Knowledge Discovery in Data Warehouses

Themistoklis Palpanas
themis@cs.toronto.edu
Department of Computer Science
University of Toronto
10 King's College Road, Toronto
Ontario, M5S 3G4, CANADA

Abstract

As the size of data warehouses increase to several hundreds of gigabytes or terabytes, the need for methods and tools that will automate the process of knowledge extraction, or guide the user to subsets of the dataset that are of particular interest, is becoming prominent. In this survey paper we explore the problem of identifying and extracting interesting knowledge from large collections of data residing in data warehouses, by using data mining techniques. Such techniques have the ability to identify patterns and build succinct models to describe the data. These models can also be used to achieve summarization and approximation. We review the associated work in the OLAP, data mining, and approximate query answering literature. We discuss the need for the traditional data mining techniques to adapt, and accommodate the specific characteristics of OLAP systems. We also examine the notion of *interestingness* of data, as a tool to guide the analysis process. We describe methods that have been proposed in the literature for determining what is interesting to the user and what is not, and how these approaches can be incorporated in the data mining algorithms.

1 Introduction

During the past few years we have experienced a considerable growth in the amount of data produced and managed by different organizations worldwide. This growth led in many cases to the introduction of multiple database systems within the same organization in order to deal with different aspects of the data [CCS93]. Nevertheless, the poor data analysis functionality that the traditional databases offered was the incentive for the advent and development of *data warehouse* systems. These systems store consolidated, historical, and summarized data, and are designed to support complex, mostly ad hoc queries, which involve large portions of the stored data.

Typically, data warehouses use multidimensional models in order to effectively represent the wealth of information they manage [CD97]. The data is organized in dimensions which describe in a natural way most of the attributes associated with the data of interest. The dimensions are in turn organized into hierarchies, with data aggregated at each level, which enhances the functionality of the system. The operations that the system supports include increasing and decreasing the level of aggregation along any number of dimension hierarchies, selection and projection across dimensions and hierarchy levels, and defining various orientations of the multidimensional view of the data. The above kind of technology is referred to as *On Line Analytical Processing* (OLAP), and we will discuss it further in the next sections.

Data warehouses tend to be fairly big in size, usually orders of magnitude larger than operational databases. This happens because the warehouse gathers information from all the data sources within an organization, and in addition, this information is stored for long periods of time. Undoubtedly, the capabilities for data analysis and knowledge extraction are considerable in such an environment. This is due mainly to two reasons. First, the data structures used in data warehouses are specifically designed to support decision support. Thus, they can be exploited in the knowledge extraction process. Second, data warehouses manage the aggregated knowledge of an organization, in the sense that they consolidate data from many sources and therefore store a richer data collection than any other database in the organization. Nevertheless, as the size of the dataset increases to several hundreds of gigabytes or terabytes, all the analysis techniques performed manually become extremely cumbersome, and inefficient. Therefore, the need for methods and tools that will automate the process of knowledge extraction, or guide the user to subsets of the dataset that are of particular interest, is becoming prominent.

While numerous traditional data mining techniques have been discussed with respect to the OLAP context [Han98], in this paper we take a step further. We argue that mining algorithms have to get a better understanding of the OLAP environment in order to harness its full power, and become more involved in order to meet the analyst's requirements. We review different ways of identifying and extracting knowledge from large collections of data. We examine the notion of *interestingness* of data, that is, ways of determining what is interesting to the user and what is not. Subsequently, we explore algorithms and techniques that operate on large datasets and find the interesting portions therein. Note, that this survey does not intend to be exhaustive, but rather indicative of the different approaches, and the future research directions in the area of data mining in data warehouses.

1.1 Paper Outline

The outline of the paper is as follows. We give an overview of the OLAP technologies in Section 2. In Section 3 we briefly discuss data mining with emphasis on association rules, and we explore the notion of interestingness. A review of data approximation techniques follows in Section 4. We also comment on the relation of these techniques to data mining. In Section 5 we investigate the specific problem of data mining in data warehouses. Finally, we conclude in Section 6.

2 OLAP Technology

2.1 Overview

The standard logical model for representing the data in a data warehouse is the *data cube*. The data cube is a multidimensional view of all the data, which are represented as a set of numeric measures (or facts), and organized in several dimensions of interest [CD97]. The numeric measures are the values of the data we are interested in, and the ones the analysis will be performed on. The dimensions are attributes of the data. They provide the context for the measures, and are organized in hierarchies of multiple levels. The measures are aggregated at each hierarchy level for each dimension to offer a more general view of the base data. Figure 1 depicts an example of a data cube. In this example the measures can be

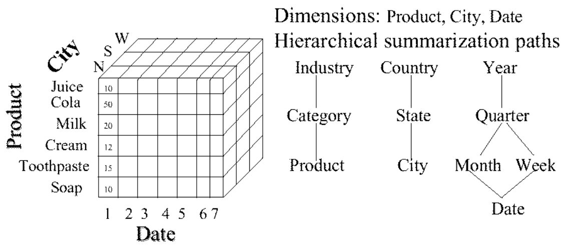
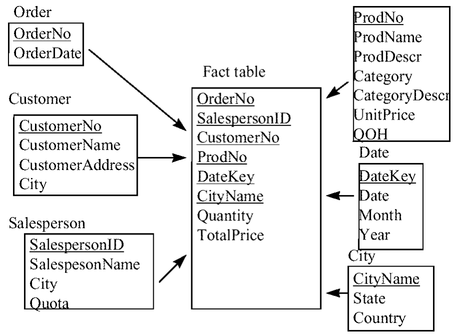


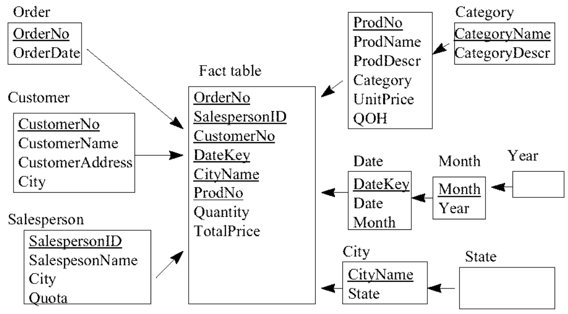
Figure 1: Example of a data cube.

quantities such as total sales, revenue, and inventory.

