Architecture for Pattern-Base Management Systems

Manolis Terrovitis

National Technical University of Athens,
Dept. of Electrical and Computer Eng.,
Athens, Hellas
mter@dbnet.ece.ntua.gr

Panos Vassiliadis

University of Ioannina,
Dept. of Computer Science,
Ioannina, Hellas
pvassil@cs.uoi.gr

Abstract. In many modern applications we have to deal with huge volumes of data. Many techniques have been developed on how to extract knowledge, statistical usually, from them, especially in the context of data mining. The results of such operations are abstract and compact representations of the original data, which we name *patterns*. Still, these patterns have to be further elaborated to be used in an effective way. In this paper we present the architecture of a *Pattern Base Management System* that can be used to efficiently store, and query patterns. We present its logical structure and we comment on the criteria of whether the existing systems for storing and manipulating data can cover the special user requirements that patterns impose.

1. Introduction

Today's world produces an enormous amount of data in a regular basis. Huge amounts of *raw data* have to be processed and taken into consideration before new solutions and decisions are made in several fields. Raw data are facts that have an implicit meaning and have been recorded from various sources in the real world. This recording can be done by humans through data entry procedures, or by collecting measurements from various instruments or devices, e.g., cellular phones, environment measurements, monitoring of computer systems, etc. (Figure 1). The determining property of raw data is the vastness of their volume; moreover, a certain degree of heterogeneity is also present.

Clearly, data in such huge volumes do not constitute *knowledge* per se; i.e., they cannot be directly exploited by human beings and no useful information can be deduced simply by their observation. Thus, more elaborate techniques are required in order to extract the hidden knowledge and make these data valuable to the end-users. Data mining [BeLi96, FPSU96, HaKa01, Vaz+02] was developed to help extract knowledge from the raw data, using algorithms that could discover several statistic properties in the original data. Data mining produces results like association rules, clusters, decision trees and other structures that describe properties of the raw data. The common characteristic of all these techniques is that big portions of the available data are abstracted and represented by a small number of knowledge-carrying representatives, which we call *patterns*. Patterns in this sense do not appear only in data mining but in several other fields, like signal processing, image recognition, spatial databases and others.

The patterns despite being already the results of some elaboration on the raw data, are not, usually, in a form that can lead us directly to real life results. We need tools that will permit us to compare, query and store the patterns in order to retrieve the information we want. This paper describes the architecture of a system named *Pattern Database Management System* (PBMS) that will provide such tools for pattern manipulation.

The main focus of this work is dual: (a) it describes motivation behind and the architecture of the PBMS, and (b) it explains why such a system in needed and why no existing system for storing and manipulating data can be used. In Section 2 we explain what patterns are, and give briefly some mathematical background for the PBMS. In Section 3 the architecture of the PBMS is described. In Section 4 we list criteria for PBMS characteristics and in Section 5 we conclude.

2. Patterns and Mathematical background

To effectively describe what patterns are in our context and why they are so useful, we present an exemplary scenario, summarized in Figure 1. The available raw data come from a super market, where the goods purchased from each customer are recorded in the database of the super market. While this vast volume of data is not providing us with any particular information on the buying habits of the customers, we can apply a set of knowledge discovery algorithms to the underlying data and come up with some knowledge. In the case of Figure 1, we have applied an association-rules algorithm, thus the knowledge-carrying abstractions of data are association rules. Observe that the association rule of Figure 1, apart from providing the end-user with some hidden knowledge over the underlying data, can be linked to the subset of raw data it is related to (see the arrows in Figure 1).

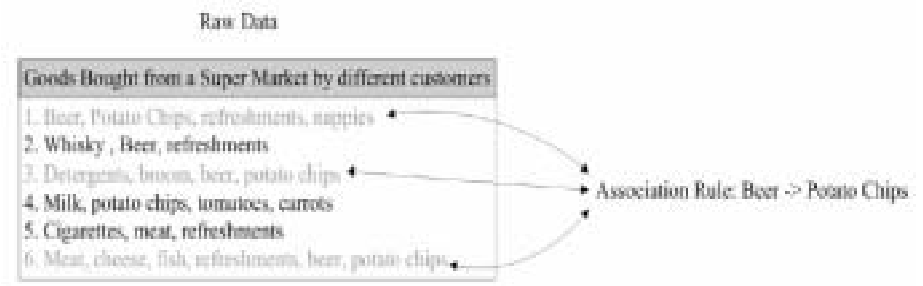
Raw Data
Goods Bought from a Super Market by different customers
1. Beer, Potato Chips, refreshments, nappies
2. Whisky , Beer, refreshments
3. Detergents, broom, beer, potato chips
4. Milk, potato chips, tomatoes, carrots
5. Cigarettes, meat, refreshments
6. Meat, cheese, fish, refreshments, beer, potato chips
Association Rule: Beer -> Potato Chips

