ISSN 0976 – 1411

Available online at www.internationaleJournals.com

International eJournals

INTERNATIONAL eJOURNAL OF MATHEMATICS AND ENGINEERING

www.InternationaleJournals.com

International eJournal of Mathematics and Engineering 229 (2013) 2233 – 2241

INTRUSION DETECTION THROUGH DATA MINING APPROACH

Jaswanth Bhargav[1], Syed Umar[2] and Paladugu Hiranmai[3]
Department of ECM, KL University, A.P. INDIA

Abstract: The goal of an *intrusion detection system* (IDS) is to identify authorized and unauthorized intruders by differentiating anomalous network activity from normal network traffic. Data mining methods have been used to build automatic intrusion detection systems. The central idea is to utilize auditing programs to extract a set of features that describe each network connection or host session, and apply data mining programs to learn rules that capture intrusive and non-intrusive behavior. The goal of this paper is to provide a survey of some works that employ data mining techniques for intrusion detection and to address some technical issues. A new idea is proposed in the paper that will view intrusion detection from a data warehouse perspective and integrate data mining and on-line analytical processing (OLAP) for intrusion detection purposes.

Keywords: Intrusion detection, data mining, data warehousing

Introduction

With the ever-increasing growth of computer networks and emergence of electronic commerce in recent years, computer security has become a priority. Since intrusions take advantage of vulnerabilities in computer systems and

Socially engineered penetration techniques, in addition to intrusion prevention techniques (such as user authentication), intrusion detection is often used as another wall of protection. This is not an easy task due to the vastness of the network activity data and the need to regularly update the IDS to cope with new, unknown attack methods or upgraded computing environments. Mukherjee, Heberlein, and Levitt [1] defined intrusion detection as identifying unauthorized use, misuse, and abuse of computer systems by both inside and outside intruders. There are many categories of network intrusions [2]. Examples include SMTP (Send Mail) attacks, password guessing, IP spoofing, buffer overflow attacks, multi-scan attacks, denial of service (DoS) such as ping-of-death, SYN flood, etc. Intrusion detection can broadly be divided into two categories: *misuse detection* and *anomaly detection* [3]. Misuse detection is based on the knowledge of system vulnerabilities and known attack patterns, while anomaly detection assumes that an intrusion will always reflect some deviation from normal patterns. Many AI techniques have been applied to both misuse detection and anomaly detection. Pattern matching systems like rule-based expert systems, state transition analysis, and genetic algorithms are direct and efficient

ways to implement misuse detection. On the other hand, inductive sequential patterns, artificial neural networks, statistical analysis and data mining methods have been used in anomaly detection. Data mining can be defined as the process of discovering implicit, unknown and useful information from databases [4]. Data mining methods can be applied to extensively gathered audit data to compute models that can capture the intrusive and non-intrusive behavior. Audit data consists of pre-processed time-stamped audit records, each with a number of features. They contain useful data from which a well-designed data mining system can discover beneficial information. For example, a typical audit log file can contain source IP address, destination IP address, type of service, status flag etc. for a connection. Data mining approaches provide automatic models that eliminate the need to manually analyze and encode intrusion patterns. The goal of this paper is to provide a review of previous work that has applied data mining in intrusion detection. Some performance issues will also be addressed. Since data mining techniques have to deal with large amount of audit data, the intrusion detection task can be viewed from a data warehouse perspective. A new idea is proposed in this paper that integrates data mining and on-line analytical processing (OLAP) for intrusion detection purposes. The remainder of the paper is organized as follows. Section 2 provides a review of related work. Section 3 discusses some technical issues that need to be addressed. Section 4 proposes an idea for intrusion detection that is based on data warehousing and data mining. Finally, the paper ends with concluding remarks in Section 5.