The physical design of the aforementioned data model is different for *Multidimensional-* and *Relational*-OLAP servers (MOLAP and ROLAP respectively). MOLAP servers directly support a multidimensional view of the data, achieved through a multidimensional storage engine. This approach results in more natural ways of data representation, but requires special care when storing data since data sets in high dimensions tend to be sparse. On the other hand, ROLAP servers take advantage of the existing relational database technology. The database consists of a fact table, storing all the measures, and dimensional tables around it. Having one table per dimension leads to the *star* schema, and by normalizing the dimension tables we get the *snowflake* schema (Figure 2).



(a) Star Schema



(b) Snowflake Schema

Figure 2: Two different approaches in the physical design of ROLAP servers.

2.2 The Data Cube Operator

The problem of formalizing the computation of a data cube is discussed in [GBLP96]. This study focuses on the relational database systems, and addresses the need for an additional operator. The authors propose the *data cube* or *CUBE* operator, that computes the $N-$dimensional aggregates of a dataset with N aggregation attributes. This is a generalization of the SQL aggregation functions and the *GROUP BY* operator, which produces $0-$ and $1-$dimensional answers respectively. The *CUBE* operator may be simulated by a union of 2^N *GROUP BY*s for N aggregation attributes. These *GROUP BY*s are depicted using the *lattice* notation [HRU96] (Figure 3), where each node represents a *GROUP BY* (also called *cuboid*). Nevertheless, this approach is much more cumbersome and hard to write. Furthermore, the final query would result in an equally large number of data scans and sorts, which is obviously inefficient. The *CUBE* operator allows the user

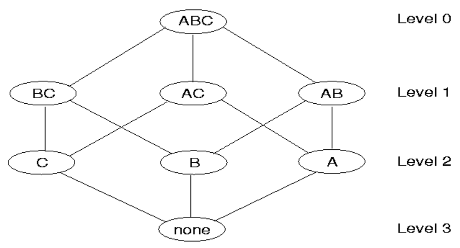


Figure 3: The lattice notation for a data cube with three dimensional attributes.

to move between different levels of aggregation of the data by removing or adding aggregation attributes. Going up the levels is called a *roll-up*, while the opposite operation, going down the levels, is termed a *drill-down*. This study presents a classification of the aggregation functions into three categories: *distributive*, which can be computed separately for disjoint subsets of the dataset and then the partial results are merged (e.g., count, sum, min, max); *algebraic*, that can be expressed using some of the distributive function (e.g., average); and *holistic*, which can only be computed for the whole dataset (e.g., median, rank).

Several algorithms have been proposed for the efficient computation of the data cube [AAD+96] [RS97] [ZDN97]. An overview of various indexing schemes used in OLAP databases is presented by Sarawagi [SS94] [Sar97], and a novel approach is described by Roussopoulos, Kotidis et al. [RKR97]. The problem of maintaining a data cube after we have computed it is also an interesting research direction [MQM97] [KR99].

3 Data Mining

3.1 Overview

The enormous collections of data now available by numerous organizations render the need for automatic methods of intelligent data analysis imperative. The framework that describes all the necessary steps involved in the process of knowledge extraction from databases is referred to as *Knowledge Discovery in Databases* (KDD). KDD is the process of identifying valid, novel, potentially useful, and ultimately understandable structures in data [Fay98]. In the above definition, data is the set of facts in the database, and structure refers to either a parsimonious description of a subset of the data, or a model representing the source that generated the data. The KDD process is comprised of many steps which involve data preparation, search for structures, knowledge evaluation, refinement, and consolidation with the existing expert knowledge. The term *data mining* refers to only one of the aforementioned steps. It is the step in the KDD process that, under acceptable computational efficiency limitations and selective evaluation criteria, enumerates structures over the data.

In the next section we will focus on the literature of association rule mining, since it is the topic that propelled a great deal of work in the database community. For a more complete discussion of the data mining techniques the interested reader should refer to a survey of the area [CHY96].

3.2 Association Rules Mining

Mining association rules is one form of data mining that attracted a lot of attention during the last years. It is essentially the search for dependencies which hold among items in a large dataset. The problem was first formulated by Agrawal et al. [AIS93]. Let $\mathcal{I}= \{i_1, i_2, \ldots, i_m\}$ be a set of items, and $\mathcal{D}$ a set of transactions, where each transaction T in $\mathcal{D}$ is a subset of $\mathcal{I}$. Let X, Y be sets of items. A transaction T *contains* X if and only if $X \subseteq T$. An association rule is an implication of the form $X \rightarrow Y$ (if X then Y), where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$ and $X \cap Y = \emptyset$. Each rule is associated with two figures. The *confidence* c, which is the percentage of transactions in D containing X that also contain Y, and the *support* s, which is the percentage of transactions in D that contain $X \cup Y$.

The association rule mining algorithms decompose the problem into two phases. First, they find all the sets of items (*itemsets*) that have transaction support above minimum support. These itemsets are called *frequent* itemsets. Second, they use the frequent itemsets to derive the association rules that hold on the database. Note that it is the first step that dominates the total execution time, and especially its early stages. The first algorithm that efficiently computed association rules from large data sets is *APRIORI* [AS94]. Apriori makes consecutive passes over the data. In the k-th pass it constructs the candidate set C_k of frequent items *k-itemset*. These itemsets have cardinality k, and are derived from the frequent itemsets of the previous pass, since any subset of a frequent itemset is frequent as well (the very first pass determines C_1). Then, the algorithm calculates the support of the items in C_k to determine which of them have minimum support. The ones that are above the threshold form the frequent itemset L_k, which is the basis for producing C_{k+1}. This process stops when no new frequent itemsets are discovered. In order to generate C_{k+1}, Apriori uses L_k. It first joins L_k with itself, and then prunes these itemsets $c \in C_{k+1}$ that contain a k-subset not belonging in L_k. The *APRIORITID* algorithm [AS94] tries to reduce the database passes, by computing in memory all the frequent itemsets L_i for $i > 1$ after the initial pass. At the end, one more pass is required in order to determine which of the produced itemsets are actually frequent. This approach though, is only feasible when all the intermediate results fit in memory.

Several variations of the above algorithms try to minimize the required I/O operations. These include *Direct Hashing and Pruning* [PCY95], the *PARTITION* algorithm [SON95], a technique incorporating sampling [Toi96], and the *Dynamic Itemset Counting* algorithm [BMUT97]. The association rule mining algorithms were also extended in different ways. *APRIORI* was generalized to take into account the hierarchies that are present on items [SA95], while another extension of the algorithm allowed for the discovery of frequent *sequential patterns* [AS95] [SA96], where the order of transactions is taken into account as well.