Figure 1. Patterns (association rules) and their mapping to raw data [VaHT02]

Patterns, thus, can be regarded as *artifacts*, which describe (a subset of) raw data with similar properties and/or behavior, providing a compact and rich in semantics representation of data. In our setting, patterns must satisfy the following two *properties*:

1. We assume that a mapping between the raw data space and the pattern space is always possible; in the most general case the cardinality of this mapping is `many-to-many`. Hopefully, the amount of data associated to a single pattern through this mapping is large enough to qualify the pattern as a **compact** representation of data.
2. The extracted patterns have to preserve as much knowledge as possible with respect to the raw data they correspond to. In other words, they must be **semantically rich**, both through the knowledge preservation and their conciseness.

Based on the above discussion, we are ready to give a first, informal, definition for patterns.

Definition 1 A pattern can be defined as a compact and rich in semantics representation of raw data. □

Next, we discuss the mathematical foundations of patterns and their relationship to raw data. We can base the discussion on the following assumptions, which directly stem from the aforementioned definitions:

– There exists a *source space* (the space of raw data) and a *pattern space*.
– There always exist *relationships* among the members of the source space and the members of the pattern space. In general, these relationships can be of cardinality `many-to-many`, i.e., a pattern can correspond to more than one data items and vice versa.

Source Space Characteristics. We can assume that each data item in the source space is characterized by a *finite number*, say N, *of dimensions*. In the case of a single relational table, these dimensions would probably correspond to the fields of the table (although one can also consider scenarios where only a subset of these fields is considered, or where extra information, like statistical information derived from catalog histograms, annotates each tuple). We can assume, as usual, that each dimension of the data items belongs to an infinitely countable domain. We will call $dom(x)$ the domain of each dimension. If $A_1,\dots,A_N$ are the dimensions of the data items, the source space is defined as $dom(A_1)x\dots xdom(A_N)$, which we will call D^N. The set of actually stored data comprise a subset of D^N, which we call active source space; we will refer to the active source space as D_A^N .

Pattern Space Characteristics. Independently of which kinds of patterns we are working on, we can assume that *each pattern is characterized by a finite number*, say M, *of dimensions* or *features*. We can assume, as usual, that each dimension of a pattern belongs to an infinitely countable domain. Again, we will call $dom(x)$ the domain of each dimension. If $B_1,\dots,B_M$ are the dimensions of the patterns, we define the pattern space as $dom(B_1)x\dots xdom(B_M)$, which for brevity we will call D^M. Note, that as usually, only a finite subset of the pattern space will be eventually stored, queried and managed inside the pattern management system; similarly to data, we will refer to this subset as the active pattern space D_A^M .

Relationship of data and pattern spaces. Without loss of generality, we can say that each pattern is related to more than one data items. As far as data items are concerned, a data item can also be related to more than one pattern due to:

(a) fuzziness, i.e., the application of an algorithm that does not result in a single crisp pattern value for each data item;

(b) overlapping output, e.g., in the case of a pair of association rules where one is a more general variant of the other;

(c) the simple fact that there can be patterns of different types corresponding to a single data item (e.g., because two different algorithms were executed over the same set of raw data).

