An Overview of Database Mining Techniques

Bob Kero   Lucian Russell   Shalom Tsur

Argonne National Laboratory
Decision and Information Sciences Division
Argonne, Illinois, 60439
{bkero, russell, tsur}@dis.anl.gov

Wei-Min Shen

USC Information Science Insitute
4676 Admirality Way
Marina del Rey, CA., 90292
shen@isi.edu

Abstract

This paper presents an overview of data mining techniques in the context of of the deductive-inductive cycle model. The claim made in this paper is that a combination of both of these modes of inference in one model offers adavantages over the exclusive use of either induction or deduction. After an introduction of the model, the paper describes an architecture of the Knowledge Miner environment and its relation to the $\mathcal{LDL}$++ deductive database technology. In particular, it defines the notion of a *meta query* and shows how this concept can be used to take advantage of the full power of a deductive data base system. The paper mentions a number of inductive methods that can be used within this framework and describes some data mining experiments performed with the Knowledge Miner.

1 Introduction

Very large collections of data often serve as a corporate resource—to be used for a variety of purposes, many of which are unknown or unspecified at the time of collection. The term Database Mining[1] is used to denote a class of technologies, all designed to assist in the discovery or the verification of commercially valuable knowledge in this vast corporate data resource.

There is an increasing interest in the commercial world for database mining methods:

> "The real breakthroughs will come from the ability to 'mine' information—the process of quickly gathering and analyzing the millions of bits of data a business generates each day—and steer various pieces to the right people in the organization."
>
> The Information Revolution, *Business Week*, July 1994, pp. 104.

And more on the business aspects of this technology can be found e.g., in [Ber94, et al.]. Examples of data mining applications include:

- The analysis of airline seat reservation data for patterns of commercial value.
- Chip production data for the identification of flaws in the manufacturing process.
- Communication network management data for anomalous behavior, e.g., the "convoy" problem [NT89]
- Scientific data analysis; e.g., electrophoresis gel data for DNA sequence identification.
- Analysis of retail and other marketing data for timely identification of consumer trends. We will elaborate on applications of this type in the sequel.
- Data base schema integration. In many real-life situations the nature of the data in a data base are only partially understood. It is e.g., unclear to what extent the data actually adhere to their schema definition or whether certain integrity constraints such as keyness of attributes or referential integrity hold within relations or among different relations. Data mining techniques are useful in

[1] The literature also refers to Database mining as "Data Dredging," "Knowledge Mining," or "Knowledge Discovery" and an attempt is sometimes made to distinguish between these activities. For our purposes however we will regard these terms as synonymous.

the elucidation of these constraints and are used to supplement the (often partial) information given in the data description.

Different methods can be applied to the process of Data Mining: statistical methods, neural networks, genetic algorithms and others. We will not cover this multitude here but concentrate on those methods that produce an *explicit representation of the knowledge gained.* In contrast, the knowledge gained using other methods is often *implicit* e.g., using neural networks. It is beyond the scope of this review to compare the merits of these different approaches. Among the explicit methods, we distinguish between *inductive* and *deductive* methods. Inductive methods derive their knowledge from examples: a test or training set of the data while the deductive methods, in contrast, derive their knowledge from the explicit statement of rules representing a high level concept or hypothesis that need to be verified using the data at hand. Such hypotheses are formulated by human experts. We will argue that the most efficient data mining methods combine both approaches in a seamless fashion and will sketch the architecture of an environment for data mining that allows for this combined approach.

In classifying the problems of interest we examine them from three different aspects:

1. Strategy used.

2. Representation of discovered knowledge.

3. Application domain.

The strategy used will be outlined in the next section in which we intoduce a model for data mining. The discovered knowledge is kept in a knowledge base in the form of meta patterns. This form of knowledge representation will be discussed in Section 4. The methods used are generic and are thus applicable to many different domains.

2 A Model for Data Mining

Database mining is an evolving technology: a point of confluence of ideas developed in other contexts; mostly in machine learning, database systems and statistics. One of our objectives is to present these ideas in a uniform and consistent fashion. We will develop therefore a model of the database mining process that can be used for this purpose.

