to describe the interactive analysis and exploration of warehouse data and is understood to comprise the support for manual navigation, selection and representation of data. In particular multidimensional views and operations are considered as typical OLAP features. The usage of OLAP technology can be characterized as *verification-driven* approach to data analysis, i.e. the data analyst usually starts with some hypothesis and is supported by respective tool facilities for verification.

In contrast to OLAP, methods for "knowledge discovery in databases" (KDD) aim at the automatic detection of *implicit*, previously *unknown* and potentially *useful* patterns in the data and conversion into pieces of new information that can influence decision and strategies. The data analysis based on KDD is *discovery driven*, i.e. does not start with given hypotheses but searches for new ones (within a given hypothesis space). *Data Mining* algorithms constitute the kernel within the often cyclic and multi-step KDD processes. Other than classic statistical approaches the exploitation of AI technology for data mining tasks was neglected for a long time. However, techniques from the areas of Machine Learning (esp. Inductive Logic Programming), Fuzzy Technology and Neural Networks promise more elaborate kind of knowledge to be discovered in huge data collections.

In order to explore how OLAP tools employed on top of MASY can be complemented with Data Mining tools Swiss Life set up the project DAWAMI. This project is concerned with the design and implementation of the Data Mining environment ADLER (= **A**nalysis, **D**ata Mining and **L**earning **E**nvironment of **R**entenanstalt/Swiss Life). DAWAMI aims in particular at enabling end users to execute mining tasks as independently as possible from data mining experts' support (**Goal 1**) and at making a broad range of data mining applications possible (**Goal 2**) by integrating a variety of different mining methods into a common environment for data transformation and mining.

The rest of this paper is organized as follows: Section 2 summarizes the data integration done in MASY. In Section 3 we discuss our approach to information extraction, namely setting up a suite of Data Mining tools. Section 4 describes the meta data management task within the mining environment which is supported by the repository system ConceptBase. Section 5 enumerates several concrete applications for DAWAMI. Section 6 concludes with an outlook to further steps of the project and to open problems.

2 Homogenization and Integration

The masses of digital data available at Swiss Life – not only data from insurance contracts but also from external sources, such as information about the socio-demographic structure and the purchasing power of the population in the various parts of the country – led to the development of the central integrated Data Warehouse MASY [Fri96]. It comprises data from six OLTP systems: contract data (about 650,000 contracts, some 500,000 clients) plus externally available data collections. Some of the data sources are shown in Figure 1. The basic insurance contract data e.g. stem from the system EVBS while GPRA contains (personal) data about all Swiss Life clients and partners. The system BWV is a publicly available catalogue of all (3 Million) Swiss households.

MASY is implemented on top of ORACLE and follows a ROLAP warehouse architecture, i.e. employs on top of the relational structures a multidimensional OLAP-frontend based on ORACLE EXPRESS. The database itself has both a normalized warehouse scheme gained from integrating the schemas of all source systems, and a derived (redundant) denormalized so-called Galaxy schema intended to efficiently support multi-dimensional access.

The first version of the warehouse in operation is expected to contain around 20 Giga Bytes of data, distributed over approximately 30 tables and 600 attributes.

3 Mining for Information Extraction

Based on the homogenized data in MASY the Data Mining environment ADLER offers tools and a methodology for supporting the information extraction task. While OLAP tools concentrate on mainly interactive exploration, abstraction and restructuring of Data (which also leads to new information) Data Mining provides algorithms for automatic acquisition of knowledge about previously unknown contents of the data. Of course, there always remain interactive tasks, e.g. judging the quality of identified clusters and derived decision trees wrt. to the mining goals. Figure 1 shows the overall architecture of ADLER and its relationship to MASY and the OLTP sources respectively.