3.3 Interestingness of Rules

Knowledge discovery tries to solve the problem of data analysis by identifying interesting patterns in huge collections of data. Essentially, this is a process of reducing and condensing the knowledge implicit in the data, so that the effort of the analyst is alleviated. Nevertheless, discovery systems are generating a wealth of patterns, most of which are of no interest to the user [FPSM91]. Therefore, a measure that captures the amount of information that a discovered pattern conveys is essential.

Measures of this kind are termed measures of *interestingness*, and can be classified into *objective* and *subjective* [ST96b]. The objective measures of interestingness appraise a pattern in terms of its structure and the underlying data used in the discovery process. Examples of such a measure are the support and confidence thresholds [AIS93], which are extensively used in association rule mining. These measures are defined over a rule $X \rightarrow Y$ as functions of the probabilities $p(X)$, $p(Y)$, and $p(X \wedge Y)$. Objective measures can substantially reduce the number of produced patterns, and only report the most "strong" ones. However, it is often the case that these measures of interestingness do not capture all the complexities of the pattern discovery process. A reported pattern may correspond to prior knowledge or expectations. It may refer to uninteresting attributes or attribute combinations. It may also be redundant if it is similar to or contained in another pattern. Thus, the analyst has to go through a large number of results, only a small subset of which is of real interest to her. The subjective measures of interestingness try to remedy this situation. They do not depend solely on the data they operate on, but also on the user who examines the patterns. The prior knowledge of the user is incorporated in the discovery process, and only the patterns that are interesting to the specific user are reported.

In the following paragraphs we will focus on subjective measures of interestingness. We will review some of the existing approaches for uncovering interesting rules, which can be classified in two categories. The *template-based* techniques require explicit input from the analyst, while the *model-based* methods are only loosely-coupled to user input.

3.3.1 Template-Based Rule Discovery

One way of making sure that the system captures and reports patterns that are interesting to the user, is to instruct the system what to look for prior to the discovery process. Klemettinen et al. [KMR+94] propose the use of *rule templates*. Rule templates are expressions of the form $A_1 \wedge \ldots \wedge A_k \rightarrow A_{k+1}$ which describe a set of rules (that are of interest to the analyst), by specifying what attributes occur in the antecedent and what attribute is the consequent. The attributes can be classified in a class hierarchy by the user. Then, each A_i in the above template may be instantiated to either an attribute name, a class name C, or an expression $C+$ or $C*$[1]. The user can explicitly specify both what is interesting, *inclusive templates*, and what is not, *exclusive templates*. Then the system reports only the rules that match the inclusive templates after having filtered out the rules that match the exclusive templates.

Padmanabhan and Tuzhilin [PT98] discuss a belief-driven method for discovering unexpected patterns (expressed as rules). The method considers rules and beliefs of the form $X \rightarrow A$, where X is a conjunction of atomic conditions and A is a single atomic condition. The rules are automatically produced by investigating the database, while the beliefs are provided by the analyst. The system associates with each rule measures of support and confidence, in order to identify the strong rules. Among those, only the unexpected ones are reported. Rule $A \rightarrow B$ is unexpected (or interesting) with respect to belief $X \rightarrow Y$ on the dataset $\mathcal{D}$ if three conditions hold: first, the statement $B \wedge Y$ evaluates to *FALSE* (i.e., B and Y logically contradict each other); second, the statement $A \wedge X$ is supported by the facts in the database; and third, the rule $A \wedge X \rightarrow B$ holds. Then, the rule $A \rightarrow B$ is unexpected since the rule $A \wedge X \rightarrow \neg Y$ also holds, which is opposite to the user's belief.

Related to the above work is the framework described by Liu et al. [LHC97]. The proposed system works in the following way. First, a decision tree classifier partitions the tuples of the database into a set of distinct classes. Each of the partitions can be expressed as a rule of the form $P_1 \wedge \ldots \wedge P_r \rightarrow C$, where P_i is a condition on an attribute of the database, and C one of the (known) classes. Second, the user provides a set of rules that expects to hold in the database. Then, the tuples specified by the user rules are classified as *conforming* and *unexpected* according to the initial classification. In the final step, the decision tree classifier produces rules that describe the conforming and the unexpected tuples. At the end of the procedure the system not only informs the user on the validity of the rules, but also provides

[1] The symbol "+" denotes one or more repetitions, and the symbol "*" zero or more.

the user with a characterization of the conforming and unexpected tuples.

The *Data-Monitoring and Discovery-Triggering* (DMDT) paradigm [ST96a] is a framework that supports the user involvement in the discovery process through the use of templates. A similar approach, in that the mining procedure is viewed as an interactive process with the user being an active participant, is proposed in other studies as well [TUA+98] [NLHP98]. However, the main focus of these studies is the efficient integration of association rules mining in database management systems. Such techniques will allow the analyst to pose focused mining queries in the database, receive fast indicative answers, and then refine the queries according to the new results. However, it is the user that has to direct the system in the discovery process, which can be extremely cumbersome and ineffective for large datasets. The model-based rule discovery systems described in the next section try to remedy this problem.

3.3.2 Model-Based Rule Discovery

In model-based rule discovery the user is no longer required to explicitly state which patterns are interesting. Rather, it is the knowledge extraction system that builds a model of the data, and based on that decides what would be interesting enough to report to the analyst. The premise is that the model captures the general trends of the data. Then, the portions of the data that do not follow these trends are deemed interesting. (As we will see later, the system may need some input from the user, but this would be minimal compared to the template-based approach).

Silberschatz and Tuzhilin [ST96b] propose subjective measures of interestingness which, unlike the template-based case, are domain independent. The interestingness of some pattern is defined as its distance from the set of beliefs (in some appropriate space). The more a pattern deviates from the belief system, the more interesting it is. The system may encompass beliefs of two types:

Hard Beliefs. These express the constraints of the domain under consideration. They cannot be changed, and if some data contradicts those beliefs this data is automatically termed interesting (i.e., the data may be wrong).

Soft Beliefs. These are beliefs that the user is willing to change with new evidence. In order to appreciate soft beliefs, we assign a degree to each of them, that specifies how strongly we believe in it.

Note, that hard beliefs are subjective since not everyone would agree as to which are the implicit constraints of each domain.