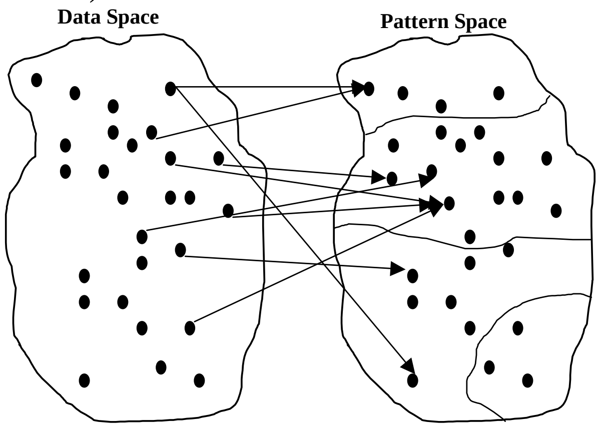

Figure 2. Data and pattern spaces, along with pattern subspaces.

Observe Figure 2, where we can see the relationship between the source space and a certain pattern space. As mentioned before, the relationship is not a function, but in general $f_{DP}: D^N \rightarrow D^M$ is a many-to-many relationship.

Relationship characteristics. By observing the Venn diagram of Figure 2, we can enrich the data-pattern relationship f_{DP} with extra information. Basically, we can define:

(a) *participation measures for the relationship*, e.g., how large subsets of D^N, D^M it involves, if it is total or injective, etc;

(b) *importance measures for a data item*, e.g., how many patterns does it correspond to (practically denoting how important or interesting it might be);

(c) *importance measures for a pattern*, e.g., how many data items it corresponds to, whether the set of its data includes the set of data of another pattern (in which case we can test for pattern zoom-in/out, and of course, identity).

Another issue that we can discuss is that of (a) rich and (b) compact representation of the data items from the part of the patterns. For both of these issues we can give a naive metric. Let us start with richness of representation. Clearly, we cannot always (or, do not wish to) store the entire relationship of patterns and data in persistent storage. Instead, we can introduce *condensed representations* of this relationship (e.g., in the form of expressions in a certain language). The richness of representation can be computed as the fraction of the relationships captured by this condensed representation over the total number of relationships of f_{DP} (actually, we can define two such functions, exactly as the Information Retrieval community defines precision and recall). Another measure could be the success ratio of the 'inverse' mapping, if such a mapping can be derived. As far as the compactness of the representation is concerned, a coarse measure would be the ratio of $\texttt{size}(D_A^M)*M/\texttt{size}(D_A^N)*N$. (Note that M/N is not a good measure, since an association rule over a relation with N attributes can involve up to $2N$ attributes; still if we could characterize 1M data rows with 1K association rules or 5 clusters, this would be compact enough representation).

Subspaces of the pattern space. Another interesting observation lies in Figure 2, where we can see that in general, *the pattern space is the union of a finite set of different sub-spaces*. We can envisage this division in two different cases. In the first case, we can consider a certain subspace comprising the set of association rules over a table of raw data and another subspace comprising the set of decision trees over the same table. In general, exactly as we define tables in the relational paradigm, we can define collections of semantically related patterns (i.e., subspaces of the grand pattern space) at design time. In the second case, one can consider that all the patterns produced by a certain algorithm (e.g., association rules) form a certain subspace (i.e., we have one subspace for decision trees, another for frequent item sets, and so on). Each subspace can be further decomposed in more subspaces, in the same spirit (e.g., the subspace of clusters can be further divided to subspaces according to the type of clustering algorithm / cluster scheme of the patterns). Again, the problem of pattern similarity between the members of the two subspaces is raised. In general, subspaces can be considered as artificial constructs necessary to group semantically and structurally similar patterns together, with the purpose of their further querying.

3. The reference architecture for the Pattern-Base Management System

Patterns can be managed by using a *Pattern-Base Management System* exactly as database records are managed by a database management system. In our setting, a Pattern-Base Management System (PBMS) can be defined as follows.

Definition 2. A **Pattern-Base Management System** (**PBMS**) is a system for handling (storing/processing/retrieving) patterns defined over raw data in order to efficiently support pattern matching and to exploit pattern-related operations generating intentional information. □

The reference architecture of a PBMS is depicted in Figure 3. It consists of three major layers of information organization. In the bottom of Figure 3, we depict the data stores that contain raw data (forming thus, the *Raw Data Layer*). Raw data can be either managed by a DBMS or can be stored in files, streams or any other physical mean that is managed outside a DBMS. At the top of Figure 3, we depict the PBMS repository that contains patterns, which forms the *Pattern-Base Management System Layer*. Finally, in the middle of Figure 3, we can observe the intermediate mappings that relate patterns to their corresponding data, forming the *Intermediate Data Layer*. Intermediate mappings facilitate the justification of any knowledge inferred at the PBMS with respect to the raw data; for example, they could be used to retrieve the rows that produced the association rule of Fig. 1. Ideally, we would like this layer to be part of PBMS, involving specialized storage and indexing structures. For practical purposes, though, the PBMS should be constructed in such a way that it functions even if intermediate results are out of its control (which we would expect as the most possible scenario in real-world scenarios), or even absent.