3.1 Integrating Data Mining Algorithms

In order to realize Goal 2 stated in Section 1 ADLER supports a great variety of data mining approaches including classical statistical ones as well as techniques from the areas of Machine Learning and Neural Networks. This requires an open framework that allows the integration of quite different types of algorithms

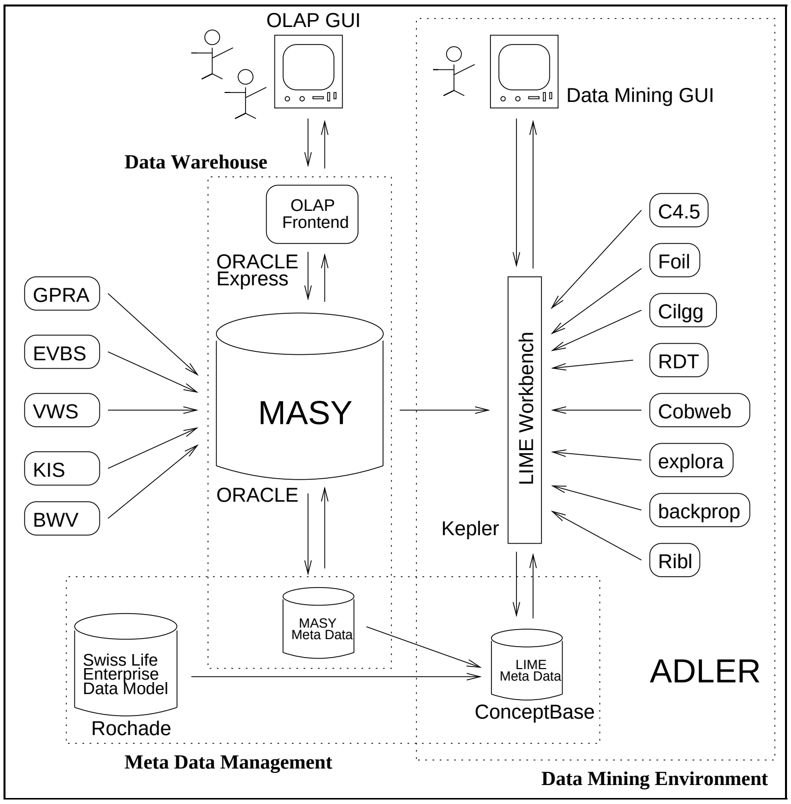

Figure 1: Architecture and Embedding of ADLER

and a high extensibility. The *heart* of the DAWAMI architecture is the Data Mining workbench KEPLER [WWSE96]. This system is the successor of the machine learning toolbox Mobal [MWKE93] and relies on the idea of "Plug-Ins": Its tool description language and the primitives of a basic API enable the building of wrappers around a given implemented mining algorithm and its inclusion into the applicable set of tools.

Figure 1 shows some examples for algorithms that are considered as candidates for becoming components of ADLER. With respect to their output we can distinguish between the following categories:

1. detection of associations, rules and constraints: a common application of these techniques are e.g. market basket analyses.
2. identification of decision criteria: based on decision trees as one possible result we can support tasks like credit assignments.
3. discovery of profiles, prototypes and segmentations: classes of clients with similar properties e.g. can be grouped together and handled in a uniform way.

Another categorization concerns the kind of input data allowed: sets of attribute-value pairs describing properties of certain data objects represented in one single relation (*attribute-based approaches*) or input tuples from different relations (*relational approaches*). The latter category in particular allows to include additional background knowledge and arbitrary combinations of different classes of data objects. While statistical, most Machine Learning and commercially available Data Mining algorithms are attribute-based, Inductive Logic Programming approaches fall into the second category [Kie96]. Compared to solely applying statistical methods the integration of Machine Learning approaches (in addition to giving up the restriction to single relations as input) has e.g. the following advantages:

- more adequate treatment of nominal (categorial) attributes, esp. for hierarchically ordered values,
- faster heuristic methods, e.g. for clustering,
- more comprehensible results.

3.2 End User Support

The *interface* already offered by KEPLER is currently being extended by us with user-friendly database access operators (i.e., to support the selection, configuration and sampling of data from the warehouse that form the basis of the intended mining operations) and additional presentation and evaluation facilities. These extensions contribute to Goal 1 (cf. Sec.1) and are complemented with providing an application methodology relating different mining goals with respective methods.

4 Meta Data Management in ADLER

We now turn to the management of meta data within ADLER. After explaining its basic role we describe the model implemented in the information repository ConceptBase for capturing the meta data relevant for mining with ADLER and give an example for the support during the preprocessing and transformation phase which precedes the actual mining activities.