Previous Works

Association Rules

The goal of mining association rules is to derive multi-feature (attribute) correlations from a database table. An association rule is an implication of the form $X \rightarrow Y$ [c,s], where X and Y are disjoint item sets, s is the support of $X \grave{E} Y$ (indicating the percentage of total records that contain both X and Y), c is the confidence of the rule and is defined as $sX\grave{E}Y/sX$ [5]. It has been observed that program executions and user activities exhibit frequent correlations among system features. A typical example of an association rule obtained from audit data can be *ftp* ® *get*[.4,.1], which implies 40% of the time when the user uses the ftp command, get command is also invoked and doing so constitutes 10% of the commands issued by the user. Audit data can be formatted into a database table where each row is an audit record and each column is a field (system feature) of the audit records. Lee and Stolfo [6] extended the basic association rules algorithms to capture the consistent behaviors in program execution and user activities. The rules mined from audit data are merged and added into an aggregate rule set to form the user's normal profile. Two rules are merged if their right and left hand sides are exactly the same or their RHSs can be combined and LHSs can also be combined, and the support and confidence values are close. For example, *service=http* ® *src_bytes*=20 and *service=http* ® *src_bytes*=30 can be combined into *service=http* ® 20£*src_bytes*£30. To analyze a user login session, frequent patterns are mined from the sequence of commands during the session and this new pattern set is compared with the normal profile. Similarity functions are used to evaluate deviations involving missing or new rules, violation of the rules (same antecedent but different consequent), and significant changes in support of the rules. For example, if the new set has n patterns and m patterns among them can be merged with patterns in the profile set, then the similarity score can be m/n.

Frequent Episodes

It is often required to study the frequent sequential patterns of network events in order to understand the nature of many attacks. Lee and Stolfo [6] used frequent episodes [7] to represent the sequential audit record patterns. Given a set of time-stamped event records, an interval [*t*1..*t*2] is the sequence of event records starting at timestamp *t*1 and ending at *t*2. A frequent episode rule is an implication of the form *X*, *Y*® *Z* [*c*,*s*,*w*], where *X*, *Y*, and *Z* are item sets that together form an episode, *s* is the support of *X*È*Y*È*Z* (indicating the percentage of total records that contain *X*, *Y*, and *Z*), *c* is the confidence of the rule (defined as *sX*È*Y*È*Z*/*sX*È*Y*), and *w* is the maximum width of an occurrence. A typical example of a frequent episode rule can be (*service=http*, *flag=S*0, *dest_host=louie*), (*service=http*, *flag=S*0, *dest_host=louie*) ® (*service=http*, *flag=S*0, *dest_host=louie*) [.9,.05,.3]. This implies 90% of the time after two *http* connections with *S*0 flag are made to host *louie*, within 3 seconds from the first of these two, the third similar connection is made, and this pattern occurs in 5% of the data. Lee and Stolfo [6] extended the original frequent episodes algorithm to compute frequent sequential patterns in two phases. First, it finds the frequent associations using the set of *axis features* (described next), and then it generates the frequent sequential patterns from these associations. The selection of the time window size *w* is critical here. A good window size should allow the capturing of sufficient patterns.

Feature Selection/Reduction

One of the problems in mining rules from audit data is that certain features are essential in describing the data, while others provide only auxiliary information. The rules should describe patterns only related to the essential features. Rules containing irrelevant features are not useful and they may become misleading to some extent. For example, an association *src_bytes*=50 ® *flag=SF* seems irrelevant since the relation between the number of bytes from the source (*src_bytes*) and the normal status (*flag*=SF) is very unlikely to carry any significance. Hence, it is important to consider only the relevant set of features that can best describe the system behavior. This set of features is referred to as *axis* features. For a particular intrusion type, some of the features might have a direct or indirect relationship with the signatures of the intrusion type. However, this set of features may not relate to other intrusion types. A minimized set of features is also required to reduce the computational requirements and achieve real time response to intrusion detection. Hence, selection of appropriate feature set is critical for the performance of the IDS. Lee and Stolfo [6] selected the axis features by empirical knowledge. Shi [8] used genetic algorithms to automatically select the appropriate set of features for a specific type of intrusion. His method showed improvements over approaches where the features are selected from empirical evidence. Mukkamala, Gagnon, and Jajodia [9] discussed three feature reduction methods: *significance test*, *mutual significance test*, and *rule-based methods*. Significance test determines whether or not a feature contributes correctly in classifying an observed user activity as intrusive or non-intrusive. If the significance of a feature is lower than a critical threshold value at a given confidence level, that feature can be eliminated. The mutual significance test determines the inter-feature dependencies. When a feature is present and significant, the other one may be insignificant and thus can be eliminated. Rule-based methods are used to derive a set of IF-THEN decision rules that identify user behavior. The features not appearing in the rules may be eliminated.