Formally, the interestingness of a pattern E relative to (soft) belief system B, containing beliefs α_i, and previous evidence ξ is defined as $I(E, B, \xi) = \sum_{\alpha_i \in B} w_i |d(\alpha_i|E, \xi) - d(\alpha_i|\xi)|$, where $d(\cdot)$ is the degree of belief, w_i is the weight of belief α_i and $\sum_{\alpha_i \in B} w_i = 1$. This definition measures by how many degrees the beliefs in B changed as a result of a new pattern E. The above measure can also be used for *belief-driven* discovery systems in the following way. Let D be a database, and δD some new data that is added to D. If there is a belief α in B such that $d(\alpha|\delta D, D) \neq d(\alpha|D)$, then there exists a pattern E in δD such that $I(E, B, D) \neq 0$. Therefore, such an event could trigger the discovery process for the interesting patterns contained in δD. This technique can be incorporated in systems like the DMDT [ST96a] described previously.

Now, the problem is one of defining in mathematical terms the degree of soft beliefs. In the Bayesian approach, the degree of a belief α, $d(\alpha|\xi)$, is defined as the conditional probability $P(\alpha|\xi)$ that α holds given some previous evidence ξ. Given new evidence E we evaluate the new degree of belief in α, $d(\alpha|E, \xi) = P(\alpha|E, \xi)$, using Bayes rule.

In the statistical approach, a belief is expressed as a statistical hypothesis. For example, the belief "women and men receive equal grades in courses" can be formulated as the null hypothesis $H_0 : \mu_w = \mu_m$, where μ_w and μ_m are the average grades of women and men respectively. Then, the degree of belief is defined as the significance level for which the null hypothesis test is accepted. However, the statistical approach can only accommodate some of the beliefs (e.g., the belief "women receive better grades than men" cannot be tested in this way), which makes it impractical for the general case.

Chakrabarti et al. [CSD98] propose a method for mining surprising patterns. The study focuses on the analysis of inter-item correlations along time. The measure of interest is based on the number of bits needed to encode an itemset sequence using some model. The data is an ordered (by time) sequence of baskets, associated with a set of k items. Each basket contains one of the 2^k possible subsets of the k items in it. Thus, a basket can be viewed as the outcome of tossing a 2^k sided coin. The entire sequence of data is encoded according to a model M which is assumed to hold on the data. Then, the code length associated with the encoding of the data $\vec{x}$ is $L(M) + L(\vec{x}|M)$ bits. Note, that since the model M is specifically chosen for the data $\vec{x}$, $L(\vec{x}) > L(\vec{x}|M)$. This way the problem of overfitting is avoided, because the procedure that minimizes the code length takes into account the length of the model as well (which is embedded in the code length expression).

The algorithm proceeds as follows. First, it segments the dataset in such a way that each segment can be associated with a different random process generating the data therein. Then, a different model is built for each segment. The model describes the data by taking into account the counts of each item appearing in the sequence, and computing the probability that it appears in the basket along with the rest of the items. At this point we have a code length corresponding to each itemset (the number of bits required to represent the model and the data). Since we wish to be able to compare itemsets of different lengths, we need to have one more code length to use as the base line. This is achieved by computing another model over the data (using the same segmentation). In the second model we incorporate a parameter θ, that accounts for the difference between the estimated and the actual parameters of the model. (The θ parameter is the same for the entire sequence of data, hence, it cannot be used to derive exact estimates). Then, the most interesting itemsets are those that rank high in the ratio or difference of the two aforementioned models.

Jermaine and Miller [JM99] study the problem of concise representation of large multidimensional datasets. The solution involves focusing on the interesting or unexpected subsets of the database, and allowing more bits for their representation. A part of the dataset is termed unexpected if it deviates from the assumption of attribute independence.

The proposed algorithm incrementally builds a model for the data, which captures the unexpected regions. It begins with the assumption that all the attributes are independent. Then, the data space is partitioned into disjoint hyper-rectangular regions in such a way that within each new region we are as close to the attribute independence assumption as possible. A parameter t measures the deviation from independence. The result is a minimal *t-expected* partitioning of the dataset, which means that there are no partitions with more than t units away from independence. It is minimal in the sense that we cannot find another partition of the dataset with fewer regions. Due to the inherent complexity of the problem, the algorithm actually finds only an approximation of the minimal t-expected partition, using heuristics. However, the authors claim that even a poor approximation is still a useful description of the irregularities of the dataset.

3.4 Deviation Detection

An interesting category of unexpected patterns form the *deviants* or *outliers*. An outlier is "an observation that appears to deviate markedly from other members of the sample in which it occurs" [BL94]. This fact may raise suspicions that the specific observation was generated by a different mechanism than the rest of the data. This mechanism may be an erroneous procedure of data measurement and collection, or an inherent variability in the domain of the data under inspection. Nevertheless, in both cases such observations are interesting, and the analyst would like to know about them.

There is extensive literature in the statistics community regarding outlier detection [BL94]. Yet, this work is not directly applicable to databases since it pre-supposes that the user knows the distribution which generated the data. Based on this knowledge, statistical tests can examine the probability that the given distribution produced the data point under consideration, and accordingly accept it or reject it. The problem is that in the vast majority of cases the distributions that produced the data are unknown, and extremely hard to estimate. In some cases we may inherently not be able to model the data with some distribution (e.g., in a database with mixed numeric and text values). Moreover, when dealing with large collections of data we need efficient algorithms that are fast and scale linearly or near-linearly with the dataset size. Thus, other methods for outlier detection are necessary. In the following paragraphs we will discuss some of the techniques used in the database community.

3.4.1 Detecting Deviants In Databases

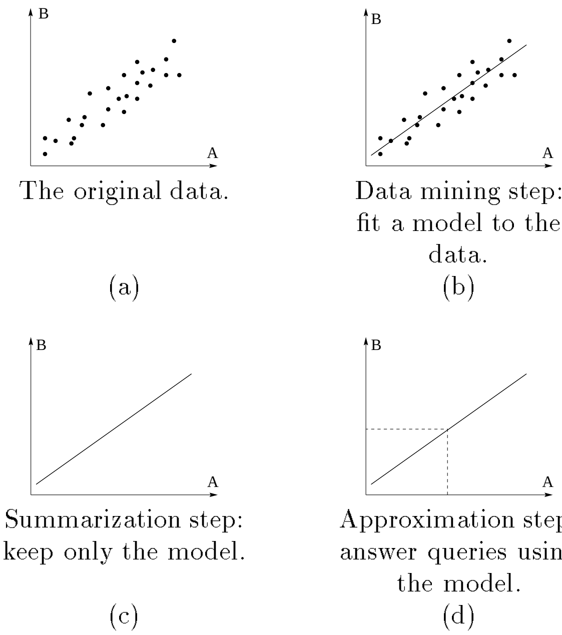
The problem of finding outliers in large, multidimensional datasets is studied by Knorr and Ng [KN98]. They introduce the notion of distance-based outliers $DB(p, D)$ which are defined as follows. An object O in a dataset T is a $DB(p, D)$-outlier if at least a fraction p of the objects in T lies further than distance D from O. The above definition is a generalization of the notion of outliers supported by statistical outlier tests for standard distributions. The advantage is that this approach does not require any prior knowledge of the underlying data distribution, but rather uses the intuitive explanation that an outlier is an observation that is sufficiently far from most other observations in the database. However, it does not provide a ranking for the outliers, which may be useful in some cases.