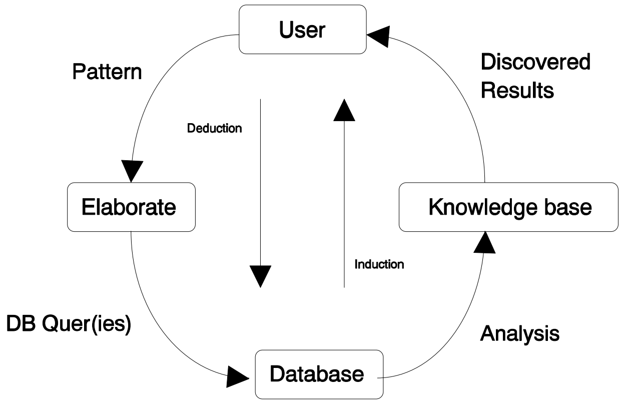

Figure 1: Deductive-inductive cycle for knowledge discovery in databases

Matheus *et al* [MCPS93] describe a generic architecture for Knowledge Discovery in Databases. In their scheme, high-level patterns are translated to database queries which are executed and return results. Subsequently, the results are focused, analyzed for existing patterns, evaluated and returned as discoveries. Each of these stages is supported by a knowledge base containing domain specific information pertinent to the discovery process and the system is governed by a controller that decides when and how to invoke each of its components. The basic tenet of this architecture is that the discovery process in databases is a *sequential, inductive flow:* from the data, via the stages described to discovered results. In contrast, the architecture that we present in this paper assumes that an efficient discovery process combines both inductive and deductive components. Essentially, we believe that the discovery process is an iteration between high-level, user-specified patterns their (deductive) elaboration to database queries, the (inductive) analysis of the queries *which result in modified knowledge that will be used in the next cycle* and so on, until the satisfactory discovery of results. This idea is depicted in Fig. 1.

The model we promote in this paper is the happy marriage of two branches of research:

the "Data Dredging' idea presented in [Tsu90][2] and the Machine Learning approach presented in [She90, She92]. The former offers the later a deductive method to access real databases for verifying hypotheses, while the later offers the former the idea of a discovery loop that involves deduction, induction, and human intuition, and a set of inductive data analysis tools. In contrast, in the Data Dredging model, the inductive component was absent and was subsumed by the user, who observed the deductive results, (intuitively) derived the new knowledge, and modified the query part during the next cycle. In the sequel we will elaborate on each of the components of this model, present a system that is built along these lines and discuss the relation between this system and deductive database technology. We will discuss various options for the inductive analysis of the results and conclude with some ideas for further research in this area.

3 Deductive database systems

Deductive database systems are database management systems whose query languages and storage structure are designed around a logical model of data. These systems permit the expression of queries both on *intensional* (i.e., stored) data, by means of the declared schema, as one would do using SQL, as well as on *extensional* data by means of declared rules. The technology can thus be thought of as an extension to the existing relational database technology that increases the power of the query language. Among other features, the rule-based extensions permit the specification of queries using recursive rules, the classical example which is that of the *ancestor* relation:

$$\begin{aligned} ancestor(X,Y) &\leftarrow parent(X,Y). \\ ancestor(X,Y) &\leftarrow ancestor(X,Z), parent(Z,Y). \end{aligned}$$

Where $parent(X,Y)$ is the predicate matching the stored data relation in the database. A great deal of research effort went into the design of efficient compilation and optimization techniques for the transformation of the extensional queries into