Fuzzy Data Mining

Audit data contains quantitative features. During mining, the quantitative data are partitioned into intervals. But a sharp boundary problem results from this partition that may create problems in intrusion detection. For example, let us assume [*a*1..*ap*] and [*ap*+1..*an*] are two intervals for a quantitative attribute *A*, *ap* has a support of 15%, *ap*+1 has a support of 5%, and the support threshold is 10%. Even if *ap*+1 lies near a high support value, it may not gain enough support. Now, if the interval [*a*1..*ap*] is mined as normal pattern, the interval [*ap*+1..*an*] will be considered as abnormal. Similarly, an intrusive pattern with a small variance may fall inside [*ap*+1..*an*] and remain undetected. To overcome this boundary problem, Luo and Bridges [10] developed an intrusion detection system by integrating fuzzy logic with data mining algorithms (association rules and frequent episodes). They categorize quantitative features into categories having fuzzy membership values. For example the feature *datasize* can be divided into three categories *low*, *med*, and *high*. If it is a non-fuzzy feature, then a particular value of *datasize* would fall into exactly one category and would have a membership of 1 for that category and 0 for all other categories. But if it is a fuzzy feature, then a particular value of *datasize* can fall into more than one category with some fuzzy membership values. The authors used a normalized measure to compute the fuzzy membership values. For example, a particular value of *datasize* can be "*low*" with 0.9 and "*med*" with 0.1. Though the system works well generally, the selection of fuzzy membership function parameters is done by experience that may lead to some false alarms. Shi [8] used genetic algorithms to automatically optimize the fuzzy-membership function parameters. In his approach, he defined a chromosome to consist of a sequence of the fuzzy function parameters. The process starts with a random initial population of chromosomes where each chromosome is a possible set of parameters. A fitness function is used that gives preferences to high similarity between rules mined from reference and non-intrusive data and to low similarity between rules mined from reference and intrusive data. The process evolves a population of chromosomes to come up with an optimized set of parameters. As a continuation of this work, Bridges and Vaughn [11] proposed a prototype intelligent intrusion detection system (IIDS) utilizing fuzzy mining and genetic algorithms.

Incremental Mining

To make the mining process efficient, the audit data needs to be detailed enough and needs to contain data at the lowest possible level. Practically, it becomes very difficult to tackle such large data since the whole data set needs to be scanned to compute the support of attributes. Barbara, Jajodia, and Wu [12] described an incremental technique that uses association rules and classification techniques to detect attacks. It does not use the entire data set to mine rules. The rules are categorized according to the time of the day and day of the week. An incremental on-line algorithm is used to detect rules that receive strong support during a sliding window of pre-determined size. It compares these rules with a previously generated profile and selects rules not in that profile. A new rule is reported as suspicious if its support exceeds a threshold. For a set of suspicious rules, a drill-down operation is performed to find the raw data in the audit trail that gave rise to these rules, and these rules are fed to a decision tree that classifies the suspicious activity either as a known or an unknown attack type.

Level-Wise Mining

Many attacks occur at the application level and can be finished in a single connection. It becomes difficult to detect these by using association rules or frequent episode rules since the

support of the generated rule may not be enough to pass the threshold. If the threshold is decreased to that level, many false alarms may be generated. To overcome this, Lee, Stolfo, and Mok [13] proposed a level-wise mining technique that first finds the episodes related to high frequency axis attribute values. Then the support threshold is iteratively reduced to find the episodes related to the low frequency axis values by restricting the participation of the old axis values that already have output episodes.

Performance Issues

A major shortcoming of the current IDSs that employ data mining methods is that they can give a series of false alarms in case of a noticeable systems environment modification and a user can deceive the system by slowly changing behavior patterns. There can be two types of false alarms in classifying system activities in case of any deviation from normal patterns: *false positives* and *false negatives*. False positive alarms are issued when normal behaviors are incorrectly identified as abnormal and false negative alarms are issued when abnormal behaviors are incorrectly identified as normal. Though it's important to keep both types of false alarm rates as low as possible, the