The study describes three algorithms, which require the user to specify the parameters p and D. Note that a suitable value for D can only be determined through experimentation. The first algorithm is a simple nested loop algorithm, whose complexity is linear on the number of dimensions k, but quadratic on the size of the dataset N. The second algorithm operates on in-memory datasets, providing complexity linear on N, but exponential on k. The third algorithm works with disk-resident datasets, and guarantees no more than three passes over the entire database. Nevertheless, the exponential growth of complexity on the dimensionality renders this approach viable only on low dimensional datasets.

In subsequent work [KN99] the algorithms are extended to discover outliers in lower dimensionality subspaces of the data as well. The definition of outliers is augmented by the characterizations *strong* and *weak*. An outlier O in the attribute space $\mathcal{A}_{\mathcal{P}}$ is strong if there are no outliers in any subspace $B \subset \mathcal{A}_{\mathcal{P}}$. Otherwise O is termed a weak outlier.

The aforementioned technique is close to the template-based methods discussed in a previous section. An example of the model-based approach is the technique described by Arning et al. [AAR96], which identifies *exceptions* (set of tuples) among the tuples of a large database. In a database with a set of items I, an exception set I_j is the subset of items that contributes the most to the dissimilarity of I. The goal is to find the exception set I_j with the smallest cardinality.

The proposed algorithm uses a dissimilarity function $\mathcal{D}(I_x)$, that measures the degree of relevance of the items in I_x. It may be any function that returns a low value if the elements in I_x are similar to each other, and a high value otherwise. An example of such a function for numerical data is the variance. Obviously, the challenge is to devise a function that will effectively capture the notion of similarity, and at the same time be computationally efficient. Nevertheless, it is not obvious what the choice of function $\mathcal{D}(\cdot)$ should be in various domains, and which class of functions can capture local phenomena as opposed to global. This point is important, since in the latter case we will end up identifying merely the extreme points of the dataset.



The original data.

(a)

Data mining step: fit a model to the data.

(b)

Summarization step: keep only the model.

(c)

Approximation step: answer queries using the model.

(d)

Figure 4: From data mining to approximate query answering.

4 Approximate Query Answering

4.1 Overview

The rapid growth in size of databases outpaces the technological advances which allow fast disk access and data processing. This results in degradation of performance. One of the most efficient ways to remedy this situation is *data reduction*. Data reduction techniques are analogous to lossy compression schemes. They try to summarize huge collections of data using only small amounts of storage space. The summaries are supposed to capture the intricacies of the dataset, and be able to provide estimations about the actual data. In approximate answering we are willing to sacrifice some of the accuracy for the sake of faster response times and reduced storage requirements.

In this section we examine the relationship between summarization and data mining. Then, we review some of the work relevant to approximate query answering. The focus will be on histogram techniques, which have attracted a lot of attention and are widely used. A more elaborate discussion of the summarization and approximation methods available can be found elsewhere [BDF+97].

4.2 The Relationship to Data Mining

Before attempting to establish the connection between data mining and data reduction, it would be useful to answer another question first: is data summarization related to approximate query answering? The answer to this question is affirmative. Indeed, there are a number of summarization techniques that can also be used to produce estimates of the actual data [BDF+97]. Though, we do not argue that every data reduction method can be used for approximate query answering. For the rest of this discussion we will consider the methods that belong to the intersection.

The role of data mining is to discover general patterns that describe the data. (We will ignore the fact that sometimes we are only interested in a small subset of those patterns). These patterns may have the form of rules, or some model. Each of the generated patterns represents a subset of the raw data. Actually, it is exactly because there are many instances in the data supporting a pattern, that it gets generated. Therefore, the above procedure may be viewed as a means to data summarization [FPSM91] [Man99]. An even stronger statement is that data mining, or data reduction, may be regarded as learning [JM99].

Apparently, it is not clear as to what falls in each category, and under which conditions some method may be classified to one or more of the functionality classes described above. Nevertheless, we argue that in some circumstances we can use a technique to achieve both goals of knowledge extraction and data reduction (or data estimation). An example that illustrates the above point is depicted in Figure 4. In this example, a two-dimensional dataset exhibits a correlation between the two attributes. We may use a data mining technique to identify this pattern and produce a corresponding model (Figure 4.b). Then, the derived model can be regarded as a concise summary of all the data points in the dataset (Figure 4.c), allowing us to save space by storing only the model of the data. Finally, the same model can be used for approximate query answering (Figure 4.d). That is, to estimate the value of one attribute given the value of the other in the absence of the real data.

4.3 Approximation in Databases

Histograms approximate the data in one or more attributes of a relation by grouping attribute values into buckets, and approximating the real attribute values (or frequencies) in the data based on summary statistics maintained in each bucket. The mathematic formulation of the problem is as follows. Let $\mathcal{D}$ be the (finite) domain of an attribute X in relation R, and the value set $\mathcal{V} \subseteq \mathcal{D}$ be the set of values of X actually present in R. We write $\mathcal{V}$ as $\mathcal{V} = \{v_i : 1 \leq i \leq M\}$, where $v_i < v_j$ for $i < j$. The set $\mathcal{V}$ can be represented as an M-long vector V. We will refer to any sequence of contiguous elements of V as a segment. We want to construct a summary vector H by selecting $B < M$ non-overlapping segments (buckets) that together include every element in V. We also provide a reconstruction function $R(\cdot)$ that approximates vector V (of length M) from vector H (of length $B < M$), and an error function $E(\cdot)$ that measures the distance between V and $R(H)$. The challenge is to find the vector H of length B which minimizes $E(H)$, given the initial vector V.

Several types of histograms have been suggested in the literature. Among those are the *equi-width* and *equi-depth* histograms. Equi-width histograms choose bucket boundaries in such a way that all bucket ranges are equal, and equi-depth ones guarantee that the total number of tuples mapped to each bucket is the same for all buckets. More recently, new types of histograms proved to perform better [PIHS96]. These are: *maxdiff*, which places the $B-1$ bucket boundaries between the v_i values with the $B-1$ maximum differences; *v-optimal*, that minimizes the variance of the values in each bucket; and *compressed*, which dedicates an entire bucket to each one of the K highest values in V, and organizes the rest $B-K$ buckets so that all of them have the same total sum of values.