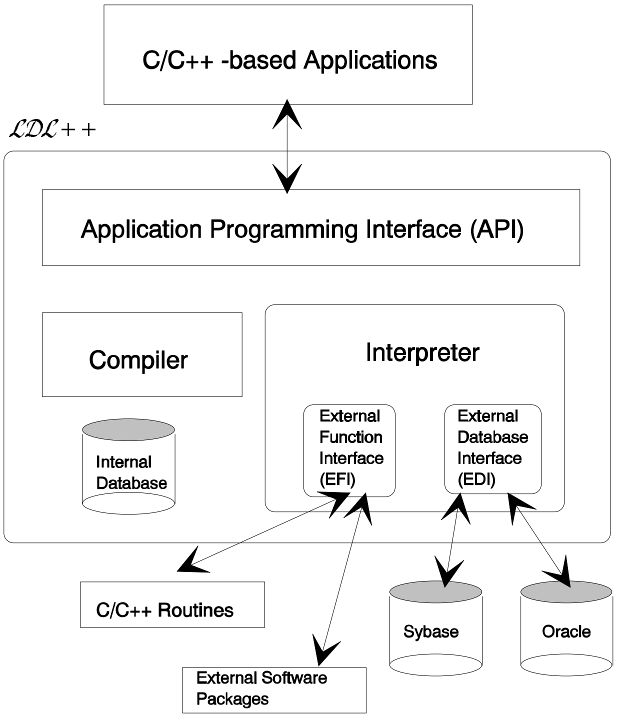

Figure 2: $\mathcal{LDL}$++ Open Architecture

efficient plans for accessing the stored data. We will assume a certain degree of familiarity with Deductive Database technology. For an introduction see [Ull88, NT89], for an overview of the main issues in this area and an account of existing research prototypes see [RU93].

Although the technological developments that led to deductive database systems were not motivated by the need for database mining, the declarative nature of the technology, that permits to specify *what* is required, rather than *how* to obtain it and leaves the elaboration of the intention to the system itself, has proven to be a powerful tool in this area. In the next section we will provide a brief overview of the $\mathcal{LDL}$++ system, which is used to implement the deductive (elaboration and database) part of the Knowledge Miner—a Database Mining Environment [SOZ94, SOZ95], that will be described in the sequel.

3.1 The $\mathcal{LDL}$++ Deductive Database System

The
$\mathcal{LDL}$++ Deductive Database System [AOTZ], is the successor to the $\mathcal{LDL}$ system that was built at MCC, Austin, TX. In its design, the system has

[2] The term "Data Dredging" never gained any wide acceptance. We have replaced it therefore with the more commonplace term "Data Mining."

benefited extensively from the experience gained with the previous system and its architecture is depicted in Fig. 2.

Among other lessons that were implemented in this system is the reliance on an *open architecture*: the ability to interface and combine the use of $\mathcal{LDL}$++ with a variety of external systems an components. Interaction with the exterior is achieved in three ways:

- **Application Programming Interface (API).** The API is a programmable interface that allows for the embedding of an $\mathcal{LDL}$++ rule set and schema declaration within a C or C++ program. Thus, it is possible for example to use a customized windows-based interface for the interaction with the system. One version of such an interface exists, which was created using the HTML mark up language. The default interface is line-oriented.

- **External Function Interface (EFI).** The EFI is a set of function that allow for the inclusion of C/C++ functions as predicates within the $\mathcal{LDL}$++ rules. This capability enables the use of specialized software such as packages for statistical analysis or the use of visualization software without having to recreate this software. It is also a convenient method for the implementation of inductive data analysis routines, such as Bayesian clustering, in this context.

- **External Database Interface (EDI).** $\mathcal{LDL}$++ has its own internal, memory-resident database. The EDI allows for access to external database server such as Sybase, Oracle and DB2. External relations are declared at the $\mathcal{LDL}$++ schema level and predicates referring to these relations are compiled into SQL queries that are executed on the servers containing the data. There is a considerable amount of effort invested in the efficient optimization of external queries. Thus, rules are transformed when possible, to move relational joins from the client side to the server side, aggregate operation on sets are performed on the server side and only the results are sent to the client and some other optimization operations are employed as well.

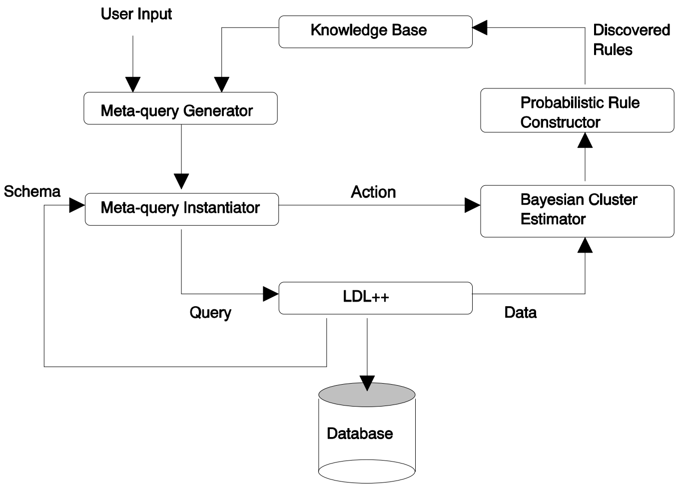

Figure 3: Architecture of Knowledge Miner

Although, as we mentioned, this system was not developed for data mining, experience has shown that it is almost ideally suited for this purpose. The open architecture allows for a simple integration of the declarative components and the inductive components which are typically implemented in C or C++. It also allows for the interchangeability of different inductive analysis routines which is very important in a knowledge mining environment.