4.1 The Role of Meta Data

Meta data plays an important role for the integration task in distributed information systems both at design and run time. Figure 1 shows three different types of meta data important in our context:

- The Swiss Life IT department administrates an overall enterprise data model that describes all relevant objects, agents and systems in the company. The actual implementation platform is Rochade.
- The MASY Data Warehouse meta data is mainly stored in certain ORACLE tables. Parts of it will also be ported to the Rochade repository. Typically, such meta data contain information about the integration and transformation process (technical meta data) and the warehouse contents itself, e.g. different (materialized) views and levels of abstraction that are available for the end users (business meta data).
- Among the meta data relevant for Data Mining we can in particular distinguish between the following categories:
  - background knowledge including access to external sources
  - knowledge about data selection and transformations needed to properly feed the mining algorithms
  - application methodology: which tools should be used to solve which problems?

Instead of handling meta data of single applications like classical data dictionary contents within the operational database system – a very common way for Data Warehouse solutions – it is much more appropriate to manage it separately from the data and combine it with other kinds of meta data. As a consequence, we introduce an independent meta data repository in ADLER that plays an outstanding role throughout the whole KDD process. Meta data additional to the one already introduced by the warehouse is needed due to a variety of reasons. Let us have a closer look at this.

Before actually starting a mining algorithm a series of data transformations is usually needed. A Data Warehouse consists of a large number of relations while most mining algorithms only accept as input tuples of one relation. Thus, either one of the existing relations must be selected from which tuples are to be sampled or a new relation must be defined from the existing ones. Most of the time the latter case will apply in order to have all attributes relevant for the mining task combined in one relation. The meta data management is needed to keep track of the views introduced so that the user can always look up what views he currently has and how they are defined. Moreover, to be able to define suitable views at all, the user needes to know about the existing relational schemas, the type of their attributes, which (groups of) attributes are keys, and which attributes can be used to join two relations. He will find all of this information in the meta data repository.

Besides defining the appropriate relational views many more transformations may be needed for a successful mining:

- Different attributes representing the same property may have a different encoding, resulting in different values with the same meaning. Such cases can still occur, despite the homogenization already done when building the data warehouse, because a mining algorithm may also access data from other sources than the warehouse alone (e.g., online databases). The additional data may be part of the input in the case of a relational mining algorithm, or is part of the background knowledge of an attribute-based mining algorithm.
- Certain attributes may be inappropriate for mining. For example, the value range of a numeric attribute may be very large, although only the magnitude of a value and not the exact figure is of interest. In such a case similarity measures as used for clustering would weight such attributes improperly. Taking the logarithm of all attribute values instead would lead to more comprehensible mining results. Another example is the case of an attribute which is cardinal in its nature but is encoded by single characters that stand for certain intervals. As a consequence, it is no longer possible to measure the distance between two values, nor to compare for greater or less between the attribute values themselves or with values of another attribute. It would be better to represent each interval by its mean value.
- Attributes derived from several other attributes may explicitly express facts which have only been implicit in the data so far. For example, instead of having an attribute for the total income of