Note that all of the above approaches do not directly address the problem of minimizing $E(H)$, but rather provide approximations. Jagadish et al. [JKM$^+$98] describe an algorithm that finds the optimal solution in time quadratic in M and linear in B. This is achieved using a dynamic programming algorithm that recursively finds for each segment the optimal split point (i.e., the point that minimizes the overall $E(H)$). The same algorithm can be augmented to provide quality guarantees on the maximum error of the estimations.

A subsequent study [JKM99] uses the above algorithm for deviation detection in time series. The main idea comes from the information theoretic principle of representation length. Deviants are termed the points whose removal of the sequence causes the largest reduction in the representation cost of the series.

The algorithm employs histograms for the representation of the time series V. Then, the problem is to find the points that once removed will decrease $E(H)$ the most. Note that when we remove a data point from V we decrease the number of buckets by one to account for the separate storage required for it. It turns out that the aforementioned dynamic programming algorithm can be adapted to solve this problem, albeit, in time quadratic to the size of the dataset. The advantage of this approach is that it can effectively capture deviant points in local contexts, that is in relation to their neighbours, as well as in the global aspect. It is also interesting to note that the above technique (storing the deviant points in separate buckets) leads to more accurate histograms for the same storage space.

Matias et al. [MVW98] propose the use of a multiresolution wavelet decomposition technique to approximate a data distribution. The (Haar) wavelet transform takes as input a signal $X = \{x_1, \ldots, x_n\}$, and computes the averages and pairwise differences of all the pairs (x_{2i}, x_{2i+1}). Then, it recursively replaces each pair of points with their average value, and keeps the differences (along with the index of the point position) which are the *coefficients* of the transformed signal. In order to get the signal back, we only need the very last point value (where the recursion stopped), and the resulting coefficients. It turns out that we can disregard any coefficients with sufficiently small magnitude at the cost of incurring a small error.

For a given storage space, the algorithm has to decide which m coefficients to keep for the representation of the original data. The study describes a set of greedy heuristics for this task. The complexity of the algorithm is linear in $|\mathcal{D}|$ (since the transformation requires padding all points in $\mathcal{D}-\mathcal{V}$ with zeroes). This is approximately the cost of answering queries as well (assuming $m << |\mathcal{D}|$). The experimental evaluation shows that this method performs better than *maxdiff* histograms in the general case. However, a more elaborate comparison among the different approaches is necessary.

Faloutsos et al. [FJS97] investigate a problem relevant to the histogram techniques. Histograms can be viewed as a summarized representation of the data. This study describes a method for estimating the original detailed data from the stored summary. Obviously, the problem is under-specified. One way of making up for the missing equations is to assume uniformity of values. However, the study shows that this assumption leads to poor estimates of the real values. Instead, the authors propose the technique of *linear regularization* which essentially tends to smoothen the distributions it approximates. In other words, adjacent values in the solution will only have a small difference. Experiments show that this is a valid assumption for many datasets and results in better estimates. The algorithm has complexity linear in the size of the dataset. However, it does not scale up to many dimensions.

4.4 Approximation in Data Cubes

The problem of providing approximate answers for data cube structures has not been addressed sufficiently yet. The two methods outlined in the following paragraphs are merely extensions of previous approaches to many dimensions.

Poosala and Ganti [PG99] propose the use of the *MHIST* multidimensional histograms [PI97], which are the multidimensional version of *maxdiff*. The *MHIST* histogram starts by considering the whole data space as a single bucket. Then, it splits along the dimension whose 1-dimensional aggregate distribution contains the two adjacent points with maximum difference. The algorithm exploits the lattice structure of a data cube, and constructs histograms for only a subset of the cuboids (the rest are approximated using the computed histograms). The solution is based on a greedy heuristic, and runs in time exponential to the number of attributes in the data cube.

The wavelet transform extends naturally in many dimensions, and is the approach explored by Vitter et al. [VWI99]. When considering multiple dimensions, the wavelet transform is applied on all dimensions in a sequential manner (i.e., one dimension at a time). The complexity of the algorithm is quasi-linear in the total number of cells of the data cube, and there is an additional cost for answering queries, which depends on the cardinality of the dimensions involved in the query. Recently, this approach was extended to accommodate sparse data cubes [VW99].

The experimental results of the above two methods are not comparable, because different error measures are used in the studies. Once again, extensive studies should be performed in order to evaluate the different approaches.

5 Data Mining in Data Cubes

5.1 Overview

The OLAP paradigm was introduced as an efficient and effective means to data exploration [CCS93]. The premise was that the multidimensional view of the data along with the flexibility of the data cube operator would constitute a viable solution for the decision support problem. However, the sheer size of data cubes renders the task of intelligent analysis formidable. This fact merely transposed the problem of having to study a wealth of information from the bottom layer databases to the decision support OLAP cubes. Hence, the need is now becoming apparent for intelligent analysis of data cubes as well.

Data mining is the research area that has the potential to address the above need. Numerous techniques have already been proposed for knowledge extraction, and many lessons have been learnt during this process. Nevertheless, there is still very little work done in the specific area of data mining in data cubes. This is due partly to the fact that OLAP technology is a new area which is not well defined yet, and also to the fact that most of the attention is naturally directed towards raw databases where the need for analysis is more pressing.

5.2 The General Case

The intersection of data mining and OLAP is extensively discussed by Han [Han97]. The author describes the *DBMiner* application which integrates OLAP technology with data mining techniques. Algorithms for performing characterization, classification, clustering, and association rule mining are embedded in the OLAP sub-system of *DBMiner*.

Such a system is useful [ZXH98], and a wide range of applications can benefit from it. Nevertheless, one would expect that all the preprocessing steps and the specialized structure of data cubes could be employed in a more involved way. That is, we should be able to identify patterns in the data that could not possibly be discovered from the same data in raw format (i.e., not organized in a data cube). In the next section we discuss an approach that follows the aforementioned path.

5.3 Detecting Deviants In Data Cubes

Sarawagi et al. [SAM98] investigate the problem of identifying outliers in data cubes. The current practice in OLAP systems is to facilitate hypothesis-driven exploration of data cubes, where the analyst tries to find interesting information by simply using the data cube operators. This study proposes instead the *discovery-driven* exploration paradigm. In this environment it is the system that recommends to the user potentially interesting paths of exploration in the data cube.