False negative alarms should be the minimum to ensure the security of the system. To overcome this limitation, an IDS must be capable of adapting to the changing conditions typical of an intrusion detection environment. For example, in an academic environment, the behavior patterns at the beginning of a semester may be different than the behavior patterns at the middle/end of the semester. If the system builds its profile based on the audit data gathered during the early semester days, then the system may give a series of false alarms at the later stages of the semester. System security administrators can tune the IDS by adjusting the profile. But it may require frequent human intervention. Again, the patterns of intrusions may be dynamic. Intruders may change their strategies over time and the normal system activities may change because of modifications to work practices. Moreover, it is not always possible to predict the level of intrusions in the future. So it is important that an IDS should have automatic adaptability to new conditions. Otherwise, an IDS may start to lose its edge. Such adaptability can be achieved by employing incremental mining techniques. Such an adaptive system should use real time data (log of audit records) to constantly update the profile. One straightforward approach can be to regenerate the user profile with the new audit data. But this would not be a computationally feasible approach. When the current usage profile is compared with the initial profile, there can be different types of deviation as mentioned in section 2.1. Each of these deviations can represent an intrusion or a change in behavior. In case of a change in system behaviors, the base profile must be updated with the corresponding change so that it doesn't give any false positives alarms in future. So the system needs to decide whether to make a change or reject it.

If the system tries to make a change to the base profile every time it sees a deviation, there is a potential danger of incorporating intrusive activities into the profile. The IDS must be able to adapt to these changes while still recognizing abnormal activities and not adapt to those. If both an intrusion and behavior change occur during a particular time interval, it becomes more complicated. Again, which rules to add, which to remove, is critical. Moreover, there are more issues that need to be addressed in case of updating. The system should adapt to rapid changes as well as gradual changes in system behavior. Selecting the time interval at which the update should take place is also an important issue. If the interval is too long, the system may miss some rapid changes or short-term attacks. If the interval is too small, the system may miss some long-

term changes. So, we consider two problems as the major issues in developing a true adaptive intrusion detection system. One is to select the time when the update should be made. The other is to select a mechanism to update the profile. To tackle the first issue, we can trace the similarity pattern found by comparing each day's activities with the base profile. If the similarity goes down the threshold line and experiences a sharp shift, we would consider that as an abnormal behavior. If the similarity goes down the threshold line, but does not experience a sharp shift, rather experiences a slow downwards trend, we would
Consider that as a possible change in behavior. It is not computationally feasible to archive audit data for a long time. So we may employ a sliding window technique to update the base profile. We can assume that system activities before a certain period of time are too old to characterize the current behavior, i.e., the audit records before that period are unlikely to contribute towards the rules that represent system activities. We can define a sliding window [*t*1, *t*2, ... , *tn*] of *n* days. As mentioned in Toivonen [14], we would maintain both the large item sets and the negative border. As time goes on, a large item set may start losing its support and an item set in the negative border may start gaining support. We would discard some large item sets in the process and include some new item sets. The update technique would reject transactions outside the sliding window as they are assumed to be old and outdated. We can use different techniques [15,16] to update the profile rule set.

Intrusion Detection: From a Data Warehouse Perspective

The data mining techniques view intrusion detection from a data analysis perspective. Their goal is to mine rules from audit trail records obtained from system activities. To make the mining process efficient, the system has to deal with a large amount of data. Data warehousing [17] techniques have been developed to deal with large amounts of data having multiple dimensions. Moreover, they provide on-line analytical processing (OLAP) that can be used in conjunction with data mining [18]. In this section, a rough architecture for an IDS will be proposed that combines data warehouse and data mining technologies. The idea partly comes from [19] where the authors used OLAP and mining to understand user access patterns from web access. Our architecture is based upon a multi-dimensional array structure, called a *data cube* [20,21]. The data cube will be built to aggregate the number of connections. A data cube can have numerous dimensions where each dimension represents an attribute with all possible values of the attribute. The data values of an attribute may be related hierarchically that specify aggregation levels and granularity of viewing data. For example, *minute*®*hour*®*day* is a hierarchy on *time* that specifies various aggregation levels. Hierarchies can be pre-defined or generated by partitioning the dimension into ranges. For example, the dimension *connection_duration* can be partitioned into categories like *low*, *med*, and *high*. Each dimension has rows for each distinct value (to store the count) plus one row for storing the sum of the counts. Here count refers to the number of connections. To build a audit log data cube, we may have the following dimensions: *timestamp* (defined on a hierarchy *minute*®*hour*®*day*), *duration* (defined on a range hierarchy *low*®*med*®*high*), *service* (possibly defined on a pre-built hierarchy), *src_host* (defined on a domain hierarchy), *dest_host* (defined on a domain hierarchy), *src_bytes* (defined on a range hierarchy *low*®*med*®*high*), *dest_bytes* (defined on a range hierarchy *low*®*med*®*high*), *flag* (possibly defined on a pre-built hierarchy). It may not be necessary to define a concept hierarchy for each dimension. But doing that would certainly give an extra edge for analysis purpose.