4 Knowledge Miner

Knowledge Miner is an environment for data mining from databases that is built in accordance with the discovery cycle described earlier. The present version is built on the $\mathcal{LDL}$++ architecture and Fig. 3 depicts the main components of this system.

The user pattern and elaboration components are implemented by means of *Meta-queries*, described in section 4.1; the results of the database queries posted by the instantiation of a meta query are analyzed by a Bayesian Cluster Estimator and transformed into new meta-query forms that are stored in the knowledge base. The knowledge base can be used in the subsequent discovery cycles and the process repeats until the user deems the knowledge to be a discovery.

4.1 Notion of Meta Query

Meta-queries are the method for the description of patterns required in a data mining operation. They can be regarded both as a (deductive) restriction[3] that, when instantiated, induces a set of queries on the underlying data and in addition causes some *inductive action* on the results returned.

An example of a Meta Query is the form:

$$P(X,Y) \wedge Q(Y,Z) \Rightarrow R(X,Z)$$

where P, Q and R are meta-relations that are instantiated to actual relations, described by the schema of the underlying database. Thus if $citizen, officiallanguage$ and $speaks$ are declared within the underlying database schema then the form can be instantiated to:

$$citizen(X,Y) \wedge offciallanguage(Y,Z) \Rightarrow speaks(X,Z)$$

In the general case, the mapping from meta-query form to an instantiated form can be controlled by additional mapping information which is kept as part of the Knowledge Base. For example, with each of the Meta predicates P, Q and R, we can associate a set of underlying relations into which they can be mapped, thus resulting in a set of instantiated forms.

4.2 Meaning of an instantiated form

The instantiated form means that if X is a citizen of country Y and the official language of country Y is Z then X speaks Z. Note that the "$\Rightarrow$" is not an implication in the propositional sense but is a *plausible deduction* that may be true in only a subset of all cases. Thus, with each instantiated form, there is an associated *strength* which is an indication of the relative number of the cases in which the deduction is true. We can use a probability to denote the strength of the instantiated form if we know the distribution functions of the values for each of the participating variables.

Let $\mathcal{F}$ denote the instantiated form $p(X,Y) \wedge q(Y,Z) \Rightarrow r(X,Z)$, and let $\mid r(X,Y) \mid$ denote the cardinality i.e., the number of distinct tuples for which $r(X,Y)$ is true. Assuming uniform distributions of values for each of the variables, the strength of $\mathcal{F}$ is:

$$strength(\mathcal{F}) = \begin{cases} 1 \\ \mid r(X,Z) \mid / \mid lefthand(X,Z) \mid \end{cases}$$

if resepctively,

$$lefthand(X,Z) \subseteq r(X,Z)$$
$$lefthand(X,Z) \supset r(X,Z)$$

where $lefthand(X,Z) = p(X,Y) \wedge q(Y,Z)$.

4.3 General Case

In the general case, an instantiated meta-query is an object

$$MQ(\mathcal{F}, result(\mathcal{F}), strength(\mathcal{F}))$$

where:

- $\mathcal{F}$ is an instantiated meta query form.
- $result(\mathcal{F})$ is its result.
- $strength(\mathcal{F})$ is its strength.

The instantiated $\mathcal{F}$ form is as:

$$a_1 \wedge a_2 \wedge \ldots \wedge a_m \Rightarrow c_1 \wedge c_2 \wedge \ldots \wedge c_n$$

where $a_1 \wedge a_2 \wedge \ldots \wedge a_m$ is the *antecedent* of the form and $c_1 \wedge c_2 \wedge \ldots \wedge c_n$ is its *consequent*. Each of the predicates in either the antecedent or the consequent may be derived from the base relations of the underlying schema. Thus, the full power of $\mathcal{LDL}++$ can be utilized in the computation of plausible deductions.