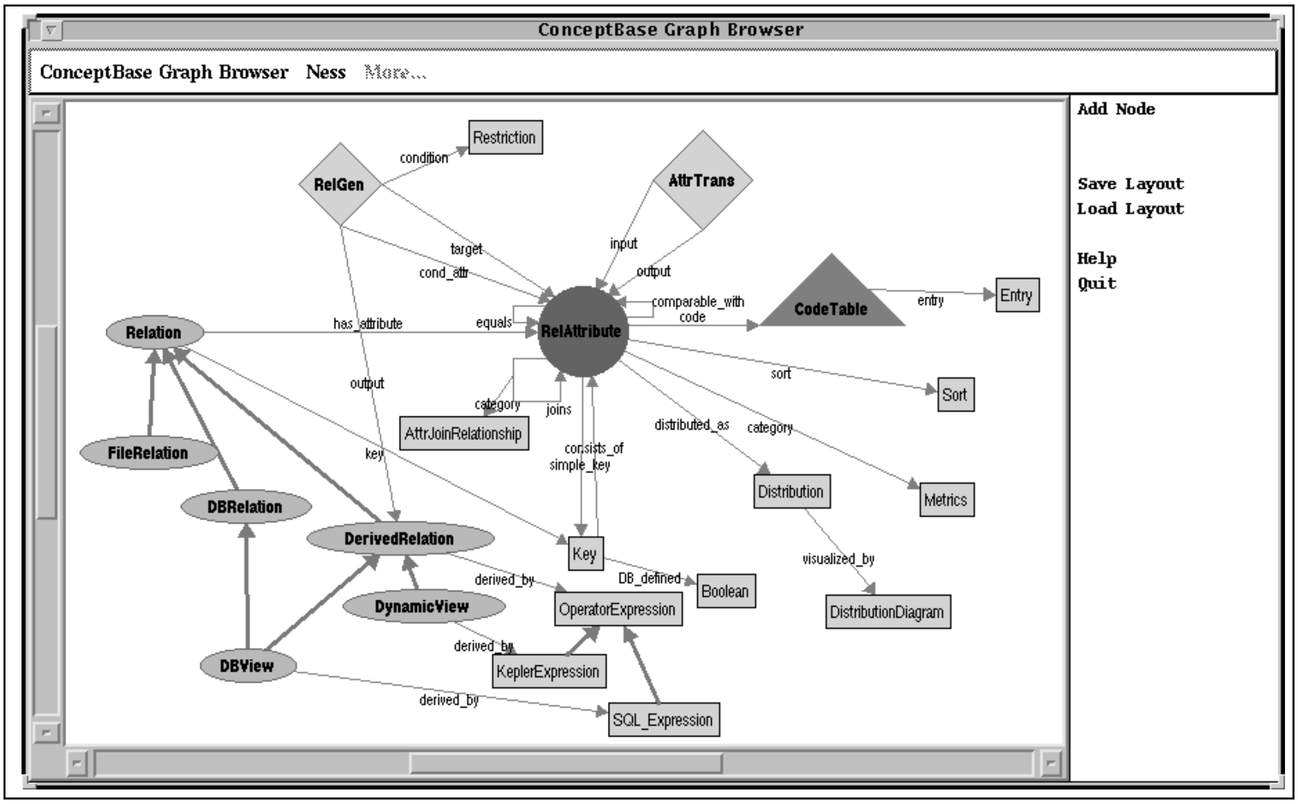

Figure 2: Meta Model of ADLER: Data Representation and Preprocessing

a household and an attribute for the number of persons living in it, better mining results could be achieved when using an attribute for the per capita income, which can easily be derived from the two given ones.

- Certain attributes may not be useful as part of a mining input and should be projected away. Examples are nominal attributes with too many missing values or with too many different values so that they have nearly key property (or are even keys).

All of the above mentioned kinds of transformations are only possible when there is a meta data repository with the required information about attributes. Moreover, with such a repository the KDD environment can decide upon the required, recommended, and the possible transformations itself and perform them automatically, maybe after having asked the user for permission.

Finally, another very important functionality that becomes possible with a meta data repository is a component that guides the user through the KDD process and gives him advice (cf. Goal 1). For example, when the decision tree constructed by a mining algorithm is too complex ADLER may tell the user that certain attributes in the input data are noisy and after all quite insignificant for the mining task so that they should be left out. The system may also examine the characteristics of the input data and of the output and suggest which parameters of the mining algorithm to set to a different value.

Due to the requirements on a meta data repository discussed above the necessity of a more powerful representation formalism than just entries in dedicated relational tables arises. ADLER employs the deductive and object-oriented meta data management system ConceptBase [JGJ+95] and its knowledge representation language Telos to describe the mining meta data, relevant excerpts of the MASY meta data and the links to the enterprise meta data stored in Rochade.