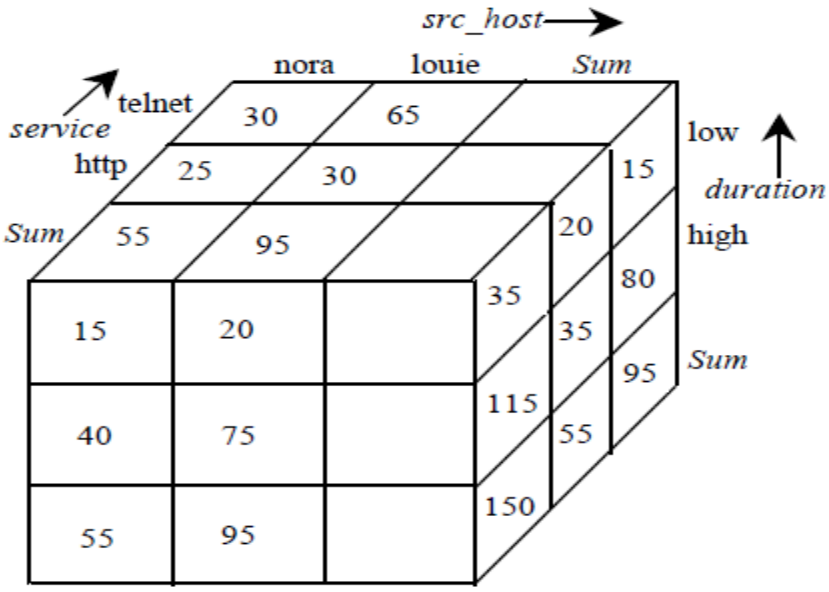

Figure 1: A 3-D data cube

Let us consider the 3-D cube shown in Fig 1. It contains three dimensions: *service*, *src_host*, and *duration*. Data is generally aggregated at a coarse level, and then at successively finer levels. Table 1 is a relational representation of the cube of Fig 1. Data is aggregated by *service*, then by *src_host*, and then by *duration*. The data is aggregated at three levels. A typical query on such a data cube can be "Give the total number of connections for each telnet service for each host with a low connection duration". Data can be manipulated with great flexibility and viewed from different perspectives by the use of data cubes. OLAP operations like *roll-up*, *drilldown*, and *slice-and-dice* [17] can be applied on the data. These operations offer analytical modeling capabilities. Going from specific to general by climbing up the aggregation hierarchy is called *roll-up*. Going from generalized data to more specific by stepping down the aggregation hierarchy is called *drill-down*. *Slice-and-dice* operation reduces the dimensionality of data by projecting the data on a subset of dimensions for selected values of other dimensions. A typical example can be the sub-cube obtained by the projection: *src_host_domain* □□"edu".

It is possible to discover implicit knowledge in the audit data by applying data mining techniques like *association*, *prediction*, *classification and clustering* [18] on the audit log data cube. Next we illustrate how association analysis can be performed on a data cube. A data cube can offer added flexibility and efficiency in association rule mining. It simplifies the process of grouping data according to one or a set of dimensions. Aggregate values like "count" facilitates the mining process; and moreover multi-level association can be mined by moving along the hierarchy in any dimension. Two types of associations can be mined in a data-cube: *inter-dimension* and *intra-dimension* [18].

Inter dimension association provides the association among different dimensions of the data cube. For example, from Table 1, we can infer *service*□*telnet* □□*duration*□*high*. Intra-dimension association provides the association with regard to one or a set of reference dimensions by grouping the remaining set of dimensions into a transaction like set. A typical example can be "The telnet service from Nora is significantly higher on Mondays compared to Sundays".

Service	*Src_host*	*duration*	*Number of connections by service by src_host by duration*	*Number of connections by service by src_host*	*Number of connections by service*
telnet	nora	low	5		
		high	25		
				30	
	louie	low	10		
		high	55		
				65	
					95
http	nora	low	10		
		high	15		
				25	
	louie	low	10		
		high	20		
				30	
					55

Table 1: Aggregation levels for a data cube

Moreover, constraint-based association rule mining can also be performed. This may turn out to be very helpful in our intrusion detection paradigm where we may be interested in the relation between specific system features. In addition to association, important operations like *prediction* can also be applied which may give added capabilities to an IDS.