Using standard $\mathcal{LDL}$++ notation, let $lefthand(L) \leftarrow a_1, a_2, \ldots, a_m$ where L is a subset of variables appearing in the antecedent and let $righthand(R) \leftarrow c_1, c_2, \ldots, c_n$ where R is a subset of variables appearing in the consequent. Then, the result of the application of the instantiated form $\mathcal{F}$ is given by:

$$result(\mathcal{F}) = \{LR \mid lefthand(L) \wedge righthand(R)\}$$

[3] This function is sometimes referred to in the literature as the "focus"

and its strength by:

$$strength(\mathcal{F}) = \begin{cases} 1 \\ \mid righthand(R) \mid / \mid lefthand(L) \mid \end{cases}$$

if resepctively,

$$lefthand(L) \subseteq righthand(R)$$
$$lefthand(L) \supset righthand(R)$$

Note that the particular example used in section 4.1 does in fact use $\mathcal{LDL}++$ in only a limited way and the result can be derived by computing

$$\Pi_{(X,Z)}[citizen(X,Y) \bowtie officiallanguage(Y,Z) \bowtie speaks(X,Z)]$$

Furthermore, note e.g., that for two forms, $\mathcal{F}_1 : a, b \Rightarrow c$ and $\mathcal{F}_2 : a \Rightarrow b, c$; $result(\mathcal{F}_1) = result(\mathcal{F}_2)$ *but their strengths may be different.*

Example *Use of Recursion in a Meta-Query.*

An example employing the recursive query capability of $\mathcal{LDL}$++ would be:

$$ancestor(X,Y) \wedge age(Y,Z) \Rightarrow Z > 13.$$

i.e., assuming that we are dealing with humans, "if Y is an ancestor of X then plausibly, the the age of Y is at least 13." In this example the strength would vary of course depending on the threshold age chosen.

5 Inductive Data Analysis

A vareity of methods exist to perform the inductive part of the the inductive-deductive cycle described in this paper. The choice of the method depends on the purpose of the data mining operation. Among the more common methods employed we list the following:

- *Classification*: partitioning the result of the deductive query into subsets; see e.g., Quinlan [Qui86].
- *Abstraction*: deriving higher-level concepts from the results of the deductive query; see Han *et al* [HCC93] and Dhar and Tuzhilin [DT93].

A third method is Bayesian Clustering, which is employed in the Knowledge Miner. We elaborate on this method in the next section.

5.1 Bayesian Clustering

An example of Bayesian Clustering [SOZ95] is that of data mining in a large database of chemical compounds, their ingredients and their physical properties. The specified a meta form is:

$$Ingredients(X, c_1, c_2) \Rightarrow Property(X,Y), cluster(Y,S)$$

i.e., for compound X the presence of ingredients c_1 and c_2 implies that X possesses property Y and that this property is clustered around certain values. Note that the strength of this deduction is 1 i.e., compound X always possesses *some* property Y. Property values of Y are numerical and the purpose of the Bayesian Clustering action is to ascertain whether these are randomly distributed or do cluster around particular values.

Ingredients and properties of a compound can, in general, be distributed among a large number of different tables in the database all of which need to be instantiated by the meta form. In this example we assume that the underlying schema consists of 4 tables: *ingredients1(X,Y), ingredients2(X,Y), elasticity(X,Y)* and *density(X,Y)* where $ingredients1$ and $ingredients2$ contain compounds X with ingredients Y and *elasticity* and *density* contain respectively two different properties of compound X with value Y.

The meta form, instantiated on the underlying schema, expands into:

$$ingredients1(X, c_1), ingredients2(X, c_2) \Rightarrow elasticity(X,Y), cluster(Y,S)$$

$$ingredients1(X, c_1), ingredients2(X, c_2) \Rightarrow density(X,Y), cluster(Y,S)$$

The result of these instances would be (c_1, c_2, Y) i.e., a set of property values Y for the elasticity and strength properties respectively, for all compounds containing ingredients c_1 and c_2.

The action on this result attempts to cluster the property values using Bayesian cluster analysis. Thus, $cluster(Y,S)$ is a function on the result, returning a set S of clusters for the values of property Y. Each cluster is described by a triplet (m, v, l) where m is the mean value, v is the variance, and l is the likelihood of the class. In the experiment described in [SOZ95],

$$cluster((3060, X, density)) \rightarrow \\ \{(1.243, 0.46, 0.99), (-999.9, 0.2, 0.01)\}$$

i.e., the presence of ingredient 3060, and any other ingredient in a compound implies that the density property of the compound is clustered around two median values 1.243 and 0.2. The later cluster is probably an indication of erroneous data in the database. A similar rule can be derived if the meta form would be instantiated on the other property table.