4.2 The ADLER Meta Model

Figure 2 shows the graphical representation of an O-Telos object collection managed by ConceptBase. The displayed objects consitute parts of the meta schema used in ADLER to control the preprocessing and mining activities and to describe the source data and their relationships. We leave out the above mentioned explicit connection to the warehouse sources and start with relations available for the mining process. Apart from database relations (`DBRelation`) we consider also relations extracted from flat text files available as additional background knowledge not integrated in the warehouse (mainly for experimental purposes) and so-called dynamic views that do not exist in the database (as database views defined on top of base relations by SQL expressions) but are created on the fly during the mining process. KEPLER provides certain dedicated operators to define such relations and even to store them locally. Instances of `DBView` and `DynamicView` are derived relations which are constructed in the preprocessing phase of a KDD session. All types of relations have columns, namely attributes (instances of `RelAttribute`).

Attribute values are very often discrete and possible values of an attribute can be organized in code tables, particularly for reasons of efficiency. An entry in such a table relates a short code word with the actually intended value. While the codes are sufficient for internal handling, the "long" values come into play when interacting with the user. Our database currently employs about 250 code tables with the number of entries ranging from 2 to several hundreds. Discrete attributes can be either nominal or ordinal, a further category comprises cardinal data which is not only ordered but also allows to measure distances between values. These categories are covered by `Metrics`. Furthermore, from a semantic point of view attributes can be assigned to sorts, like money or years. The data type of an attribute (like `NUMBER`, `STRING`) is also recorded in the meta data base but not shown in Figure 2 as well as the allocation of relations in databases, owner resp. user information etc.

One or more attributes usually constitute the `key` of a relation, i.e. their value(s) uniquely identifies a tuple. Single attribute keys can be called `simple_key`, like a client identifier (see table `KNDSTM` below).

Particularly interesting for the restructuring of base data by generating new derived relations are relationships between attributes. On the one hand we have basic relationships given e.g. by foreign key relationships between different relations. In this case we assume `equals` relations between the involved attributes. Typically, joins between relations do not involve arbitrary attributes but only those where it makes sense. The model allows to state such attribute pairs by specifying `joins` links, possibly together with additional information concerning the join result (e.g. 1:n or n:m matching values). Links between attributes labeled `comparable_with` give additional hints for possible comparisons during the mining phase. More complicated relationships between attributes result from applying attribute transformation operators (instances of `AttrTrans`) to an attribute (as `input`) to obtain another (new) attribute (`output`). The application of such operators will be explained in the example below. The generation of derived relations requires operators, too (`RelGen`). These specify `target` attributes of the (new) relation, a `condition` part posing certain restrictions both on these attributes as well as on additional ones (`cond_attr`). From all involved attributes the associated relations needed for executing the necessary joins and performing the selections can be derived implicitly.

4.3 Preprocessing Support: An Example

In order to give an idea how the meta schema explained above can be used in ADLER at runtime we now show an example stemming from the preparation of our first analyses of the data available in MASY. For the example we assume three relations:

- `KNDSTM`: client data relation with key attribute `CKNDNBR` for the client number and about 100 further attributes, e.g. birthdate (`DGBTDTE`), martial status (`NZIVSTD`), number of children (`NANZKND`) and buying power class (`CKFLKLS`);
- `KNVPOL`: insurance policy relation; relates policy numbers and certain other information (in particular the contract date `DPOLDTE`) about contracts with foreign key `KP_CKNDBR` to `KNDSTM`;
- `CELL`: general data (e.g. type of flats or houses (`NGBDART`) or typical professional qualification (`NFNKSTF`) of inhabitants) about "cells", i.e. units of 30 - 50 households in Switzerland, with key `CELLID`.

The records in `KNDSTM` are partly linked to cells by the (foreign key) attribute `K_CELLID`. Thus we can obtain further information about the client's environment via the cell he is living in.

In Telos notation the meta data repository would contain this information as shown as an excerpt in Figure 3. Note, that we *instantiate* the classes in Figure 2. The `has_attribute` values of the instances of `DBRelation` simply name the relevant attributes. The foreign key relationships mentioned above are modelled as instances of the `equals` link from `RelAttribute` to itself.

In order to construct a relation `NKND` for a certain mining task which restricts the client set to those people who made a new contract since the beginning of last year, we perform the transformation steps given below. We are in particular interested in the age and the (weighted) buying power class distribution and want to include the `NGBDART` and `NFNKSTF` attributes from `CELL`.