The algorithm mines the data for exceptions, and summarizes the results at different levels of the hierarchy. Exceptions are termed the cell values (of any aggregation level) that differ significantly from the anticipated value calculated using a model that takes into account all the aggregates in which the value participates. Clearly, this is a model-based approach as presented in Section 3.3. The model used in this study is similar to table analysis methods used in the statistics community, and has the advantage of taking into account all the dimensions in which a value belongs in order to characterize it.

Experiments with real datasets exhibit the flexibility of the algorithm in identifying outliers. Furthermore, the algorithm is transformed so that it can be incorporated in the computation of the datacube. This is crucial, since the complexity of the algorithm is prohibitive otherwise. Even then, the overhead introduced by the computation of the model is significant (up to more than 100%). Note that this approach requires that the whole datacube be computed, which is not a common practice, especially for large datasets. Moreover, the algorithm cannot update the model when new data arrive.

A subsequent study [Sar99] describes a method that produces an informed explanation for drops or increases that are observed in the data. More specifically, the *DIFF* operator is introduced, which explores the reasons for which a certain aggregated quantity is lower or higher in one cell compared to another. The reasons are expressed in the form of other cell values, belonging to aggregation levels of finer detail, that are most responsible for the difference under investigation. The challenge here is to not simply report all (or even the largest) changes in the detailed data, but rather provide a concise explanation. The former approach would either produce too many results or account for a small portion of the difference, while the latter has a high information content because it summarizes rows describing similar changes.

The problem is formulated in information theoretic terms as follows. A sender has a cube C_b and wants to transmit it to a receiver who already knows about cube C_a. In addition, the receiver has a summary $\mathcal{A}$ of N values, that describes the differences of the cubes. Then, the objective is to find the $\mathcal{A}$ (i.e., determine which N tuples to choose) which minimizes the amount of additional information that the receiver has to get in order to reconstruct C_b without errors. The solution is based on a dynamic programming algorithm which scales linearly with the parameters of the problem.

6 Challenging Directions

As we discussed earlier, the problem of knowledge extraction in data cubes becomes more and more important as the amount of information that is stored therein increases. The OLAP system should have the ability to automate or guide the decision support process, by providing insight into the data. The aforementioned goal can be achieved in various ways, some of which we described in the last section.

Evidently, the data mining techniques have a substantial role to play in the advantage of decision support systems. They can operate on large collections of data, and they have the ability to identify patterns and build succinct models to describe the data. The patterns reveal key characteristics of the processes that produce the data, and the models capture the major trends and correlations. These models, apart from giving precious information on the way the data is organized, can also be used to achieve summarization and approximation, delivering multiple benefits to the user. As we presented in this study, data mining and data summarization techniques can interact in order to determine the portion of the dataset that is worth storing. In some cases the above procedure maintains (or can be instrumented to maintain) enough information to enable approximate query answering. This can be a valuable by-product of data mining, especially when considering the need for storing large datasets and for providing fast answers.

It must be noted though, that the mining techniques have to evolve in order to be maximally beneficial in the new environment. They need to adapt, and accommodate the specific characteristics of OLAP systems, such as the multidimensional and hierarchical structure of data cubes. That means they should take into account and efficiently process numerous different subsets of dimensions, so as to uncover knowledge hidden in various subcubes of the original data cube. Furthermore, the algorithms should be able to produce answers at different granularities, following the hierarchies imposed on the dataset. Then, the user might choose the level of detail of the answer, paying the corresponding cost in computation time and effort. It is an interesting research direction to investigate how the traditional mining techniques fit in this new environment, and what are the new opportunities for data mining.

Last but not least, we believe that measures of interestingness are an essential ingredient for successful mining. Either explicitly determined by the user, or automatically recognized by the system, they help to focus the discovery process on new, unexpected knowledge. They can also be combined with data reduction techniques in order to evaluate the information content of each piece of the data. The low-value data can then be discarded, leaving only the relevant and interesting subsets for further analysis.

Acknowledgements

We would like to thank Alberto Mendelzon and José Blakeley for their comments that helped improve the content of this paper.

All the images presented herein, except for Figure 4, come from the corresponding papers.

References

[AAD+96] Sameet Agarwal, Rakesh Agrawal, Prasad Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, and Sunita Sarawagi. On the Computation of Multidimensional Aggregates. In *VLDB International Conference*, pages 506–521, Mumbai (Bombay), India, September 1996.

[AAR96] Andreas Arning, Rakesh Agrawal, and Prabhakar

Raghavan. A Linear Method for Deviation Detection in Large Databases. In *International Conference on Knowledge Discovery and Data Mining*, pages 164–169, Portland, OR, USA, August 1996.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference*, pages 207–216, Washington, DC, USA, May 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *VLDB International Conference*, pages 487–499, Santiago de Chile, Chile, September 1994.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, March 1995.

[BDF+97] Daniel Barbará, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph M. Hellerstein, Yannis Ioannidis, H. V. Jagadish, Theodore Johnson, Raymond Ng, Viswanath Poosala, Kenneth A. Ross, and Kenneth C. Sevcik. The New Jersey Data Reduction Report. *Bulletin of the Technical Committee on Data Engineering*, 20(4):3–45, 1997.

[BL94] V. Barnet and T. Lewis. *Outliers in Statistical Data*. John Wiley and Sons, Inc., 1994.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *ACM SIGMOD International Conference*, pages 255–264, Tuscon, AZ, USA, June 1997.

[CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP to User-Analysts: An IT Mandate. http://www.hyperion.com, 1993.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *Sigmod Record*, 26(1):65–74, 1997.

[CHY96] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: An overview from database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[CSD98] Soumen Chakrabarti, Sunita Sarawagi, and Byron Dom. Mining Surprising Patterns Using Temporal Description Length. In *VLDB International Conference*, pages 606–617, New York, NY, USA, August 1998.

[Fay98] Usama Fayyad. Mining Databases: Towards Algorithms for Knowledge Discovery. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1), 1998.

[FJS97] Christos Faloutsos, H. V. Jagadish, and N. D. Sidiropoulos. Recovering Information from Summary Data. In *VLDB International Conference*, Athens, Greece, August 1997.

[FPSM91] William J. Frawley, Gregory Piatetsky-Shapiro, and Chistopher J. Matheus. Knowledge Discovery in Databases: An Overview. *Gregory Piatetsky-Shapiro and William J. Frawley editors, Knowledge Discovery in Databases*, pages 1–27, 1991.

[GBLP96] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In *International Conference on Data Engineering*, pages 152–159, New Orleans, LO, USA, March 1996.