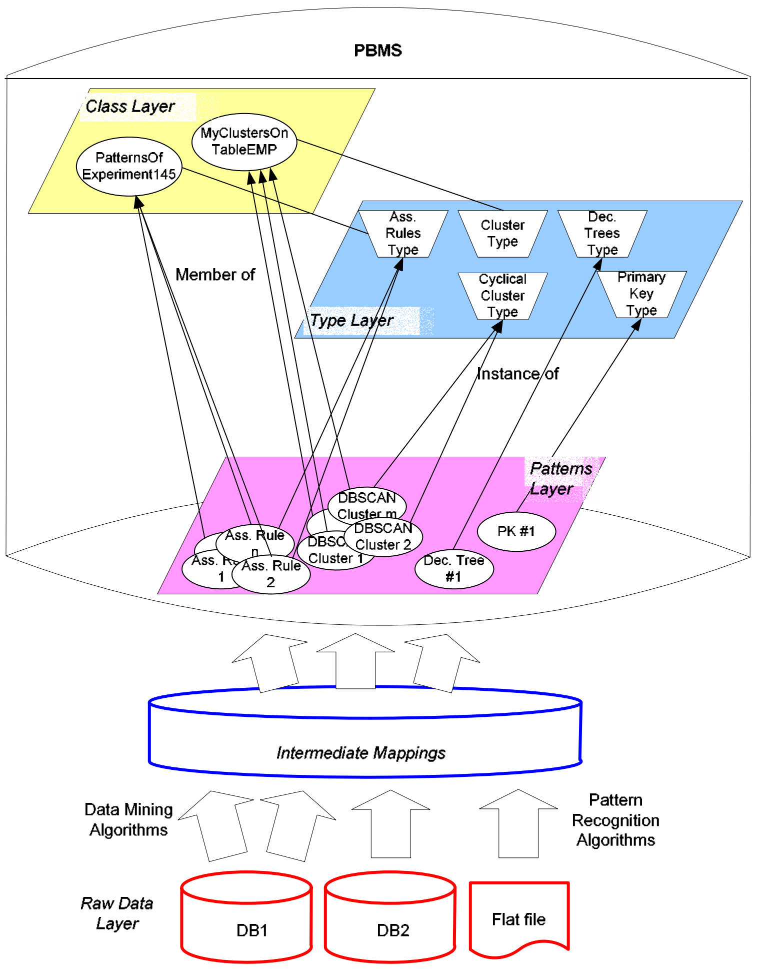

Figure 3. Reference architecture for the Pattern-Base Management System

In the sequel, we will delve into the internals of the PBMS, which will be the focus of our research. We distinguish three internal *conceptual* layers in the architecture of the PBMS (in contrast to the aforementioned *physical* layers of the previous paragraph):

- *Pattern layer*, which is populated with patterns. These are the basic entities that are stored in the PBMS.
- *Type layer*, which holds built-in and user defined types for patterns.
- *Class layer*, which holds built-in and user-defined definitions of pattern classes, i.e., collections of semantically related patterns.

The last two layers are conceptually “higher” than the pattern layer as the entities they define are more abstract. Their role is supportive in the definition and manipulation of the patterns. Next, we present a brief description of all the entities that appear in this conceptual description of the PBMS:

- *Pattern Layer*. The patterns can be either identified through query processing or given a-priori, according to the classification proposed in Section 1. Patterns are related with raw data via the intermediate mappings. In the first case, they can be obtained by different data mining algorithms. In Figure 3, three algorithms have been applied: an algorithm for the extraction of association rules, the DBSCAN algorithm [EKSX996] for the extraction of clusters and an algorithm for the extraction of a decision tree. Besides this, a user can specifically define and store a pattern over the raw data, e.g., a primary key constraint, a functional dependency or an integrity rule. In Figure 3 we depict the case where the designer of the database with the raw data, has defined a primary key constraint over them.
- *Type Layer*. The PBMS pattern types, describe the syntax of the patterns. Patterns of same type share similar characteristics. The intuition behind this is that a large number of operations can be applied to patterns with similar characteristics and get *potentially meaningful* results. Thus, defining a pattern as an instance of a specific type ensures that certain operations will be applicable to it. For example, the overlap of a circle and an inference rule cannot even be defined in an intuitive manner, whereas the overlap between two circles may be of interest. Moreover, the lesson learned from object-oriented databases dictates that different internal representations can be suitable for different types of patterns. Normally, we anticipate the PBMS to come with a set of built-in, popular pattern types. Still, the type layer must be *extensible*, simply because the set of pattern types that it incorporates must be extensible. In fact, we understand the extensibility mechanism as a requirement for the model, so that whenever a new kind of patterns is devised, it can be gracefully incorporated in the PBMS. The type layer must characterize patterns based on the type of source data they are related with, their structure, and the measures we want to associate with them.
- *Class Layer*. The PBMS classes are collections of patterns which share some semantic similarity. Patterns that are members of the same class are obligatorily required to belong to the same type. Classes are used to create patterns with predefined semantics given by the designer; by doing so, the designer makes it easier for the users to work on them in a meaningful way. For example, a class may comprise members that are clusters over `AGE` and `SALARY`, i.e., patterns that represent groups of employees in a company that have approximately the same age and get similar salaries. In order to model these concepts, we assume that supportive entities not appearing in Figure 3, called *features*, belongs to the PBMS. Features form an ontology in the PBMS, whose elements are used in the definitions

of patterns and classes. Features facilitate giving semantics to the patterns or classes that are defined over them. We can imagine features as a dynamic interface over raw data. For example, a feature labeled AGE can be dynamically linked to several data representing the age of employees.

One could elaborate more on extra potential characteristics of the class and type layers; in fact, in [Riz+03] we have introduced the notion of specialization hierarchies among types and the possibility of composing/refining composite types. For lack of space, we refer the interested reader to [Riz+03].

Another meta-entity of the PBMS is *the language* that is used to define the structure of patterns and the condensed expression for representing the mappings between the patterns and their respective raw data. The expressive power of the language and the available features determine the kind of patterns that we can express in a certain PBMS. The PBMS features provide the *semantic domain* of the PBMS; the supported language on the other hand, provides the *expressive power* of the PBMS. By extending the defined features and/or by using a more expressive language, more patterns can be stored and manipulated by the PBMS.

5. Towards a PBMS Manifesto

In this section we will discuss on the necessary principles and features for a PBMS. Clearly, this represents a personal viewpoint; still we believe it is well justified by interesting practical applications. First, we give a short list of requirements and characteristics for a PBMS. We group the requirements in functional and system areas. Then, we will briefly compare our approach with respect to the current state of the art in object-relational and object-oriented DBMS's.

Data model

1. A PBMS will be based upon a *generic uniform model* that covers all kinds of patterns. The model captures all static, dynamic and well-formedness properties of *all* patterns (i.e., structure, operations and integrity constraints), just as the relational model does for database records.
2. A PBMS model represents its instances in a *compact* and *rich in semantics* fashion (i.e., we gain a concise representation with respect to the raw data and we do not lose information; in fact, we abstract new knowledge by escaping from the vast volume of raw data).
3. A PBMS will support different types of patterns in an *extensible fashion*. Whenever new kinds of patterns are discovered, or considered to be of interest to the PBMS users, they will be smoothly integrated in the PBMS, through an extensibility mechanism.
4. A PBMS will allow its user to be able to identify interesting subsets of the pattern space on the basis of their *semantical similarity*. This corresponds to the aforementioned notion of classes.
5. A PBMS will support *composite* patterns, generated from simpler ones. This implies that several levels of representation/abstraction should exist among

patterns, including different levels of granularity, multi-dimensionality, recursion, hierarchies, etc.