1. The birthday attribute `DGBTDTE` of `KNDSTM` has a specific date default for missing values. This default value influences the age distribution, so it should be explicitly elminated by `NULL`. The result is an attribute column `DGBTDTE'`.

```
KNDSTM in DBRelation with
  has_attribute
    a1:  CKNDNBR;
    a27: DGBTDTE;
    a34: NANZKND;
    a35: NZIVSTD;
    a45: CKFLKLS;
    a67: K_CELLID;
    ...
end
```

```
KNVPOL in DBRelation with
  has_attribute
    a1:  KP_CKNDBR;
    a17: DPOLDTE
    ...
end
```

```
CELL in Relation with
  has_attribute
    a1:  CELLID;
    a49: NGBDART;
    a53: NFNKSTF;
    ...
end
```

```
KP_CKNDNBR in RelAttribute with
 equals
    e:   CKNDNBR
end
```

```
K_CELLID in RelAttribute with
  equals
    e:   CELLID
end
```

Figure 3: Relation and attribute specification in O-Telos

2. From `DGBTDTE'` and the current date the age `NALT` of each client can be obtained.

3. The buying power class of a client should be weighted with his marital status and the number of children. We first code the values of the marital status as numerical values (new attribute `NZIVSTD`) under the assumption that single persons can afford more things (value `1`) while the value of married people is neutral (`0`) and the value of divorced and separately living persons is negative (`-1`).

4. The new weighted buying power class `CPKFLKLS` results from adding this value to the old class value and further reducing it by one for every two children.

5. Relation `NKND` is defined as consisting of the attributes constructed during the previous steps and those of `CELL` mentioned above. In addition to the join between `KNDSTM` and `CELL` and the attribute transformations a join with `KNVPOL` is required in order to eliminate clients without "new" contracts. The join conditions in both cases are derived implicitly from the `equals` links in the model. The selection condition concerning the contract date `DPOLDTE` has to be given explicitly.

Each of the above steps is realized by instances of `AttrTrans` and `RelGen` (Step 5). Instances of `AttrTrans` belong to different types of transformation classes (specializations of `AttrTrans`) with suiting parameters. Instances of `RelGen` can be understood as generalized multi-join operators which are applied to attributes only but with given selection condition and implicit derivation of the participating relations. The attribute transformations concern either one attribute (`NullIntr`, `AttrDecode`, `AttrComp` for replacing certain values by `NULL`, performing general value replacements or arbitrary computations) or several attributes (`AttrComb` for arbitrary combinations of values).

Step 1: build `DGBTDTE'`

```
NullDGBTDTE in NullIntr with
  input
    i:  DGBTDTE
  output
    o:  DGBTDTE'
  value
    v1: "10000101";
    v2: "99999999"
end
```

The `value` attribute (defined for `NullIntr`) specifies the date values (digits of year, month and day) that were 'misused' for marking missing values and should be replaced by `NULL`.

Step 2: build `NALT`

```
CompNALT in AttrComp with
  input
    i: DGBTDTE'
  output
    o: NALT
  comp_expression
    c: ":o = substr(today,4,1) - substr(:i,4,1)"
end
```

The `comp_expression` value allows the derivation of the output value `:o` for attribute `o` of `CompNALT` from the input value `:i` of `i` by subtracting the year component of `:i` from the system date `today`.

Step 3: build `NZIVSTD'`

```
DecNZIVSTD in DecodeAttr with
  input
    i:     NZIVSTD
  output
    o:     NZIVSTD'
  from
    from1: "single";
    from2: "married";
    from3: "separated;
    from4: "divorced"
  to
    to1:    1;
    to2:    0;
    to3:   -1;
    to4:   -1
end
```

Each `from` value is replaced by its corresponding `to` value.