6 The Knowledge Base

The knowledge base accumulates the knowledge that is gathered during the inductive data analysis. In addition, it stores knowledge that is required for the elaboration of meta queries into instantiated forms. For example, a meta predicate used in the Bayesian Clustering example (sec. 5.4) is of the form $Ingredients(X, c_1, c_2)$. The knowledge base includes a syntactic transformation:

$$Ingredients(X, Y, Z) \rightarrow \\ Ingredients(X, Y), Ingredients(X, Z).$$

that is used to elaborate on *Ingredietns* prior to its assignment to schema relations. Other knowledge is contained to speed up the process by restrictions on the elaboration process. For example, instead of letting *Ingredients* "roam" over all of the underlying schema relations, its elaboration is restricted to particular underlying relations and likewise, the elaboration of *Property*. Probabilistic rules, such as the ones derived from the chemical database using Bayesian clustering are also included in the Knowledge base and are used for the subsequent deductive elaboration. At present the utilization of the accumulated knowledge is the responsibility of the user An open issue is that of the autonomy or the degree of supervision require and the feasibility of the automatic inclusion of the accumulated knowledge in the process. Experience so far shows however that the user, who ha the ability to observe the results is often the best judge and we expect that he/she will always be "in the loop" to supervise and control th discovery process. The importance of a sophisticated tool for th visualization of the results has proved to be invaluable and it will b an integral component of the Knowledge Miner environment.

7 Conclusion

In this paper we presented an overview of techniques for data mining. W discussed these in the context of a model for data mining and its implementation in form of the Knowledge Miner environment. This system which is built on the open architecture of the $\mathcal{LDL}$++ deductive database has been tested so far in a number of different enterprises such as th telecom industry, the chemical industry and recently, in the semiconductor manufacturing industry. Major issues that are still open concern the efficiency of the computation of meta form instantiations and their strengths. The recent trend to move enterprise management problems to parallel computer systems, both of the SMP and MPP classes, is of importance to these issues as well an we expect to adapt the technology reported here to these new architectures. Other issues concern the integration of this technology with other enterprise management solutions. In particular data warehousing and schema integration. Lastly, the underlying data domain used so far is relational Vast source of untapped knowledge reside in other forms of data, in particular unstructured text and semi-structured sources such as SGML marked-up documents. We plan to extend the technology to cover these sources as well.

Acknowledgment

The authors wish to thank Avi Silberschatz for the careful reading and thoughtful commentary on an earlier version of this paper.

References

[AOTZ] N. Arni, K. Ong, S. Tsur, and C. Zaniolo. $\mathcal{LDL}$++ : A second-generation deductive database system. *ACM Transactions on Database Systems (TODS)*. Submitted for publication.

[Ber94] J. Berry. Database Marketing. *Business Week*, September 1994.

[DT93] V. Dhar and A. Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):926–938, December 1993.

[HCC93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29–40, February 1993.

[MCPS93] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. Systems for Knowledge Discovery in Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):903–913, December 1993.

[NT89] S. Naqvi and S. Tsur. *A Logical Language for Data and Knowledge Bases.* Computer Science Press, 1989.

[Qui86] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[RU93] R. Ramakrishnan and J.D. Ullman. A Survey of Deductive Database Systems. *Jornal of Logic Programming*, 12:1–26, 1993.

[She90] Wei-Min Shen. Machine learning with the cyc knowledge base. Technical Report AI-90-17, Microelectronics and Computer Technology Corp. (MCC), Austin, TX., 1990.

[She92] Wei-Min Shen. Discovering regularities from knowledge bases. *Journal of Intelligent Information Systems*, 7:623–635, 1992.

[SOZ94] W. Shen, K. Ong, and B. Mitbander C. Zaniolo. Using metaqueries to integrate inductive learning and deductive database technology. In *AAAI Workshop on Knowledge Discovery from Databases*, 1994.

[SOZ95] W. Shen, K. Ong, and B. Mitbander C. Zaniolo. *Metaqueries for Data Mining*, chapter 15. Advances in Knowledge Discovery and Data Mining. MIT Press, 1995.

[Tsu90] Shalom Tsur. Data dredging. *IEEE Data Engineering Bulletin*, 13(4):58–63, December 1990.

[Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 2. Computer Science Press, 1988.