Architecture

6. A PBMS will have its own mechanisms for *representing and storing its entries*.
7. A PBMS will cooperate with DBMS's storing *raw data*; however raw data can also be stored outside a DBMS.
8. A PBMS must be able to manage also *pattern extraction and creation*, through the appropriate mechanisms.
9. A PBMS will allow access to *intermediate results of pattern creation algorithms*; still, whenever access to these data is not possible, the PBMS will continue to function properly, compensating the loss of knowledge through appropriate mechanisms.

Query Language and Processing

10. A PBMS processes different kinds of queries (because of different user needs), possibly even on raw data and *returns more intuitive results to users*.
11. A PBMS employs a query language which can at least perform the following tasks:
    - *Pattern matching*, meaning the mapping of a certain (new) input to a set of patterns already found in the PBMS and its reverse operation, which leads from the pattern towards the raw data that correspond to it
    - *Deductive capabilities*, meaning that logical inferences on the basis of the pattern representation are one of the desiderata for the query language. This can involve several operations, such as *zoom in/out* over patterns that can be composed with part-of relationships (i.e., a pattern that corresponds to a large set of data is possibly related to a pattern that corresponds to a subset of these data), *pattern composition* of patterns, which are candidates to compose a complex pattern, *roll-up and drill-down* over multidimensional hierarchies of features, etc.
    - *Meta querying facilities*, enabling the possibility of querying the composing elements of a pattern base (e.g., types, classes, etc.) in order to extract meaningful conclusions and to enable the administration of the PBMS
12. A PBMS is useful in order to process those queries *more efficiently than a normal DBMS* would do.

We tend to believe that the PANDA approach is not yet another object-oriented or object-relational variant. Although it might look quite close due to the class/type system, the two approaches are intrinsically different:

- In terms of the model, the discriminating feature is the requirement for a semantically rich representation of patterns, achieved by separating structure and measure on the one hand, by introducing the expression component on the other.
- From the functional point of view, instead of pointer-chasing object-oriented queries, novel querying requirements arise, including (a) ad hoc operations over the source and pattern spaces and their mapping, (b) pattern matching tests,

facilitated from the separation of structure and measure, and (c) reasoning facilities based on the expression component of patterns.

– In terms of system architecture, the relevance of queries aimed at evaluating similarity between patterns and the request for efficiency call for alternative storage and query optimization techniques.

6. Conclusions and Future Work

In this paper, we have dealt with the introduction of the mathematical foundations and the architecture for pattern management. First, we introduced the intuition and mathematical foundations for pattern management. Next, we presented the architecture of a *Pattern Base Management System* that can be used to efficiently store and query patterns. Finally, we commented upon the criteria and characteristics of PBMS's and on whether the existing systems for storing and manipulating data can cover the special user requirements that patterns impose.

Clearly, under this general framework, the PANDA consortium has several interesting tasks to accomplish. From our part, our future research includes primarily theoretical aspects like the introduction of a powerful language for expressing pattern semantics, the possibility of contributing to the model of [Riz+03] e.g., by introducing constraints on patterns; and finally by contributing to a flexible query language and its respective algebra for retrieving and comparing complex patterns.

References

[BeLi96] Michael Berry, Gordon Linoff. "Data Mining Techniques: For Marketing, Sales, and Customer Support". John Willey, 1996.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96): pp. 226-231, 1996

[FPSU96] U. Fayyad, G. Piatesky-Shapiro, P. Smuth & R. Uthurusamy (eds). "From Data Mining to Knowledge Discovery: An Overview". Advances in Knowledge Discovery and Data Mining. AAAI Press, 1996.

[HaKa01] Jiawei Han, Micheline Kamber. "Data Mining: Concepts and Techniques". Academic Press, 2001

[Riz+03] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, E. Vrachnos. Towards a logical model for patterns. *Submitted for publication.*

[VaHT02] M. Vazirgiannis, M. Halkidi, G. Tsatsaronis. The logical model of a Pattern Management System (The AUEB viewpoint). PANDA Internal Report PANDA -- AUEB – 001. June 2002.

[Vaz+02] M. Vazirgiannis, M. Halkidi, G. Tsatsaronis, E. Vraxnos, D. Keim, P. Xeros, Y. Panagis, A. Pikrakis (UoA). Related Research. PANDA Internal Report, PANDA – AUEB, KONSTANZ, UoA, CTI-- <001> . October 2002.