Step 4: build `CPKFLKLS`

```
CombCPKFLKLS in AttrComb with
  input
    i1: NZIVSTD';
    i2: NKFLKLS;
    i3: NANZKND
  output
    o:  CPKFLKLS
  comp_expression
    c:  ":o = :i2 + :i1 - (:i3 div 2)"
end
```

Step 5: generate `NKND`

```
GenNKND in RelGen with
  output
    o:  NKND
  target
    t1: CKNDNBR;
    t2: NALT;
    t3: CPKFLKLS;
    t4: NGBDART;
    t5: NFNKSTF
  cond_attr
    a1: DPOLDTE
  condition
    c: " :a1 > '19960101'"
end
```

Besides the advantage of having a documentation of executed transformations, another motivation for modelling the preprocessing steps is to enable the user to specify the required transformations in a stepwise fashion, while the actual generation of a specified relation is done automatically by ADLER. The automatic generation builds upon the concretely modelled operations like those shown above. Even for the small number of attributes in our example the resulting (SQL) expressions become very complex and their direct manual specification is error-prone and laborious. Of course, there should also be GUI support for protecting the user from having to type the still cryptic Telos specifications.

Based on the relations constructed by various transformations a further preprocessing task concerns the computation of value distributions. The result would also be stored in the meta data repository as instantiation of `Distribution` and a corresponding link from each attribute of interest. The distribution of our synthesized attributes `CPKFLKLS` and `NALT` is shown in the diagrams in Figure 3.

4.4 Meta Data for Mining Activities

While our plans for recording transformation and preprocessing steps are relatively clear, the considerations of how to represent the actual mining activities are more vague. As a first step it is necessary to model the mining algorithms wrt. different input requirements, parameters and results. Each mining session consists of executing algorithms (i.e. instantiating the model) in order to pursue a certain mining goal which obviously relates to the type of the algorithm and the available data sources. We think it is useful to record the complete path from the first transformation in the preprocessing phase to the successfully generated result, like a decision tree, which satisfies the intended goals. In future sessions it can be reconstructed how the whole analysis process took place. Furthermore, we are able to generalize at least to a certain degree by taking together similar mining steps and taking over the significant (e.g. parameter settings) properties into the mining methodology mentioned above, thus giving future users typical examples of a successful mining process. In addition, the preprocessing and mining activities do not constitute a sequential but cyclic process where its is useful even during the same mining session to have the possibility to go back to previous steps.

In this respect, ADLER clearly becomes a support environment that helps the user to keep track of things done and frees him from the burden of dealing with low-level SQL stuff and file in- and ouput for constructing his relational views and storing intermediate results. We thus see our work very much in the direction [BA96] calls for.

5 Applications

For Swiss Life Data Mining has a high potential to support marketing initiatives that preserve and extend the market share of the company. In order to experiment with different mining algorithms and to develop a fixed palette of tools offered in ADLER a number of concrete applications were identified and selected for the first phase of DAWAMI:

Potential Clients: One might think that the wealthy inhabitants of a certain geographical area are the most promising candidates for acquiring new contracts, but this is usually not the case. An interesting data mining task is therefore to find out what the typical profiles of Swiss Life clients are with respect to the various insurance products. An insurance agent can then use these profiles to determine from a publicly available register of households those non-clients of his geographic area which are quite possibly interested in a certain kind of life insurance.

Client Losses: One way to reach lower cancellation rates for insurance contracts is via preventive measures directed to clients that are endangered due to personal circumstances or better offers from competitors. By mining the data about previous cancellations using as background knowledge unemployment statistics for specific regions and questionnaire data, we expect to obtain classification rules that identify clients that may be about to terminate their insurance contracts.

Other mining tasks concern the development of standardized *Type Tests* that allow to include psychographic aspects into client characterizations, the identification of differences between the typical Swiss Life clients and those of the competitors, and the segmentation of all persons in the MASY warehouse into the so called *RAD 2000 Target Groups* based on a set of fuzzy and overlapping group definitions developed by the Swiss Life marketing department some years ago.