[Han97] Jiawei Han. OLAP Mining: An Integration of OLAP with Data Mining. In *IFIP Conference on Data Semantics*, pages 1–11, Leysin, Switzerland, October 1997.

[Han98] Jiawei Han. Towards On-Line Analytical Mining in Large Databases. *ACM Sigmod Record*, 27(1):97–107, 1998.

[HRU96] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing Data Cubes Efficiently. In *ACM SIGMOD International Conference*, pages 205–216, Montreal, Canada, June 1996.

[JKM+98] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal Histograms with Quality Guarantees. In *VLDB International Conference*, pages 275–286, New York, NY, USA, August 1998.

[JKM99] H. V. Jagadish, Nick Koudas, and S. Muthukrishnan. Mining Deviants in a Time Series Database. In *VLDB International Conference*, pages 102–113, Edinburgh, Scotland, September 1999.

[JM99] Christopher Jermaine and Renée J. Miller. Lossy data reduction without loss of information. *Submitted for publication*, 1999.

[KMR+94] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *ACM International Conference on Information and Knowledge Management*, pages 401–407, Gaithersburg, MD, USA, November 1994.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB International Conference*, pages 392–403, New York, NY, USA, August 1998.

[KN99] Edwin M. Knorr and Raymond T. Ng. Finding Intensional Knowledge of Distance-Based Outliers. In *VLDB International Conference*, pages 211–222, Edinburgh, Scotland, September 1999.

[KR99] Yannis Kotidis and Nick Roussopoulos. DynaMat: A Dynamic View Maintenance System for Data Warehouses. In *ACM SIGMOD International Conference*, Philadelphia, PA, USA, June 1999.

[LHC97] Bing Liu, Wynne Hsu, and Shu Chen. Discovering Conforming and Unexpected Classification Rules. In *Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, Nagoya, Japan, August 1997.

[Man99] Heikki Mannila. Theoretical Frameworks for Data Mining. Invited talk in ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, May 1999.

[MQM97] Inderpal Singh Mumick, Dallan Quass, and Barinderpal Singh Mumick. Maintenance of Data Cubes and Summary Tables in a Warehouse. In *ACM SIGMOD International Conference*, pages 100–111, Tuscon, AZ, USA, June 1997.

[MVW98] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-Based Histograms for Selectivity Estimation. In *ACM SIGMOD International Conference*, pages 448–459, Seattle, WA, USA, June 1998.

[NLHP98] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory Mining and Pruning Optimizations of Constrained Associations Rules. In *ACM SIGMOD International Conference*, pages 13–24, Seattle, WA, USA, June 1998.

[PCY95] Jong-Soo Park, Ming-Syan Chen, and Philip S. Yu. An Effective Hash Based Algorithm for Mining Association Rules. In *ACM SIGMOD International Conference*, pages 175–186, San Jose, CA, USA, May 1995.

[PG99] Viswanath Poosala and Venkatesh Ganti. Fast Approximate Answers to Aggregate Queries on a Data Cube. In *International Conference on Scientific and Statistical Database Management*, Cleveland, OH, USA, 1999.

[PI97] Viswanath Poosala and Yannis E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *VLDB International Conference*, pages 486–495, Athens, Greece, August 1997.

[PIHS96] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *ACM SIGMOD International Conference*, pages 294–305, Montreal, Canada, June 1996.

[PT98] Balaji Padmanabhan and Alexander Tuzhilin. A Belief-Driven Method for Discovering Unexpected Patterns. In *International Conference on Knowledge Discovery and Data Mining*, pages 94–100, New York, NY, USA, August 1998.

[RKR97] Nick Roussopoulos, Yannis Kotidis, and Mema Roussopoulos. Cubetree: Organization of and Bulk Incremental Updates on the Data Cube. In *ACM SIGMOD International Conference*, pages 89–99, Tuscon, AZ, USA, June 1997.

[RS97] Kenneth A. Ross and Divesh Srivastava. Fast Computation of Sparse Datacubes. In *VLDB International Conference*, pages 116–125, Athens, Greece, September 1997.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *VLDB International Conference*, pages 407–419, Zurich, Switzerland, September 1995.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 3–17, Avignon, France, March 1996.

[SAM98] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven Exploration of OLAP Data Cubes. In *International Conference on Extending Database Technology*, pages 168–182, Valencia, Spain, March 1998.

[Sar97] Sunita Sarawagi. Indexing OLAP Data. *IEEE Data Engineering Bulletin*, 20(1):36–43, 1997.

[Sar99] Sunita Sarawagi. Explaining Differences in Multidimensional Aggregates. In *VLDB International Conference*, pages 42–53, Edinburgh, Scotland, September 1999.

[SON95] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *VLDB International Conference*, pages 432–444, Zurich, Switzerland, September 1995.

[SS94] Sunita Sarawagi and Michael Stonebraker. Efficient Organization of Large Multidimensional Arrays. In *International Conference on Data Engineering*, pages 328–336, Houston, TX, USA, February 1994.

[ST96a] Avi Silberschatz and Alexander Tuzhilin. User-Assisted Knowledge Discovery: How Much Should the User Be Involved. In *ACM-SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, ON, Canada, June 1996.

[ST96b] Avi Silberschatz and Alexander Tuzhilin. What Makes Patterns Interesting in Knowledge Discovery Systems. In *IEEE Transactions on Knowledge and Data Engineering*, volume 8, pages 970–974, May 1996.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In *VLDB International Conference*, pages 134–145, Mumbai (Bombay), India, September 1996.

[TUA+98] Dick Tsur, Jeffrey D. Ullman, Serge Abiteboul, Chris Clifton, Rajeev Motwani, Svetlozar Nestorov, and Arnon Rosenthal. Query Flocks: A Generalization of Association-Rule Mining. In *ACM SIGMOD International Conference*, Seattle, WA, USA, June 1998.

[VW99] Jeffrey Scott Vitter and Min Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *ACM SIGMOD International Conference*, pages 193–204, Philadelphia, PA, USA, June 1999.

[VWI99] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. Fast Approximate Answers to Aggregate Queries on a Data Cube. In *International Conference on Scientific and Statistical Database Management*, Cleveland, OH, USA, 1999.

[ZDN97] Yihong Zhao, Prasad Deshpande, and Jeffrey F. Naughton. An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. In *ACM SIGMOD International Conference*, pages 159–170, Tuscon, AZ, USA, June 1997.

[ZXH98] Osmar R. Zaïane, Man Xin, and Jiawei Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Advances in Digital Libraries*, pages 19–29, Santa Barbara, CA, USA, April 1998.