Conclusion

Data mining methods provide automatic intrusion detection capabilities. They mine knowledge from audit data to characterize normal and abnormal user behavior. Some of the works done to-date were introduced in the paper. One of the major limitations of the systems is that they lack adaptability to changing behavior patterns. Some technical issues were discussed which are critical in developing a true adaptive, real-time intrusion detection system. As the data mining methods have to deal with large amount of data, a combination of data warehousing and data mining technologies can be utilized for intrusion detection purposes. Such a model was proposed in the paper that can apply mining programs on top of a data warehouse built with audit data. The idea presented was preliminary in nature. An attempt was made to exemplify it rather than going for any formalism or experimentation. It can be investigated for further exploration.

References

- B. Mukherjee, L. Heberlein, and K. Levitt, Network intrusion detection, *IEEE Network*, 8(3), 1994, 26-41.
- R. Graham, FAQ: Network intrusion detection systems, 1999. (Downloaded from http://www.infosyssec.com/infosyssec/netintrufaq.htm)
- Sundaram, An introduction to intrusion detection, *Crossroads: The ACM Student Magazine*, 2(4), 1996.
- G. Piatetsky-Shapiro and W. Frawley, *Knowledge Discovery in Databases* (Menlo Park, CA: AAAI Press/ The MIT Press, 1991).
- R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large databases, *Proc. 1993 ACM SIGMOD International Conf. on Management of Data*, Washingtion, DC, 1993, 207-216.
- W. Lee and S. Stolfo, Data mining approaches for intrusion detection, *Proc. 7th USENIX Security Symposium* (*SECURITY '98*), San Antonio, TX, 1998, 79-94.

- H. Mannila and H. Toivonen, Discovering generalized episodes using minimal occurrences, *Proc. 2nd International Conf. on Knowledge Discovery and Data Mining* (*KDD '96*), Portland, OR, 1996, 146-151.
- F. Shi, *Genetic algorithms for feature selection in an intrusion detection application*, masters thesis, Mississippi State University, Mississippi State, MS, 2000.
- R. Mukkamala, J. Gagnon, and S. Jajodia, Integrating data mining techniques with intrusion detection, *Proc. 13th Annual IFIP WG 11.3 Working Conf. on Database Security*, Seattle, WA, 1999.
- J. Luo and S. Bridges, Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection, *International Journal of Intelligent Systems*, 15(8), 2000, 687-703.
- S. Bridges and R. Vaughn, Fuzzy data mining and genetic algorithms applied to intrusion detection, *Proc. 23rd National Information Systems Security Conf.*, Baltimore, MA, 2000.
- D. Barbara, S. Jajodia, and N. Wu, Mining unexpected rules in network audit trails, Personal communications, 2000.
- W. Lee, S. Stolfo, and K. Mok, Mining audit data to build intrusion detection models, *Proc. 4th International Conf. on Knowledge Discovery and Data Mining* (*KDD '98*), New York City, NY, 1998, 66-72.
- H. Toivonen, Sampling large databases for association rules, *Proc. 22nd international conf. on very large data bases* (*VLDB'96*), Mumbai, India, 1996, 134-145.
- S. Lee and D. Cheung, Maintenance of discovered association rules: When to update?, *Proc. 1997 ACMSIGMOD workshop on research issues on data mining and knowledge discovery* (*DMKD'97*), Tucson, AZ, 1997.
- S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, An efficient algorithm for the incremental updation of association rules in large databases, *Proc. 3rd international conf. on knowledge discovery and data mining* (*KDD'97*), Newport Beach, CA, 1997, 263- 266.
- S. Chaudhuri and U. Dayal, An overview of data warehousing and OLAP technology, *SIGMOD Record*, 26(1), 1997, 65-74.
- J. Han, Towards on-line analytical mining in large databases, *SIGMOD Record*, 27(1), 1998, 97-107.
- O. Zaiane, M. Xin, and J. Han, Discovering web access patterns and trends by applying OLAP and data mining technology on web logs, *Proc. IEEE Forum on Research and Technology Advances in Digital Libraries* (*ADL '98*), Santa Barbara, CA, 1998, 19-29.
- R. Agrawal, A. Gupta, and S. Sarawagi, Modeling multidimensional databases, *Proc. 13th International Conf. on Data Engineering* (*ICDE '97*), Birmingham, UK, 1997, 232-243.
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals, *Data Mining and Knowledge Discovery*, 1(1), 1997, 29-53.