6 Outlook and Goals

While Goal 2 stated in Section 1 is a more architectural issue that can be handled as sketched in Section

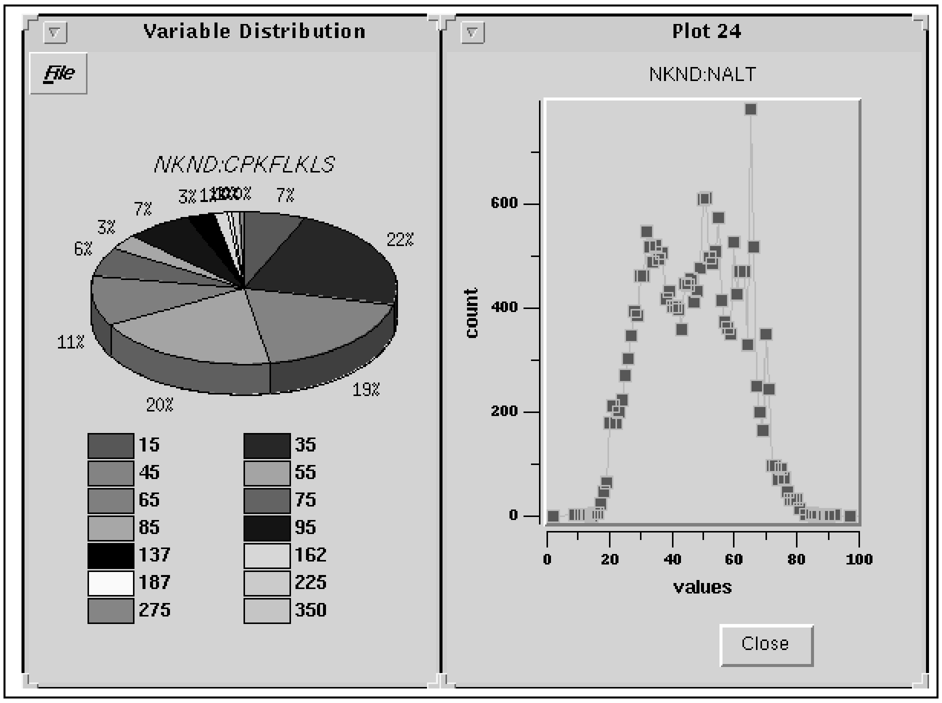

Figure 4: Value distribution of synthesized attributes

3.1 Goal 1 is very ambitious. Data Analysis at Swiss Life currently is conducted by specialists of the Information Center for Market Analyses. As a first step these people should be enabled to apply ADLER in addition to the statistics tools (mainly SPSS) they are already using. In the long run we want also reach users with less background in data analysis. The success in this direction heavily depends on the quality of the ADLER GUI and the provided mining methodology.

Data Mining is a special form of knowledge acquisition. The results of mining, e.g. a classification component that allows to classify people into specific client segments, should be made available to other analyzers than just the one who initiated the current mining process. Therefore, Data Mining has to be embedded in a general concept for Knowledge Management at Swiss Life which also includes an integration task but now on a more conceptual than on a source data level. Meta data resources as mentioned above will definitely be a central part around which Knowledge Management is to be built up.

Besides these more global links the activities and plans described above will serve as a starting point for establishing a stable Data Mining infrastructure at Swiss Life. We want to show that the very common application of database technology for managing huge amounts of data in the company is naturally complemented by technology from the areas of Machine Learning (for data mining) and Knowledge Representation (for modelling domains and high level system integration).

References

[BA96] R.J. Brachman and T. Anand. The process of knowledge discovery in database. In *Proc. of the 2nd Int. Conf. On Knowledge Discovery and Data Mining.* AAAI Press, 1996.

[Fri96] M. Fritz. The employment of a data warehouse and OLAP at Swiss Life (in German). Master's thesis, University of Konstanz, Dec. 1996.

[JGJ+95] M. Jarke, R. Gallersdoerfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal of Intelligent Information Systems*, 4(2):167–192, 1995.

[Kie96] J.-U. Kietz. *Inductive Analysis of Relational Data (in German).* PhD thesis, Technical University Berlin, Oct. 1996.

[MWKE93] K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications.* Academic Press, 1993.

[WWSE96] S. Wrobel, D. Wettschereck, E. Sommer, and W. Emde. Extensibility in data mining systems. In *Proc. of the 2nd Int. Conf. On Knowledge Discovery and Data Mining.* AAAI Press, 1996.